

Oracle8*i*

PL/SQL パッケージ・プロシージャ リファレンス

リリース 8.1

2000 年 2 月

Oracle8i PL/SQL パッケージ・プロシージャ リファレンス, リリース 8.1

原本名: Oracle8i Supplied PL/SQL Packages Reference, Release 2 (8.1.6)

原本著者: Michele Cyran.

原本協力者: Mark Bauer, D. Alpern, G. Arora, N. Bhatt, S. Chandrasekar, T. Chang, G. Claborn, R. Decker, A. Downing, J. Draaijer, J. Finnerty, R. Frank, A. Ganesh, J. Gosselin, R. Govindarajan, B. Goyal, S. Harris, B. Himatsingka, C. Iyer, H. Jakobsson, A. Jasuja, M. Jungerman, P. Justus, A. Kalra, M. Kamath, S. Kotsovolos, V. Krishnamurthy, J. Krishnaswamy, J. Kundu, B. Lee, J. Liu, P. Locke, A. Logan, N. Mallavarupu, J. Mallory, R. Mani, S. Mavris, A. Mozes, J. Muller, C. Murray, K. Muthukkaruppan, S. Muthulingam, R. Pang, R. Park, G. Pongracz, T. Portfolio, L. Price, S. Puranik, M. Ramacher, S. Ramakrishnan, D. Raphaely, S. Ray, A. Rhee, S. Samu, U. Sangam, A. Saxena, J. Sharma, E. Soylemez, A. Srinivasan, J. Srinivasan, I. Stocks, R. Sujithan, A. Swaminathan, K. Tarkhanov, A. Tsukerman, A. To, S. Urman, S. Vivian, D. Voss, W. Wang, R. Ward, S. Wertheimer, R. Wessman, J. Wijaya, D. Wong, L. Wu, R. Yaseen

Copyright © 1996, 1999, Oracle Corporation. All rights reserved.

Printed in Japan.

制限付権利の説明

プログラム (ソフトウェアおよびドキュメントを含む) の使用、複製または開示は、オラクル社との契約に記された制約条件に従うものとします。著作権、特許権およびその他の知的財産権に関する法律により保護されています。

当プログラムのリバース・エンジニアリング等は禁止されています。

このドキュメントの情報は、予告なしに変更されることがあります。オラクル社は本ドキュメントの無謬性を保証しません。

* オラクル社とは、Oracle Corporation (米国オラクル) または日本オラクル株式会社 (日本オラクル) を指します。

危険な用途への使用について

オラクル社製品は、原子力、航空産業、大量輸送、医療あるいはその他の危険が伴うアプリケーションを用途として開発されておりません。オラクル社製品を上述のようなアプリケーションに使用することについての安全確保は、顧客各位の責任と費用により行ってください。万一かかる用途での使用によりクレームや損害が発生いたしましても、日本オラクル株式会社と開発元である Oracle Corporation (米国オラクル) およびその関連会社は一切責任を負いかねます。当プログラムを米国国防総省の米国政府機関に提供する際には、『Restricted Rights』と共に提供してください。この場合次の Notice が適用されます。

Restricted Rights Notice

Programs delivered subject to the DOD FAR Supplement are "commercial computer software" and use, duplication, and disclosure of the Programs, including documentation, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement. Otherwise, Programs delivered subject to the Federal Acquisition Regulations are "restricted computer software" and use, duplication, and disclosure of the Programs shall be subject to the restrictions in FAR 52.227-19, Commercial Computer Software - Restricted Rights (June, 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065.

このドキュメントに記載されているその他の会社名および製品名は、あくまでその製品および会社を識別する目的にのみ使用されており、それぞれの所有者の商標または登録商標です。

目次

はじめに	xxxi
------------	------

1 オラクル社が提供するパッケージ

パッケージの概要	1-2
オラクル社が提供するパッケージの使用方法	1-2
新規パッケージの作成	1-3
パッケージ内容の参照	1-5
オラクル社が提供するパッケージの一覧	1-6
サブリメンタル・パッケージ内のサブプログラム	1-13
Calendar	1-13
SDO_ADMIN	1-15
SDO_GEOM	1-15
SDO_MIGRATE	1-15
SDO_TUNE	1-15
TimeScale	1-16
TimeSeries	1-17
TSTools	1-20
UTL_PG	1-21
Vir_Pkg	1-22

2 DBMS_ALERT

セキュリティ	2-2
定数	2-2
エラー	2-2

アラートの使用方法	2-3
サブプログラムの要約	2-4
REGISTER プロシージャ	2-4
REMOVE プロシージャ	2-5
REMOVEALL プロシージャ	2-5
SET_DEFAULTS プロシージャ	2-6
SIGNAL プロシージャ	2-6
WAITANY プロシージャ	2-7
WAITONE プロシージャ	2-8
例	2-9

3 DBMS_APPLICATION_INFO

サブプログラムの要約	3-2
SET_MODULE プロシージャ	3-2
SET_ACTION プロシージャ	3-3
READ_MODULE プロシージャ	3-4
SET_CLIENT_INFO プロシージャ	3-5
READ_CLIENT_INFO プロシージャ	3-6
SET_SESSION_LONGOPS プロシージャ	3-6

4 DBMS_AQ

Java クラス	4-2
列挙定数	4-2
データ構造	4-2
サブプログラムの要約	4-11
ENQUEUE プロシージャ	4-11
DEQUEUE プロシージャ	4-13
LISTEN プロシージャ	4-15

5 DBMS_AQADM

列挙定数	5-2
サブプログラムの要約	5-2
CREATE_QUEUE_TABLE プロシージャ	5-4
ALTER_QUEUE_TABLE プロシージャ	5-7

DROP_QUEUE_TABLE プロシージャ	5-8
CREATE_QUEUE プロシージャ	5-9
CREATE_NP_QUEUE プロシージャ	5-11
ALTER_QUEUE プロシージャ	5-12
DROP_QUEUE プロシージャ	5-13
START_QUEUE プロシージャ	5-14
STOP_QUEUE プロシージャ	5-15
GRANT_SYSTEM_PRIVILEGE プロシージャ	5-16
REVOKE_SYSTEM_PRIVILEGE プロシージャ	5-17
GRANT_QUEUE_PRIVILEGE プロシージャ	5-17
REVOKE_QUEUE_PRIVILEGE プロシージャ	5-18
ADD_SUBSCRIBER プロシージャ	5-19
ALTER_SUBSCRIBER プロシージャ	5-20
REMOVE_SUBSCRIBER プロシージャ	5-20
SCHEDULE_PROPAGATION プロシージャ	5-21
UNSCHEDULE_PROPAGATION プロシージャ	5-23
VERIFY_QUEUE_TYPES プロシージャ	5-23
ALTER_PROPAGATION_SCHEDULE プロシージャ	5-24
ENABLE_PROPAGATION_SCHEDULE プロシージャ	5-26
DISABLE_PROPAGATION_SCHEDULE プロシージャ	5-26
MIGRATE_QUEUE_TABLE プロシージャ	5-27

6 DBMS_BACKUP_RESTORE

Windows NT プラットフォームの Oracle に対するファイル名の正規化	6-2
---	-----

7 DBMS_DDL

要件	7-2
サブプログラムの要約	7-2
ALTER_COMPILE プロシージャ	7-2
ANALYZE_OBJECT プロシージャ	7-3

8 DBMS_DEBUG

DBMS_DEBUG の使用方法	8-2
使用上の注意	8-5

型	8-6
定数	8-8
エラー・コード	8-11
例外	8-12
変数	8-12
サブプログラムの要約	8-13
共通セクション	8-14
PROBE_VERSION プロシージャ	8-14
SELF_CHECK プロシージャ	8-15
SET_TIMEOUT ファンクション	8-16
TARGET_SESSION セクション	8-16
INITIALIZE ファンクション	8-17
DEBUG_ON プロシージャ	8-17
DEBUG_OFF プロシージャ	8-18
デバッグ・セッション・セクション	8-18
ATTACH_SESSION プロシージャ	8-19
SYNCHRONIZE ファンクション	8-20
SHOW_SOURCE プロシージャ	8-21
PRINT_BACKTRACE プロシージャ	8-23
CONTINUE ファンクション	8-24
SET_BREAKPOINT ファンクション	8-25
DELETE_BREAKPOINT ファンクション	8-26
DISABLE_BREAKPOINT ファンクション	8-27
ENABLE_BREAKPOINT ファンクション	8-28
SHOW_BREAKPOINTS プロシージャ	8-28
GET_VALUE ファンクション	8-29
SET_VALUE ファンクション	8-32
DETACH_SESSION プロシージャ	8-34
GET_RUNTIME_INFO ファンクション	8-34
GET_INDEXES ファンクション	8-35
EXECUTE プロシージャ	8-36
PRINT_INSTANTIATIONS プロシージャ	8-39
TARGET_PROGRAM_RUNNING プロシージャ	8-39
PING プロシージャ	8-40
SET_TIMEOUT_BEHAVIOUR プロシージャ	8-40

OER ブレーク・ポイント	8-41
---------------------	------

9 DBMS_DEFER

DBMS_DEFER パッケージ	9-2
サブプログラムの要約	9-2
CALL プロシージャ	9-3
COMMIT_WORK プロシージャ	9-4
datatype_ARG プロシージャ	9-5
TRANSACTION プロシージャ	9-7

10 DBMS_DEFER_QUERY

DBMS_DEFER_QUERY パッケージ	10-2
サブプログラムの要約	10-2
GET_ARG_FORM ファンクション	10-3
GET_ARG_TYPE ファンクション	10-4
GET_CALL_ARGS プロシージャ	10-6
GET_datatype_ARG ファンクション	10-7

11 DBMS_DEFER_SYS

DBMS_DEFER_SYS パッケージ	11-2
サブプログラムの要約	11-2
ADD_DEFAULT_DEST プロシージャ	11-4
DELETE_DEFAULT_DEST プロシージャ	11-4
DELETE_DEF_DESTINATION プロシージャ	11-5
DELETE_ERROR プロシージャ	11-5
DELETE_TRAN プロシージャ	11-6
DISABLED ファンクション	11-7
EXCLUDE_PUSH プロシージャ	11-8
EXECUTE_ERROR プロシージャ	11-9
EXECUTE_ERROR_AS_USER プロシージャ	11-10
PURGE ファンクション	11-11
PUSH ファンクション	11-13
REGISTER_PROPAGATOR プロシージャ	11-16
SCHEDULE_PURGE プロシージャ	11-17

SCHEDULE_PUSH プロシージャ	11-19
SET_DISABLED プロシージャ	11-21
UNREGISTER_PROPAGATOR プロシージャ	11-22
UNSCHEDULE_PURGE プロシージャ	11-23
UNSCHEDULE_PUSH プロシージャ	11-23

12 DBMS_DESCRIBE

セキュリティ	12-2
型	12-2
エラー	12-2
サブプログラムの要約	12-2
DESCRIBE_PROCEDURE プロシージャ	12-3
例	12-6

13 DBMS_DISTRIBUTED_TRUST_ADMIN

要件	13-2
サブプログラムの要約	13-2
ALLOW_ALL プロシージャ	13-2
ALLOW_SERVER プロシージャ	13-3
DENY_ALL プロシージャ	13-4
DENY_SERVER プロシージャ	13-4
例	13-5

14 DBMS_HS

例外	14-2
サブプログラムの要約	14-6
ALTER_BASE_CAPS プロシージャ	14-8
ALTER_BASE_DD プロシージャ	14-9
ALTER_CLASS_CAPS プロシージャ	14-9
ALTER_CLASS_DD プロシージャ	14-9
ALTER_CLASS_INIT プロシージャ	14-9
ALTER_FDS_CLASS プロシージャ	14-10
ALTER_FDS_INST プロシージャ	14-10
ALTER_INST_CAPS プロシージャ	14-10

ALTER_INST_DD プロシージャ	14-11
ALTER_INST_INIT プロシージャ	14-11
COPY_CLASS プロシージャ	14-12
COPY_INST プロシージャ	14-12
CREATE_BASE_CAPS プロシージャ	14-12
CREATE_BASE_DD プロシージャ	14-12
CREATE_CLASS_CAPS プロシージャ	14-12
CREATE_CLASS_DD プロシージャ	14-13
CREATE_CLASS_INIT プロシージャ	14-13
CREATE_FDS_CLASS プロシージャ	14-14
CREATE_FDS_INST プロシージャ	14-14
CREATE_INST_CAPS プロシージャ	14-14
CREATE_INST_DD プロシージャ	14-14
CREATE_INST_INIT プロシージャ	14-15
DROP_BASE_CAPS プロシージャ	14-15
DROP_BASE_DD プロシージャ	14-16
DROP_CLASS_CAPS プロシージャ	14-16
DROP_CLASS_DD プロシージャ	14-16
DROP_CLASS_INIT プロシージャ	14-16
DROP_FDS_CLASS プロシージャ	14-17
DROP_FDS_INST プロシージャ	14-17
DROP_INST_CAPS プロシージャ	14-17
DROP_INST_DD プロシージャ	14-17
DROP_INST_INIT プロシージャ	14-18
REPLACE_BASE_CAPS プロシージャ	14-18
REPLACE_BASE_DD プロシージャ	14-18
REPLACE_CLASS_CAPS プロシージャ	14-19
REPLACE_CLASS_DD プロシージャ	14-19
REPLACE_CLASS_INIT プロシージャ	14-19
REPLACE_FDS_CLASS プロシージャ	14-20
REPLACE_FDS_INST プロシージャ	14-20
REPLACE_INST_CAPS プロシージャ	14-20
REPLACE_INST_DD プロシージャ	14-21
REPLACE_INST_INIT プロシージャ	14-21

15 DBMS_HS_PASSTHROUGH

セキュリティ	15-2
サブプログラムの要約	15-2
BIND_VARIABLE プロシージャ	15-3
BIND_VARIABLE_RAW プロシージャ	15-4
BIND_OUT_VARIABLE プロシージャ	15-5
BIND_OUT_VARIABLE_RAW プロシージャ	15-7
BIND_INOUT_VARIABLE プロシージャ	15-8
BIND_INOUT_VARIABLE_RAW プロシージャ	15-9
CLOSE_CURSOR プロシージャ	15-10
EXECUTE_IMMEDIATE プロシージャ	15-11
EXECUTE_NON_QUERY ファンクション	15-12
FETCH_ROW ファンクション	15-13
GET_VALUE プロシージャ	15-14
GET_VALUE_RAW プロシージャ	15-16
OPEN_CURSOR ファンクション	15-17
PARSE プロシージャ	15-17

16 DBMS_IOT

サブプログラムの要約	16-2
BUILD_CHAIN_ROWS_TABLE プロシージャ	16-2
BUILD_EXCEPTIONS_TABLE プロシージャ	16-3

17 DBMS_JOB

要件	17-2
サブプログラムの要約	17-2
SUBMIT プロシージャ	17-3
REMOVE プロシージャ	17-4
CHANGE プロシージャ	17-5
WHAT プロシージャ	17-6
NEXT_DATE プロシージャ	17-7
INSTANCE プロシージャ	17-7
INTERVAL プロシージャ	17-8
BROKEN プロシージャ	17-9

RUN プロシージャ	17-9
USER_EXPORT プロシージャ	17-10
USER_EXPORT プロシージャ	17-11
Oracle Parallel Server での DBMS_JOB パッケージの使用法	17-12

18 DBMS_LOB

要件	18-2
LOB ロケータ	18-2
データ型	18-3
定数	18-3
例外	18-4
セキュリティ	18-4
規則および制限事項	18-5
BFILE 特有の規則および制限事項	18-7
テンポラリ LOB	18-9
テンポラリ LOB の使用上の注意	18-11
テンポラリ LOB の例外	18-12
サブプログラムの要約	18-13
APPEND プロシージャ	18-14
CLOSE プロシージャ	18-16
COMPARE ファンクション	18-17
COPY プロシージャ	18-20
CREATETEMPORARY プロシージャ	18-22
ERASE プロシージャ	18-23
FILECLOSE プロシージャ	18-25
FILECLOSEALL プロシージャ	18-26
FILEEXISTS ファンクション	18-27
FILEGETNAME プロシージャ	18-29
FILEISOPEN ファンクション	18-30
FILEOPEN プロシージャ	18-32
FREETEMPORARY プロシージャ	18-33
GETCHUNKSIZE ファンクション	18-34
GETLENGTH ファンクション	18-35
INSTR ファンクション	18-37
ISOPEN ファンクション	18-40

ISTEMPORARY ファンクション	18-41
LOADFROMFILE プロシージャ	18-42
OPEN プロシージャ	18-44
READ プロシージャ	18-45
SUBSTR ファンクション	18-49
TRIM プロシージャ	18-52
WRITE プロシージャ	18-53
WRITEAPPEND プロシージャ	18-56

19 DBMS_LOCK

要件	19-2
セキュリティ	19-2
ロックの表示と監視	19-2
定数	19-3
サブプログラムの要約	19-4
ALLOCATE_UNIQUE プロシージャ	19-4
REQUEST ファンクション	19-6
CONVERT ファンクション	19-7
RELEASE ファンクション	19-9
SLEEP プロシージャ	19-10
例	19-11

20 DBMS_LOGMNR

LogMiner の使用方法	20-2
定数	20-2
プレース・ホルダ列の使用法	20-3
サブプログラムの要約	20-5
ADD_LOGFILE プロシージャ	20-6
START_LOGMNR プロシージャ	20-7
END_LOGMNR プロシージャ	20-10
例	20-10

21 DBMS_LOGMNR_D

ディクショナリ・ファイルの作成	21-2
-----------------------	------

サブプログラムの要約	21-2
BUILD プロシージャ	21-2
例	21-3
22 DBMS_MVIEW	
DBMS_SNAPSHOT のシノニムとしての DBMS_MVIEW	22-1
23 DBMS_OBFUSCATION_TOOLKIT	
DESEncrypt プロシージャ	23-2
パラメータの説明	23-2
暗号化プロシージャの制限事項	23-2
DESDecrypt プロシージャ	23-3
パラメータの説明	23-3
DESDecryption プロシージャの制限事項	23-4
DESEncryption と DESDecryption のコード例	23-4
24 DBMS_OFFLINE_OG	
DBMS_OFFLINE_OG パッケージ	24-2
サブプログラムの要約	24-2
BEGIN_INSTANTIATION プロシージャ	24-3
BEGIN_LOAD プロシージャ	24-4
END_INSTANTIATION プロシージャ	24-5
END_LOAD プロシージャ	24-7
RESUME_SUBSET_OF_MASTERS プロシージャ	24-8
25 DBMS_OFFLINE_SNAPSHOT	
DBMS_OFFLINE_SNAPSHOT パッケージ	25-2
サブプログラムの要約	25-2
BEGIN_LOAD プロシージャ	25-3
END_LOAD プロシージャ	25-5
26 DBMS_OLAP	
要件	26-2

エラー・メッセージ	26-2
サブプログラムの要約	26-3
ESTIMATE_SUMMARY_SIZE プロシージャ	26-3
EVALUATE_UTILIZATION プロシージャ	26-4
EVALUATE_UTILIZATION_W プロシージャ	26-4
RECOMMEND_MV プロシージャ	26-5
RECOMMEND_MV_W プロシージャ	26-7
VALIDATE_DIMENSION プロシージャ	26-9
DBMS_OLAP インタフェース表	26-10

27 DBMS_ORACLE_TRACE_AGENT

セキュリティ	27-2
サブプログラムの要約	27-2
SET_ORACLE_TRACE_IN_SESSION プロシージャ	27-2
例	27-3

28 DBMS_ORACLE_TRACE_USER

サブプログラムの要約	28-2
SET_ORACLE_TRACE プロシージャ	28-2
例	28-2

29 DBMS_OUTPUT

セキュリティ	29-2
エラー	29-2
型	29-2
DBMS_OUTPUT の使用方法	29-2
サブプログラムの要約	29-3
ENABLE プロシージャ	29-3
DISABLE プロシージャ	29-4
PUT および PUT_LINE プロシージャ	29-5
NEW_LINE プロシージャ	29-6
GET_LINE および GET_LINES プロシージャ	29-7
例	29-8

30 DBMS_PCLXUTIL

DBMS_PCLXUTIL の使用方法	30-2
制限事項	30-3
サブプログラムの要約	30-3
BUILD_PART_INDEX プロシージャ	30-4
例	30-4

31 DBMS_PIPE

パブリック・パイプ	31-2
プライベート・パイプ	31-2
パイプの使用方法	31-3
セキュリティ	31-4
定数	31-4
エラー	31-4
サブプログラムの要約	31-4
CREATE_PIPE ファンクション	31-5
PACK_MESSAGE プロシージャ	31-7
SEND_MESSAGE ファンクション	31-8
RECEIVE_MESSAGE ファンクション	31-10
NEXT_ITEM_TYPE ファンクション	31-12
UNPACK_MESSAGE プロシージャ	31-13
REMOVE_PIPE ファンクション	31-14
PURGE プロシージャ	31-15
RESET_BUFFER プロシージャ	31-16
UNIQUE_SESSION_NAME ファンクション	31-17
例	31-17

32 DBMS_PROFILER

DBMS_PROFILER の使用方法	32-2
要件	32-3
収集されたデータ	32-3
セキュリティの考慮事項	32-6
サブプログラムの要約	32-6
例外生成の 2 つの方法	32-6

例外	32-6
エラー・コード	32-7
START_PROFILER ファンクション	32-8
STOP_PROFILER ファンクション	32-9
FLUSH_DATA ファンクション	32-9
PAUSE_PROFILER ファンクション	32-9
RESUME_PROFILER ファンクション	32-9
GET_VERSION プロシージャ	32-10
INTERNAL_VERSION_CHECK ファンクション	32-10

33 DBMS_RANDOM

要件	33-2
サブプログラムの要約	33-2
INITIALIZE プロシージャ	33-2
SEED プロシージャ	33-3
RANDOM ファンクション	33-3
TERMINATE プロシージャ	33-3

34 DBMS_RECTIFIER_DIFF

DBMS_RECTIFIER_DIFF パッケージ	34-2
サブプログラムの要約	34-2
DIFFERENCES プロシージャ	34-3
RECTIFY プロシージャ	34-6

35 DBMS_REFRESH

DBMS_REFRESH パッケージ	35-2
サブプログラムの要約	35-2
ADD プロシージャ	35-3
CHANGE プロシージャ	35-4
DESTROY プロシージャ	35-6
MAKE プロシージャ	35-7
REFRESH プロシージャ	35-9
SUBTRACT プロシージャ	35-10

36 DBMS_REPAIR

セキュリティ	36-2
列挙型	36-2
例外	36-2
サブプログラムの要約	36-4
ADMIN_TABLES プロシージャ	36-4
CHECK_OBJECT プロシージャ	36-5
DUMP_ORPHAN_KEYS プロシージャ	36-7
FIX_CORRUPT_BLOCKS プロシージャ	36-9
REBUILD_FREELISTS プロシージャ	36-10
SKIP_CORRUPT_BLOCKS プロシージャ	36-11

37 DBMS_REPCAT

DBMS_REPCAT パッケージ	37-2
サブプログラムの要約	37-2
ADD_GROUPED_COLUMN プロシージャ	37-6
ADD_MASTER_DATABASE プロシージャ	37-7
ADD_PRIORITY_datatype プロシージャ	37-9
ADD_SITE_PRIORITY_SITE プロシージャ	37-10
ADD_conflicttype_RESOLUTION プロシージャ	37-12
ALTER_MASTER_PROPAGATION プロシージャ	37-16
ALTER_MASTER_REPOBJECT プロシージャ	37-17
ALTER_PRIORITY プロシージャ	37-19
ALTER_PRIORITY_datatype プロシージャ	37-20
ALTER_SITE_PRIORITY プロシージャ	37-22
ALTER_SITE_PRIORITY_SITE プロシージャ	37-23
ALTER_SNAPSHOT_PROPAGATION プロシージャ	37-24
CANCEL_STATISTICS プロシージャ	37-25
COMMENT_ON_COLUMN_GROUP プロシージャ	37-26
COMMENT_ON_PRIORITY_GROUP/COMMENT_ON_SITE_PRIORITY プロシージャ	37-27
COMMENT_ON_REPGROUP プロシージャ	37-28
COMMENT_ON_REPOBJECT プロシージャ	37-29
COMMENT_ON_REPSITES プロシージャ	37-31
COMMENT_ON_SNAPSHOT_REPSITES プロシージャ	37-32

COMMENT_ON_conflicttype_RESOLUTION プロシージャ	37-33
COMPARE_OLD_VALUES プロシージャ	37-35
CREATE_MASTER_REPGROUP プロシージャ	37-37
CREATE_MASTER_REPOBJECT プロシージャ	37-38
CREATE_SNAPSHOT_REPGROUP プロシージャ	37-41
CREATE_SNAPSHOT_REPOBJECT プロシージャ	37-42
DEFINE_COLUMN_GROUP プロシージャ	37-44
DEFINE_PRIORITY_GROUP プロシージャ	37-45
DEFINE_SITE_PRIORITY プロシージャ	37-47
DO_DEFERRED_REPCAT_ADMIN プロシージャ	37-48
DROP_COLUMN_GROUP プロシージャ	37-49
DROP_GROUPED_COLUMN プロシージャ	37-50
DROP_MASTER_REPGROUP プロシージャ	37-51
DROP_MASTER_REPOBJECT プロシージャ	37-52
DROP_PRIORITY プロシージャ	37-53
DROP_PRIORITY_GROUP プロシージャ	37-54
DROP_PRIORITY_datatype プロシージャ	37-55
DROP_SITE_PRIORITY プロシージャ	37-57
DROP_SITE_PRIORITY_SITE プロシージャ	37-58
DROP_SNAPSHOT_REPGROUP プロシージャ	37-59
DROP_SNAPSHOT_REPOBJECT プロシージャ	37-60
DROP_conflicttype_RESOLUTION プロシージャ	37-61
EXECUTE_DDL プロシージャ	37-63
GENERATE_REPLICATION_SUPPORT プロシージャ	37-64
GENERATE_SNAPSHOT_SUPPORT プロシージャ	37-66
MAKE_COLUMN_GROUP プロシージャ	37-68
PURGE_MASTER_LOG プロシージャ	37-69
PURGE_STATISTICS プロシージャ	37-70
REFRESH_SNAPSHOT_REPGROUP プロシージャ	37-71
REGISTER_SNAPSHOT_REPGROUP プロシージャ	37-72
REGISTER_STATISTICS プロシージャ	37-74
RELOCATE_MASTERDEF プロシージャ	37-75
REMOVE_MASTER_DATABASES プロシージャ	37-76
REPCAT_IMPORT_CHECK プロシージャ	37-77
RESUME_MASTER_ACTIVITY プロシージャ	37-78

SEND_OLD_VALUES プロシージャ	37-79
SET_COLUMNS プロシージャ	37-81
SUSPEND_MASTER_ACTIVITY プロシージャ	37-83
SWITCH_SNAPSHOT_MASTER プロシージャ	37-83
UNREGISTER_SNAPSHOT_REPGROUP プロシージャ	37-84
VALIDATE ファンクション	37-85
WAIT_MASTER_LOG プロシージャ	37-88

38 DBMS_REPCAT_ADMIN

DBMS_REPCAT_ADMIN パッケージ	38-2
サブプログラムの要約	38-2
GRANT_ADMIN_ANY_SCHEMA プロシージャ	38-3
GRANT_ADMIN_SCHEMA プロシージャ	38-3
REGISTER_USER_REPGROUP プロシージャ	38-4
REVOKE_ADMIN_ANY_SCHEMA プロシージャ	38-6
REVOKE_ADMIN_SCHEMA プロシージャ	38-7
UNREGISTER_USER_REPGROUP プロシージャ	38-8

39 DBMS_REPCAT_INSTANTIATE

DBMS_REPCAT_INSTANTIATE パッケージ	39-2
サブプログラムの要約	39-2
DROP_SITE_INSTANTIATION プロシージャ	39-3
INSTANTIATE_OFFLINE ファンクション	39-3
INSTANTIATE_OFFLINE_REPAPI ファンクション	39-6
INSTANTIATE_ONLINE ファンクション	39-9

40 DBMS_REPCAT_RGT

DBMS_REPCAT_RGT パッケージ	40-2
サブプログラムの要約	40-2
ALTER_REFRESH_TEMPLATE プロシージャ	40-5
ALTER_TEMPLATE_OBJECT プロシージャ	40-6
ALTER_TEMPLATE_PARM プロシージャ	40-9
ALTER_USER_AUTHORIZATION プロシージャ	40-11
ALTER_USER_PARM_VALUE プロシージャ	40-12

COMPARE_TEMPLATES ファンクション	40-15
COPY_TEMPLATE ファンクション	40-16
CREATE_OBJECT_FROM_EXISTING ファンクション	40-18
CREATE_REFRESH_TEMPLATE ファンクション	40-20
CREATE_TEMPLATE_OBJECT ファンクション	40-22
CREATE_TEMPLATE_PARM ファンクション	40-26
CREATE_USER_AUTHORIZATION ファンクション	40-28
CREATE_USER_PARM_VALUE ファンクション	40-29
DELETE_RUNTIME_PARMs プロシージャ	40-32
DROP_ALL_OBJECTS プロシージャ	40-33
DROP_ALL_TEMPLATE_PARMs プロシージャ	40-34
DROP_ALL_TEMPLATE_SITES プロシージャ	40-35
DROP_ALL_TEMPLATES プロシージャ	40-36
DROP_ALL_USER_AUTHORIZATIONS プロシージャ	40-36
DROP_ALL_USER_PARM_VALUES プロシージャ	40-37
DROP_REFRESH_TEMPLATE プロシージャ	40-39
DROP_SITE_INSTANTIATION プロシージャ	40-40
DROP_TEMPLATE_OBJECT プロシージャ	40-41
DROP_TEMPLATE_PARM プロシージャ	40-43
DROP_USER_AUTHORIZATION プロシージャ	40-44
DROP_USER_PARM_VALUE プロシージャ	40-45
GET_RUNTIME_PARM_ID ファンクション	40-46
INSERT_RUNTIME_PARMs プロシージャ	40-47
INstantiate_OFFLINE ファンクション	40-49
INstantiate_OFFLINE_REPAPI ファンクション	40-51
INstantiate_ONLINE ファンクション	40-54
LOCK_TEMPLATE_EXCLUSIVE プロシージャ	40-57
LOCK_TEMPLATE_SHARED プロシージャ	40-57

41 DBMS_REPUTIL

DBMS_REPUTIL パッケージ	41-2
サブプログラムの要約	41-2
REPLICATION_OFF プロシージャ	41-3
REPLICATION_ON プロシージャ	41-3
REPLICATION_IS_ON ファンクション	41-4

FROM_REMOTE ファンクション	41-4
GLOBAL_NAME ファンクション	41-5
MAKE_INTERNAL_PKG プロシージャ	41-5
SYNC_UP_REP プロシージャ	41-6

42 DBMS_RESOURCE_MANAGER

要件	42-2
サブプログラムの要約	42-2
CREATE_PLAN プロシージャ	42-4
UPDATE_PLAN プロシージャ	42-5
DELETE_PLAN プロシージャ	42-5
DELETE_PLAN_CASCADE プロシージャ	42-6
CREATE_CONSUMER_GROUP プロシージャ	42-7
UPDATE_CONSUMER_GROUP プロシージャ	42-7
DELETE_CONSUMER_GROUP プロシージャ	42-8
CREATE_PLAN_DIRECTIVE プロシージャ	42-9
UPDATE_PLAN_DIRECTIVE プロシージャ	42-10
DELETE_PLAN_DIRECTIVE プロシージャ	42-11
CREATE_PENDING_AREA プロシージャ	42-12
VALIDATE_PENDING_AREA プロシージャ	42-13
CLEAR_PENDING_AREA プロシージャ	42-13
SUBMIT_PENDING_AREA プロシージャ	42-14
SET_INITIAL_CONSUMER_GROUP プロシージャ	42-17
SWITCH_CONSUMER_GROUP_FOR_SESS プロシージャ	42-18
SWITCH_CONSUMER_GROUP_FOR_USER プロシージャ	42-19

43 DBMS_RESOURCE_MANAGER_PRIVS

サブプログラムの要約	43-2
GRANT_SYSTEM_PRIVILEGE プロシージャ	43-2
REVOKE_SYSTEM_PRIVILEGE プロシージャ	43-3
GRANT_SWITCH_CONSUMER_GROUP プロシージャ	43-4
REVOKE_SWITCH_CONSUMER_GROUP プロシージャ	43-5

44 DBMS_RLS

動的な述語	44-2
セキュリティ	44-3
使用上の注意	44-3
サブプログラムの要約	44-3
ADD_POLICY プロシージャ	44-4
DROP_POLICY プロシージャ	44-6
REFRESH_POLICY プロシージャ	44-6
ENABLE_POLICY プロシージャ	44-7
例	44-8

45 DBMS_ROWID

使用上の注意	45-2
要件	45-3
ROWID のタイプ	45-3
例外	45-4
サブプログラムの要約	45-4
ROWID_CREATE ファンクション	45-5
ROWID_INFO プロシージャ	45-6
ROWID_TYPE ファンクション	45-7
ROWID_OBJECT ファンクション	45-8
ROWID_RELATIVE_FNO ファンクション	45-9
ROWID_BLOCK_NUMBER ファンクション	45-10
ROWID_ROW_NUMBER ファンクション	45-10
ROWID_TO_ABSOLUTE_FNO ファンクション	45-11
ROWID_TO_EXTENDED ファンクション	45-12
ROWID_TO_RESTRICTED ファンクション	45-14
ROWID_VERIFY ファンクション	45-14

46 DBMS_SESSION

要件	46-2
サブプログラムの要約	46-2
SET_ROLE プロシージャ	46-3
SET_SQL_TRACE プロシージャ	46-3

SET_NLS プロシージャ	46-4
CLOSE_DATABASE_LINK プロシージャ	46-4
RESET_PACKAGE プロシージャ	46-5
UNIQUE_SESSION_ID ファンクション	46-6
IS_ROLE_ENABLED ファンクション	46-7
IS_SESSION_ALIVE ファンクション	46-7
SET_CLOSE_CACHED_OPEN_CURSORS プロシージャ	46-8
FREE_UNUSED_USER_MEMORY プロシージャ	46-8
SET_CONTEXT プロシージャ	46-11
LIST_CONTEXT プロシージャ	46-11
SWITCH_CURRENT_CONSUMER_GROUP プロシージャ	46-12
例	46-14

47 DBMS_SHARED_POOL

インストレーション時の注意	47-2
使用上の注意	47-2
サブプログラムの要約	47-2
SIZES プロシージャ	47-2
KEEP プロシージャ	47-3
UNKEEP プロシージャ	47-5
ABORTED_REQUEST_THRESHOLD プロシージャ	47-5

48 DBMS_SNAPSHOT

DBMS_SNAPSHOT パッケージ	48-2
サブプログラムの要約	48-2
BEGIN_TABLE_REORGANIZATION プロシージャ	48-3
END_TABLE_REORGANIZATION プロシージャ	48-3
I_AM_A_REFRESH ファンクション	48-4
PURGE_DIRECT_LOAD_LOG プロシージャ	48-4
PURGE_LOG プロシージャ	48-5
PURGE_SNAPSHOT_FROM_LOG プロシージャ	48-6
REFRESH プロシージャ	48-8
REFRESH_ALL_MVIEWS プロシージャ	48-11
REFRESH_DEPENDENT プロシージャ	48-12

REGISTER_SNAPSHOT プロシージャ	48-14
UNREGISTER_SNAPSHOT プロシージャ	48-16

49 DBMS_SPACE

セキュリティ	49-2
要件	49-2
サブプログラムの要約	49-2
UNUSED_SPACE プロシージャ	49-2
FREE_BLOCKS プロシージャ	49-4
例	49-5

50 DBMS_SPACE_ADMIN

セキュリティ	50-2
定数	50-2
サブプログラムの要約	50-3
SEGMENT_VERIFY プロシージャ	50-4
SEGMENT_CORRUPT プロシージャ	50-5
SEGMENT_DROP_CORRUPT プロシージャ	50-6
SEGMENT_DUMP プロシージャ	50-6
TABLESPACE_VERIFY プロシージャ	50-7
TABLESPACE_FIX_BITMAPS プロシージャ	50-8
TABLESPACE_REBUILD_BITMAPS プロシージャ	50-9
TABLESPACE_REBUILD_QUOTAS プロシージャ	50-9
TABLESPACE_MIGRATE_FROM_LOCAL プロシージャ	50-10
TABLESPACE_MIGRATE_TO_LOCAL プロシージャ	50-11
TABLESPACE_RELOCATE_BITMAPS プロシージャ	50-12
TABLESPACE_FIX_SEGMENT_STATES プロシージャ	50-13
詳細な例	50-13

51 DBMS_SQL

DBMS_SQL の使用方法	51-2
定数	51-3
型	51-3
例外	51-4

実行フロー	51-4
セキュリティ	51-7
サブプログラムの要約	51-8
OPEN_CURSOR ファンクション	51-10
PARSE プロシージャ	51-10
BIND_VARIABLE および BIND_ARRAY プロシージャ	51-13
問合せの処理	51-17
DEFINE_COLUMN プロシージャ	51-17
DEFINE_ARRAY プロシージャ	51-19
DEFINE_COLUMN_LONG プロシージャ	51-21
EXECUTE ファンクション	51-21
EXECUTE_AND_FETCH ファンクション	51-22
FETCH_ROWS ファンクション	51-22
COLUMN_VALUE プロシージャ	51-23
COLUMN_VALUE_LONG プロシージャ	51-25
VARIABLE_VALUE プロシージャ	51-26
更新、挿入、削除の処理	51-28
IS_OPEN ファンクション	51-28
DESCRIBE_COLUMNS プロシージャ	51-29
CLOSE_CURSOR プロシージャ	51-31
エラーの位置	51-32
LAST_ERROR_POSITION ファンクション	51-32
LAST_ROW_COUNT ファンクション	51-32
LAST_ROW_ID ファンクション	51-33
LAST_SQL_FUNCTION_CODE ファンクション	51-33
例	51-34

52 DBMS_STATS

DBMS_STATS の使用方法	52-2
型	52-2
サブプログラムの要約	52-3
統計情報の設定または取得	52-5
PREPARE_COLUMN_VALUES プロシージャ	52-5
SET_COLUMN_STATS プロシージャ	52-8
SET_INDEX_STATS プロシージャ	52-9

SET_TABLE_STATS プロシージャ	52-11
CONVERT_RAW_VALUE プロシージャ	52-12
GET_COLUMN_STATS プロシージャ	52-13
GET_INDEX_STATS プロシージャ	52-15
GET_TABLE_STATS プロシージャ	52-16
DELETE_COLUMN_STATS プロシージャ	52-17
DELETE_INDEX_STATS プロシージャ	52-18
DELETE_TABLE_STATS プロシージャ	52-19
DELETE_SCHEMA_STATS プロシージャ	52-21
DELETE_DATABASE_STATS プロシージャ	52-21
統計情報の転送	52-22
CREATE_STAT_TABLE プロシージャ	52-23
DROP_STAT_TABLE プロシージャ	52-24
EXPORT_COLUMN_STATS プロシージャ	52-24
EXPORT_INDEX_STATS プロシージャ	52-25
EXPORT_TABLE_STATS プロシージャ	52-26
EXPORT_SCHEMA_STATS プロシージャ	52-27
EXPORT_DATABASE_STATS プロシージャ	52-28
IMPORT_COLUMN_STATS プロシージャ	52-29
IMPORT_INDEX_STATS プロシージャ	52-30
IMPORT_TABLE_STATS プロシージャ	52-31
IMPORT_SCHEMA_STATS プロシージャ	52-32
IMPORT_DATABASE_STATS プロシージャ	52-32
オプティマイザ統計情報の収集	52-33
GATHER_INDEX_STATS プロシージャ	52-34
GATHER_TABLE_STATS プロシージャ	52-35
GATHER_SCHEMA_STATS プロシージャ	52-37
GATHER_DATABASE_STATS プロシージャ	52-39
GENERATE_STATS プロシージャ	52-42
例	52-43

53 DBMS_TRACE

要件	53-2
制限事項	53-2

定数	53-2
DBMS_TRACE の使用方法	53-2
サブプログラムの要約	53-5
SET_PLSQL_TRACE プロシージャ	53-6
CLEAR_PLSQL_TRACE プロシージャ	53-6
PLSQL_TRACE_VERSION プロシージャ	53-7

54 DBMS_TRANSACTION

要件	54-2
サブプログラムの要約	54-2
READ_ONLY プロシージャ	54-3
READ_WRITE プロシージャ	54-3
ADVISE_ROLLBACK プロシージャ	54-3
ADVISE_NOthing プロシージャ	54-4
ADVISE_COMMIT プロシージャ	54-4
USE_ROLLBACK_SEGMENT プロシージャ	54-4
COMMIT_COMMENT プロシージャ	54-5
COMMIT_FORCE プロシージャ	54-5
COMMIT プロシージャ	54-6
SAVEPOINT プロシージャ	54-6
ROLLBACK プロシージャ	54-7
ROLLBACK_SAVEPOINT プロシージャ	54-7
ROLLBACK_FORCE プロシージャ	54-8
BEGIN_DISCRETE_TRANSACTION プロシージャ	54-8
PURGE_MIXED プロシージャ	54-9
PURGE_LOST_DB_ENTRY プロシージャ	54-10
LOCAL_TRANSACTION_ID ファンクション	54-12
STEP_ID ファンクション	54-12

55 DBMS_TTS

例外	55-2
サブプログラムの要約	55-2
TRANSPORT_SET_CHECK プロシージャ	55-2
DOWNGRADE プロシージャ	55-3

56 DBMS_UTILITY

要件	56-2
型	56-2
サブプログラムの要約	56-2
COMPILE_SCHEMA プロシージャ	56-4
ANALYZE_SCHEMA プロシージャ	56-5
ANALYZE_DATABASE プロシージャ	56-6
FORMAT_ERROR_STACK ファンクション	56-7
FORMAT_CALL_STACK ファンクション	56-7
IS_PARALLEL_SERVER ファンクション	56-8
GET_TIME ファンクション	56-8
GET_PARAMETER_VALUE ファンクション	56-9
NAME_RESOLVE プロシージャ	56-10
NAME_TOKENIZE プロシージャ	56-12
COMMA_TO_TABLE プロシージャ	56-12
TABLE_TO_COMMA プロシージャ	56-13
PORT_STRING ファンクション	56-14
DB_VERSION プロシージャ	56-14
MAKE_DATA_BLOCK_ADDRESS ファンクション	56-15
DATA_BLOCK_ADDRESS_FILE ファンクション	56-16
DATA_BLOCK_ADDRESS_BLOCK ファンクション	56-16
GET_HASH_VALUE ファンクション	56-17
ANALYZE_PART_OBJECT プロシージャ	56-18
EXEC_DDL_STATEMENT プロシージャ	56-19
CURRENT_INSTANCE ファンクション	56-19
ACTIVE_INSTANCES プロシージャ	56-20

57 DEBUG_EXTPROC

要件	57-2
インストレーション時の注意	57-2
DEBUG_EXTPROC の使用方法	57-2
サブプログラムの要約	57-3
STARTUP_EXTPROC_AGENT プロシージャ	57-3

58 OUTLN_PKG

要件	58-2
セキュリティ	58-2
サブプログラムの要約	58-2
DROP_UNUSED プロシージャ	58-2
DROP_BY_CAT プロシージャ	58-3
UPDATE_BY_CAT プロシージャ	58-3

59 UTL_COLL

サブプログラムの要約	59-2
IS_LOCATOR ファンクション	59-2
例	59-3

60 UTL_FILE

セキュリティ	60-2
ファイルの所有権と保護	60-3
例（UNIX 固有）	60-3
型	60-4
例外	60-4
サブプログラムの要約	60-5
FOPEN ファンクション	60-6
IS_OPEN ファンクション	60-7
FCLOSE プロシージャ	60-8
FCLOSE_ALL プロシージャ	60-8
GET_LINE プロシージャ	60-9
PUT プロシージャ	60-10
NEW_LINE プロシージャ	60-11
PUT_LINE プロシージャ	60-12
PUTF プロシージャ	60-12
FFLUSH プロシージャ	60-14
FOPEN ファンクション	60-14

61 UTL_HTTP

例外	61-2
----------	------

使用上の注意	61-3
サブプログラムの要約	61-3
REQUEST ファンクション	61-3
REQUEST_PIECES ファンクション	61-5
例	61-6

62 UTL_INADDR

インターネット・アドレス・パッケージ	62-2
get_host_name()	62-2
get_host_address()	62-2
例外	62-3

63 UTL_RAW

使用上の注意	63-2
サブプログラムの要約	63-2
CONCAT ファンクション	63-3
CAST_TO_RAW ファンクション	63-4
CAST_TO_VARCHAR2 ファンクション	63-5
LENGTH ファンクション	63-6
SUBSTR ファンクション	63-7
TRANSLATE ファンクション	63-8
TRANSLITERATE ファンクション	63-10
OVERLAY ファンクション	63-11
COPIES ファンクション	63-13
XRANGE ファンクション	63-14
REVERSE ファンクション	63-15
COMPARE ファンクション	63-16
CONVERT ファンクション	63-17
BIT_AND ファンクション	63-19
BIT_OR ファンクション	63-20
BIT_XOR ファンクション	63-21
BIT_COMPLEMENT ファンクション	63-22

64 UTL_REF

要件	64-2
データ型	64-2
例外	64-2
セキュリティ	64-3
サブプログラムの要約	64-4
SELECT_OBJECT プロシージャ	64-4
LOCK_OBJECT プロシージャ	64-5
UPDATE_OBJECT プロシージャ	64-6
DELETE_OBJECT プロシージャ	64-7
例	64-8

65 UTL_SMTP

connection	65-2
reply、replies	65-3
open_connection()	65-3
command()、command_replies()	65-4
helo()	65-5
ehlo()	65-6
mail()	65-6
rcpt()	65-7
data()	65-8
open_data()、write_data()、write_raw_data()、close_data()	65-9
rset()	65-10
vrfy()	65-11
noop()	65-12
quit()	65-12
例外	65-13
制限事項	65-14
応答コード	65-14
例	65-16

66 UTL_TCP

TCP/IP パッケージ (UTL_TCP)	66-2
------------------------------	------

connection	66-2
CRLF	66-3
open_connection()	66-3
available()	66-5
read_raw()	66-6
write_raw()	66-7
read_text()	66-7
write_text()	66-8
read_line()	66-9
write_line()	66-10
get_raw(), get_text(), get_line()	66-11
flush()	66-12
close_connection()	66-12
close_all_connections()	66-13
例外	66-14
例	66-14

索引

はじめに

このマニュアルでは、付属の Oracle PL/SQL パッケージについて説明します。このマニュアルの情報は、特に指定されていない限りは、すべてのプラットフォームで実行される Oracle Server の各バージョンに対して適用されます。

関連項目： Java パッケージの詳細は、『Oracle8i Java パッケージ・プロ
シージャ リファレンス』を参照してください。

この章では、次の項目について説明します。

- [パッケージの概要](#)
- [Oracle8i の新機能](#)
- [対象読者](#)
- [関連ドキュメント](#)
- [表記規則](#)

パッケージの概要

パッケージとは、論理的に関連している PL/SQL の型、アイテムおよびサブプログラムをグループ化するスキーマ・オブジェクトです。パッケージには通常、仕様部と本体の 2 つの部分があります（本体が不要な場合もあります）。仕様部はアプリケーションへのインタフェースで、使用可能な型、変数、定数、例外、カーソルおよびサブプログラムを宣言します。本体はカーソルとサブプログラムを完全に定義し、仕様部を完全にインプリメントします。

パッケージは、サブプログラムとは異なり、コールしたり、パラメータ化したり、ネスト化することはできません。ただし、パッケージの書式は、サブプログラムと類似しています。

```
CREATE PACKAGE name AS -- specification (visible part)
    -- public type and item declarations
    -- subprogram specifications
END [name];
```

```
CREATE PACKAGE BODY name AS -- body (hidden part)
    -- private type and item declarations
    -- subprogram bodies
[BEGIN
    -- initialization statements]
END [name];
```

仕様部には、アプリケーションで参照できるパブリック宣言が含まれています。本体には、インプリメンテーションの詳細とプライベート宣言が含まれていますが、アプリケーションからは参照できません。

パッケージ本体へのインタフェース（パッケージ仕様部）を変更せずに、パッケージ本体をデバッグ、拡張および置換できます。

パッケージを作成し、Oracle データベースに格納するためには、CREATE PACKAGE および CREATE PACKAGE BODY 文を使用します。これらの文は、SQL*Plus または Enterprise Manager から対話形式で実行できます。

アプリケーションから参照およびアクセスできるのは、パッケージ仕様部の宣言部のみです。パッケージ本体にあるインプリメンテーションの詳細は参照およびアクセスできません。このため、本体（インプリメンテーション）は、コール側のプログラムを再コンパイルせずに変更できます。

パッケージを使用する利点には、モジュール性、アプリケーション設計の容易性、情報の非公開、機能性の追加およびパフォーマンスの向上などがあります。

Oracle8i の新機能

Oracle8i には、8.1.5 と 8.1.6 の 2 つのサブリリースがあります。

リリース 8.1.5

このマニュアルは、当初リリース 8.1.5 を対象に作成されました。

リリース 8.1.6

次の 5 つの新規パッケージがあります。

- DBMS_BACKUP_RESTORE
- DBMS_OBFUSCATION_TOOLKIT
- UTL_INADDR
- UTL_SMTP
- UTL_TCP

次のパッケージが拡張されました。

- DBMS_DEBUG
- DBMS_DISTRIBUTED_TRUST_ADMIN
- DBMS_LOGMNR
- DBMS_LOGMNR_D
- DBMS_PCLXUTIL
- DBMS_PROFILER
- DBMS_REPAIR
- DBMS_RESOURCE_MANAGER
- DBMS_ROWID
- DBMS_SQL
- DBMS_UTILITY
- UTL_HTTP

対象読者

このマニュアルは、Oracle パッケージを使用しているか、または Oracle パッケージに関心のあるすべての方を対象にしています。また、システム・アナリスト、プロジェクト・マネージャおよびデータベース・アプリケーションの開発やチューニングに関心のある方々にも有効です。

このマニュアルでは、読者にアプリケーション・プログラミングについての作業知識があること、リレーショナル・データベース・システムの情報にアクセスするための構造化問合せ言語（SQL）の使用経験が十分にあることを前提としています。

一部の項では、オブジェクト指向プログラミングの基本概念に関する知識も必要とします。

関連ドキュメント

詳細は、Oracle8i ドキュメント・セットにある次のドキュメントを参照してください。

- 『Oracle8i アプリケーション開発者ガイド 基礎編』
- 『Oracle8i PL/SQL ユーザーズ・ガイドおよびリファレンス』

表記規則

このマニュアルでは次の表記規則を使用します。

表記	意味
...	省略符号が文またはコマンド内で使用されている場合は、例に直接関係のない部分が省略されていることを示します。
英大文字のテキスト	パッケージ名、コマンド・キーワード、データベース・オブジェクト名およびファイル名などを強調するために使用します。
コード例	SQL、PL/SQL および SQL*Plus のコマンドまたはコード文は固定幅フォントで表示されます。 例： INSERT INTO emp (empno, ename) VALUES (1000, 'SMITH'); ALTER TABLESPACE users ADD DATAFILE 'users2.ora' SIZE 50K;
< >	山カッコは、ユーザーが指定する必要のある名前を示します。
[]	大カッコは選択が任意の項目を示します。選択肢の中から 1 つ選択するか、または何も入力しなくてもかまいません。
\$	ドル記号は、OpenVMS の DIGITAL CommandLanguage プロンプトおよび Digital UNIX の Bourne シェル・プロンプトを表します。

オラクル社が提供するパッケージ

オラクル社は、Oracle Server で使用するパッケージを多数提供しています。このパッケージを使用すると、データベースの機能性の拡張や PL/SQL による SQL 機能へのアクセスが可能になります。この機能性を利用してアプリケーションを作成したり、独自のストアド・プロシージャを作成することもできます。

注意： オラクル社では、Oracle Developer や Oracle Application Server などの様々な製品に関するパッケージを提供していますが、このマニュアルでは、データベース・サーバーに関するパッケージのみ紹介します。

この章では、次の項目について説明します。

- [パッケージの概要](#)
- [オラクル社が提供するパッケージの一覧](#)
- [サブリメンタル・パッケージ内のサブプログラム](#)

関連項目： ユーザー独自のパッケージ作成方法の詳細は、『Oracle8i アプリケーション開発者ガイド 基礎編』を参照してください。

パッケージの概要

パッケージとは、関連するプログラム・オブジェクトをカプセル化した集合体で、データベースにまとめて格納されています。プログラム・オブジェクトには、プロシージャ、ファンクション、変数、定数、カーソルおよび例外があります。

パッケージには、スタンドアロンのプロシージャやファンクションに比べて多くの利点があります。主な利点は次のとおりです。

- アプリケーション開発をより効率的に計画できます。
- 権限をより効率的に付与できます。
- 依存スキーマ・オブジェクトを再コンパイルせずに、パッケージ・オブジェクトを変更できます。
- Oracle で、複数のパッケージ・オブジェクトをメモリーに一度に読み込むことができます。
- プロシージャまたはファンクションをオーバーロードできます。オーバーロードとは、同一パッケージ内に同じ名前でもプロシージャを複数作成し、各プロシージャでは異なる数またはデータ型の引数を使用することです。
- パッケージ内のすべてのプロシージャとファンクションで使用できるグローバルな変数とカーソルを含めることができます。

オラクル社が提供するパッケージの使用方法

オラクル社が提供するパッケージの大部分は、データベースが作成され、CATPROC.SQL スクリプトが実行されるときに自動的にインストールされます。たとえば、DBMS_ALERT パッケージを作成するには、ユーザー SYS で接続したときに DBMSALRT.SQL と PRVTALRT.PLB スクリプトを実行する必要があります。ただし、これらのスクリプトは、CATPROC.SQL スクリプトによって自動的に実行されます。

一部のパッケージは、自動的にインストールされません。これらのパッケージをインストールするには、該当する各章の指示に従ってください。

SQL から PL/SQL ファンクションをコールするには、コールするユーザーがそのファンクションの所有者であるか、またはユーザーにそのファンクションの EXECUTE 権限が付与されている必要があります。PL/SQL ファンクションで定義されたビューを検索するには、そのビューの SELECT 権限が必要です。ビューを検索するには個々の EXECUTE 権限は必要ありません。パッケージに関する特別要件については、それぞれの章の記述を参照してください。

新規パッケージの作成

新規パッケージの作成には、次の2つの手順があります。

1. CREATE PACKAGE 文でパッケージ仕様部を作成します。

パッケージ仕様部にはプログラム・オブジェクトを宣言できます。ここで宣言したオブジェクトは、パブリック・オブジェクトと呼ばれます。パブリック・オブジェクトは、パッケージ内の別のオブジェクトからも、パッケージの外部からも参照できます。

注意： CREATE PACKAGE 文に OR REPLACE 句を追加するとさらに便利になる場合があります。

2. CREATE PACKAGE BODY 文でパッケージ本体を作成します。

パッケージ本体にはプログラム・オブジェクトを宣言および定義できます。

- パッケージ仕様部で宣言したパブリック・オブジェクトをパッケージ本体で定義する必要があります。
- 追加のパッケージ・オブジェクトを宣言および定義できます。このオブジェクトは、プライベート・オブジェクトと呼ばれます。プライベート・オブジェクトは、パッケージ仕様部ではなくパッケージ本体で宣言されるため、パッケージ内の他のオブジェクトからのみ参照できます。パッケージの外部からは参照できません。

関連項目： 新規パッケージ作成の詳細は、『Oracle8i PL/SQL ユーザーズ・ガイドおよびリファレンス』および『Oracle8i アプリケーション開発者ガイド 基礎編』を参照してください。パッケージの格納と実行の詳細は、『Oracle8i 概要』を参照してください。

仕様部と本体の分離

パッケージの仕様部では、パブリックな型、変数、定数およびサブプログラムを宣言します。これらは、そのパッケージの有効範囲外から参照できます。パッケージの本体では、仕様部で宣言されたオブジェクトおよびパッケージ外部のアプリケーションからは参照できないプライベート・オブジェクトを定義します。

パッケージの仕様部と本体は、データベースに別々に格納されます。パブリック・プログラム・オブジェクトをコールまたは参照するその他のスキーマ・オブジェクトはパッケージ仕様部にのみ依存し、パッケージ本体には依存しません。仕様部と本体が分かれているため、パッケージ本体にあるプログラム・オブジェクトの定義を変更しても、そのプログラム・オブジェクトをコールまたは参照するその他のスキーマ・オブジェクトを無効にすることはありません。パッケージ仕様部にあるプログラム・オブジェクトの宣言を変更した場合のみ、それに依存しているスキーマ・オブジェクトが無効になります。

例 次の例は、EMPLOYEE_MANAGEMENT という名前のパッケージのパッケージ仕様部です。このパッケージには、ストアド・ファンクションが1つ、ストアド・プロシージャが2つ含まれています。

```
CREATE PACKAGE employee_management AS
    FUNCTION hire_emp (name VARCHAR2, job VARCHAR2,
        mgr NUMBER, hiredate DATE, sal NUMBER, comm NUMBER,
        deptno NUMBER) RETURN NUMBER;
    PROCEDURE fire_emp (emp_id NUMBER);
    PROCEDURE sal_raise (emp_id NUMBER, sal_incr NUMBER);
END employee_management;
```

このパッケージの本体では、ファンクションとプロシージャが定義されます。

```
CREATE PACKAGE BODY employee_management AS
    FUNCTION hire_emp (name VARCHAR2, job VARCHAR2,
        mgr NUMBER, hiredate DATE, sal NUMBER, comm NUMBER,
        deptno NUMBER) RETURN NUMBER IS
```

ファンクションでは、従業員番号を除き、フィールドに対するすべての引数を従業員表に挿入します。フィールドに対する値は順番に設定されます。このファンクションへのコールによって生成された順序番号が戻されます。

```
        new_empno    NUMBER(10);

    BEGIN
        SELECT emp_sequence.NEXTVAL INTO new_empno FROM dual;
        INSERT INTO emp VALUES (new_empno, name, job, mgr,
            hiredate, sal, comm, deptno);
        RETURN (new_empno);
    END hire_emp;

    PROCEDURE fire_emp(emp_id IN NUMBER) AS
```

プロシージャは、引数 emp_id に対応する従業員番号を持つ従業員を削除します。該当する従業員が見つからない場合は、例外が呼び出されます。

```
    BEGIN
        DELETE FROM emp WHERE empno = emp_id;
        IF SQL%NOTFOUND THEN
            raise_application_error(-20011, 'Invalid Employee
                Number: ' || TO_CHAR(emp_id));
        END IF;
    END fire_emp;

    PROCEDURE sal_raise (emp_id IN NUMBER, sal_incr IN NUMBER) AS
```

プロシージャは2つの引数を受け入れます。emp_id は従業員番号に対応する番号です。sal_incr は、その従業員の給料増額分です。

```
BEGIN

-- If employee exists, then update salary with increase.

UPDATE emp
  SET sal = sal + sal_incr
  WHERE empno = emp_id;
IF SQL%NOTFOUND THEN
  raise_application_error(-20011, 'Invalid Employee
    Number: ' || TO_CHAR(emp_id));
END IF;
END sal_raise;
END employee_management;
```

注意： この例を実際に試す場合は、最初に順序 emp_sequence を作成してください。次の SQL*Plus 文で作成できます。

```
SQL> CREATE SEQUENCE emp_sequence
> START WITH 8000 INCREMENT BY 10;
```

パッケージ内容の参照

パッケージ仕様部で宣言された型、アイテム、サブプログラムを参照するには、ドット表記法を使用します。たとえば、次のようになります。

```
package_name.type_name
package_name.item_name
package_name.subprogram_name
```

オラクル社が提供するパッケージの一覧

この項では、オラクル社が提供する各サーバー・パッケージを一覧にして、それぞれの詳細が記述されている参照先を示します。これらのパッケージは、パッケージ所有者ではなく起動ユーザーとして実行されます。特に注記がない限り、パッケージは同じ名前のパブリック・シノニムを介してコールできます。

- 注意：
- これらのパッケージが提供するプロシージャとファンクションおよびその外部インタフェースはオラクル社が所有するもので、変更される可能性があります。
 - オラクル社が提供するパッケージは変更しないでください。変更すると、内部エラーやデータベースのセキュリティ違反の原因となる場合があります。

表 1-1 オラクル社が提供するパッケージの一覧

パッケージ名	説明	参照先
Calendar (表の最後の注 2 を参照)	カレンダー・メンテナンス・ファンクションを提供します。	『Oracle8i Time Series ユーザーズ・ガイド』
DBMS_ALERT	データベース・イベントの非同期通知をサポートします。	第 2 章
DBMS_APPLICATION_INFO	監査またはパフォーマンス追跡の目的で、アプリケーション名をデータベースに登録できます。	第 3 章
DBMS_AQ	(事前定義されたオブジェクト型の) メッセージをキューに追加したり、デキューすることができます。	第 4 章
DBMS_AQADM	キューまたはキュー表にある管理ファンクションを、事前定義されたオブジェクト型のメッセージに対して実行できます。	第 5 章
DBMS_BACKUP_RESTORE	Windows NT プラットフォームでファイル名を正規化します。	第 6 章
DBMS_DDL	ストアド・プロシージャから一部の SQL DDL 文へのアクセスを提供し、DDL では使用できない特殊な管理操作を提供します。	第 7 章
DBMS_DEBUG	Oracle Server における PL/SQL デバッガ・レイヤー、プロンプトへの PL/SQL の API です。	第 8 章

表 1-1 オラクル社が提供するパッケージの一覧

パッケージ名	説明	参照先
DBMS_DEFER	レプリケート・トランザクション遅延リモート・プロシージャ・コール (RPC) 機能へのユーザー・インタフェースを提供します。分散オプションが必要です。	第 9 章
DBMS_DEFER_QUERY	ビューでは参照できない遅延リモート・プロシージャ・コール (RPC) のキュー・データの問合せを許可します。分散オプションが必要です。	第 10 章
DBMS_DEFER_SYS	レプリケート・トランザクション遅延リモート・プロシージャ・コール (RPC) 機能へのシステム管理者用インタフェースを提供します。分散オプションが必要です。	第 11 章
DBMS_DESCRIBE	ストアド・プロシージャの引数をフルネーム変換とセキュリティ・チェックとともに記述します。	第 12 章
DBMS_DISTRIBUTED_TRUST_ADMIN	Trusted Database リストをメンテナンスします。このリストは、特定のサーバーからの権限データベース・リンクが受入れ可能かどうかを判断するために使用されます。	第 13 章
DBMS_HS	異機種間サービス・ディクショナリ内のオブジェクトを作成および変更できます。	第 14 章
DBMS_HS_PASSTHROUGH	異機種間サービスを使用して、パススルー SQL 文を非 Oracle システムに送信できます。	第 15 章
DBMS_IOT	ANALYZE コマンドを使用して、索引構成表の連鎖行への参照を設定できる表を作成します。	第 16 章
DBMS_JOB	定期的に行う管理プロシージャをスケジュールできます。ジョブ・キュー用のインタフェースでもあります。	第 17 章
DBMS_LOB	Oracle ラージ・オブジェクト (Oracle Large Object: LOB) データ型 - BLOB、CLOB (読み込み / 書き込み) および BFILE (読み取り専用) における操作に対する汎用ルーチンを提供します。	第 18 章
DBMS_LOCK	Oracle ロック・マネージメント・サービスを使用して、ロックの要求、変換および解放を実行できます。	第 19 章
DBMS_LOGMNR	ログ・リーダーを初期化および実行するためのファンクションを提供します。	第 20 章
DBMS_LOGMNR_D	現行のデータベースのディクショナリ表を問い合わせ、その内容を含んだテキスト・ベースのファイルを作成します。	第 21 章

表 1-1 オラクル社が提供するパッケージの一覧

パッケージ名	説明	参照先
DBMS_MVIEW	DBMS_SNAPSHOT のシノニムです。	第 22 章
DBMS_OBFUSCATION_TOOLKIT	データ暗号化規格に対するプロシージャを提供します。	第 23 章
DBMS_OFFLINE_OG	マスター・グループのオフライン・インスタンスエーション用のパブリック API を提供します。	第 24 章
DBMS_OFFLINE_SNAPSHOT	スナップショットのオフライン・インスタンスエーション用のパブリック API を提供します。	第 25 章
DBMS_OLAP	サマリー、ディメンションおよびクエリー・リライト用のプロシージャを提供します。	第 26 章
DBMS_ORACLE_TRACE_AGENT	Oracle7 Server 内の Oracle TRACE インストルメンテーションにクライアントがコールできるインタフェースを提供します。	第 27 章
DBMS_ORACLE_TRACE_USER	Oracle7 Server の Oracle TRACE インストルメンテーションへのパブリック・アクセスをコール・ユーザーに提供します。	第 28 章
DBMS_OUTPUT	情報をバッファに蓄積して後で取り出せるようにします。	第 29 章
DBMS_PCLXUTIL	パーティション単位でローカル索引を作成するためのパーティション内の並列性を提供します。	第 30 章
DBMS_PIPE	セッション間のメッセージ送信を可能にする DBMS パイプ・サービスを提供します。	第 31 章
DBMS_PROFILER	既存の PL/SQL アプリケーションをプロファイルし、パフォーマンス上のボトルネックを識別するためのプローブ・プロファイラ API を提供します。	第 32 章
DBMS_RANDOM	組込み式の乱数ジェネレータを提供します。	第 33 章
DBMS_RECTIFIER_DIFF	2 つのレプリケート・サイト間のデータの不整合を検出および解決するために使用する API を提供します。	第 34 章
DBMS_REFRESH	トランザクション的に一貫性のある時点にまとめてリフレッシュできるスナップショットのグループを作成できます。分散オプションが必要です。	第 35 章
DBMS_REPAIR	データの破損修復プロシージャを提供します。	第 36 章
DBMS_REPCAT	レプリケーション・カタログと環境を管理および更新するためのルーチンを提供します。レプリケーション・オプションが必要です。	第 37 章

表 1-1 オラクル社が提供するパッケージの一覧

パッケージ名	説明	参照先
DBMS_REPCAT_ADMIN	対称型レプリケーション機能に必要な権限を持つユーザーを作成します。レプリケーション・オプションが必要です。	第 38 章
DBMS_REPCAT_INSTANTIATE	配置テンプレートをインスタンス化します。レプリケーション・オプションが必要です。	第 39 章
DBMS_REPCAT_RGT	リフレッシュ・グループ・テンプレートのメンテナンスと定義を制御します。レプリケーション・オプションが必要です。	第 40 章
DBMS_REPUTIL	表複製用のシャドウ表、トリガーおよびパッケージを生成するルーチンを提供します。	第 41 章
DBMS_RESOURCE_MANAGER	プラン、コンシューマ・グループおよびプラン・ディレクティブをメンテナンスします。また、変更内容をプラン・スキーマにグループ化する方法も提供します。	第 42 章
DBMS_RESOURCE_MANAGER_PRIVS	リソース・コンシューマ・グループに関連付けられた権限を管理します。	第 43 章
DBMS_RLS	行レベルのセキュリティ管理インタフェースを提供します。	第 44 章
DBMS_ROWID	ROWID を作成し、その内容を解釈するプロシージャを提供します。	第 45 章
DBMS_SESSION	SQL ALTER SESSION 文とその他のセッション情報に、ストアド・プロシージャからアクセスできるようにします。	第 46 章
DBMS_SHARED_POOL	標準の LRU メカニズムによって期限切れで削除されないようにオブジェクトを共有メモリーに保存します。	第 47 章
DBMS_SNAPSHOT (シノニム DBMS_MVIEW)	同じリフレッシュ・グループやパージ・ログの一部ではない複数のスナップショットをまとめてリフレッシュします。分散オプションが必要です。	第 48 章
DBMS_SPACE	標準 SQL を介しては使用できないセグメント領域情報を提供します。	第 49 章
DBMS_SPACE_ADMIN	標準 SQL を介しては使用できない表領域とセグメント領域の管理を提供します。	第 50 章
DBMS_SQL	動的 SQL を使用してデータベースへのアクセスを可能にします。	第 51 章
DBMS_STANDARD	ユーザー・アプリケーションと Oracle との対話に役立つ言語機能を提供します。	(表の最後の注 1 を参照)

表 1-1 オラクル社が提供するパッケージの一覧

パッケージ名	説明	参照先
DBMS_STATS	データベース・オブジェクト用に収集したオプティマイザの統計情報を、ユーザーが表示および変更するためのメカニズムを提供します。	第 52 章
DBMS_TRACE	PL/SQL トレースを起動および停止するルーチンを提供します。	第 53 章
DBMS_TRANSACTION	ストアド・プロシージャから SQL トランザクション文へのアクセスを提供し、トランザクション・アクティビティをモニターします。	第 54 章
DBMS_TTS	トランスポート可能なセットが自己完結型かどうかをチェックします。	第 55 章
DBMS_UTILITY	様々なユーティリティ・ルーチンを提供します。	第 56 章
DEBUG_EXTPROC	実行プロセスに連結できるデバッガを使用して、プラットフォーム上で外部プロシージャをデバッグできるようにします。	第 57 章
OUTLN_PKG	ストアド・アウトラインの管理に関連したプロシージャとファンクションに対するインタフェースを提供します。	第 58 章
PLITBLM	索引表操作を処理します。	(表の最後の注 1 を参照)
SDO_ADMIN (表の最後の注 3 を参照)	空間オブジェクトに対する空間索引の作成とメンテナンスをインプリメントするファンクションを提供します。	『Oracle8i Spatial ユーザーズ・ガイドおよびリファレンス』
SDO_GEOM (表の最後の注 3 を参照)	空間オブジェクトで形状操作をインプリメントするファンクションを提供します。	『Oracle8i Spatial ユーザーズ・ガイドおよびリファレンス』
SDO_MIGRATE (表の最後の注 3 を参照)	リリース 7.3.3 と 7.3.4 から 8.1.x に空間データを移行するためのファンクションを提供します。	『Oracle8i Spatial ユーザーズ・ガイドおよびリファレンス』
SDO_TUNE (表の最後の注 3 を参照)	Spatial カートリッジで使用する空間索引スキームの動作を決定するパラメータを選択するためのファンクションを提供します。	『Oracle8i Spatial ユーザーズ・ガイドおよびリファレンス』
STANDARD	すべての PL/SQL プログラムで自動的に使用可能となる型、例外およびサブプログラムを宣言します。	(表の最後の注 1 を参照)
TimeSeries (表の最後の注 2 を参照)	抽出、検索、計算、集合などの操作を時系列データで実行するファンクションを提供します。	『Oracle8i Time Series ユーザーズ・ガイド』
TimeScale (表の最後の注 2 を参照)	スケールアップとスケールダウンのファンクションを提供します。	『Oracle8i Time Series ユーザーズ・ガイド』

表 1-1 オラクル社が提供するパッケージの一覧

パッケージ名	説明	参照先
TSTools (表の最後の注 2 を参照)	管理ツール・プロシージャを提供します。	『Oracle8i Time Series ユーザーズ・ガイド』
UTL_COLL	PL/SQL プログラムで、コレクション・ロケータを使用した問合せと更新を可能にします。	第 59 章
UTL_FILE	PL/SQL プログラムで、オペレーティング・システム (operating system: OS) テキスト・ファイルの読み込みと書き込みを可能にし、標準 OS ストリーム・ファイル I/O の制限付きバージョンを提供します。	第 60 章
UTL_HTTP	PL/SQL と SQL から HTTP コールアウトを使用可能にして、インターネット上のデータへのアクセスまたは Oracle Web Server カートリッジのコールを可能にします。	第 61 章
UTL_INADDR	インターネット・アドレッシングをサポートするためのプロシージャを提供します。	第 62 章
UTL_PG	COBOL 数値データと Oracle の数値型との間の変換を行うファンクションを提供します。	『Oracle Procedural Gateway for APPC User's Guide』
UTL_RAW	複数の RAW の間で concat や substr などを行う RAW データ型に関する SQL ファンクションを提供します。	第 63 章
UTL_REF	オブジェクトへの参照を提供することによって、PL/SQL プログラムがオブジェクトにアクセスできるようにします。	第 64 章
UTL_SMTP	E メールを送信するための PL/SQL 機能を提供します。	第 65 章
UTL_TCP	サーバーと外部との TCP/IP ベースの通信をサポートする PL/SQL 機能を提供します。	第 66 章
Vir_Pkg (表の最後の注 2 を参照)	Visual Information Retrieval について、分析と変換のファンクションを提供します。	『Oracle8i Visual Information Retrieval User's Guide and Reference』

表 1-1 オラクル社が提供するパッケージの一覧

パッケージ名	説明	参照先
注 1: <p>DBMS_STANDARD、STANDARD および PLITBLM の各パッケージには、基本言語に関する機能のインプリメントに役立つサブプログラムが含まれています。このサブプログラムは、直接コールしないことをお勧めします。このため、これら 3 つの提供パッケージについては、このマニュアルでは説明しません。</p>		
注 2: <p>Time-Series、Image、Visual Information Retrieval、Audio および Server-Managed Video カートリッジ・パッケージは、パブリック・シノニムなしでユーザー ORDSYS にインストールされます。</p>		
注 3: <p>Spatial カートリッジ・パッケージは、パブリック・シノニム付きでユーザー MDSYS にインストールされます。</p>		

サプリメンタル・パッケージ内のサブプログラム

この章の後半にリストされているパッケージは、主に他の Oracle 関連資料で説明されています。この項では、これらの各パッケージが提供するサブプログラムを紹介します。詳細は、前述の表の参照先を参照してください。

Calendar

表 1-2 Calendar パッケージのサブプログラム

サブプログラム	説明
CombineCals	2 つのカレンダを結合します。
Day	日を基準単位としたカレンダを作成します。
DeleteExceptions	指定したカレンダから、指定した日付 (delExcDate) と一致するか、または日付の表 (delExcTab) に含まれるすべての例外を削除し、結果のカレンダを戻します。
DisplayValCal	ValidateCal ファンクションが戻した結果を表示します。
EqualCals	2 つのカレンダが等しいかどうかをチェックします。
GenDateRangeTab	入力カレンダ内 (または startDate から endDate パラメータの間) の有効期間をすべて表す日付範囲の表を戻します。
GetIntervalEnd	入力タイムスタンプを含んだ間隔の終了日を戻します。
GetIntervalStart	入力タイムスタンプを含んだ間隔の開始日を戻します。
GetOffset	2 番目の日付を 1 番目の日付からオフセットするためのタイムスタンプの数を戻します。
Hour	時間を基準単位としたカレンダを作成します。
InsertExceptions	指定したカレンダに、指定した日付 (newExcDate) と一致するか、または日付の表 (newExcTab) に含まれているすべての例外を挿入し、結果のカレンダを戻します。
IntersectCals	2 つのカレンダの交差部分を戻します。
InvalidTimeStamps Between	指定したカレンダに基づいて、その範囲内に無効なタイムスタンプを含んでいる表 (ORDTDateTab) を戻します。
IsValidCal	カレンダが有効な場合は 1、無効な場合は 0 (ゼロ) を戻します。
IsValidDate	指定したカレンダに基づいて、入力日付が有効か無効かをチェックします。
Minute	分を基準単位としたカレンダを作成します。

表 1-2 Calendar パッケージのサブプログラム

サブプログラム	説明
Month	月を基準単位としたカレンダーを作成します。
NumInvalidTimeStampsBetween	指定したカレンダーに基づいて、その範囲内での無効なタイムスタンプの数を返します。
NumOffExceptions	指定したカレンダーに基づいて、その範囲内でのオフ例外の数を返します。
NumOnExceptions	指定したカレンダーに基づいて、その範囲内でのオン例外の数を返します。
NumTimeStampsBetween	指定したカレンダーに基づいて、その範囲内での有効なタイムスタンプの数を返します。
OffsetDate	オフセット入力に対応するタイムスタンプを返します。
Quarter	四半期を基準単位としたカレンダーを作成します。
Second	秒を基準単位としたカレンダーを作成します。
Semi_annual	半年を基準単位としたカレンダーを作成します。
Semi_monthly	半月を基準単位としたカレンダーを作成します。
SetPrecision	指定したカレンダーの基準単位に適用した精度レベルを反映したタイムスタンプを返します。
Ten_day	10 日を基準単位としたカレンダーを作成します。
TimeStampsBetween	指定したカレンダーに基づいて、その範囲内での有効なタイムスタンプを含んだ表 (ORDDateTab) を返します。
UnionCals	2 つの入力カレンダーを結合したカレンダーを返します。
ValidateCal	カレンダーを検証し、必要に応じてカレンダーを修正し、問題と修正内容に関する情報を出力します。
Week	週を基準単位としたカレンダーを作成します。
Year	年を基準単位としたカレンダーを作成します。

SDO_ADMIN

表 1-3 SDO_ADMIN パッケージのサブプログラム

サブプログラム	説明
POPULATE_INDEX	空間データがある表の空間索引を作成します。
UPDATE_INDEX	指定された空間オブジェクト・インスタンスの空間索引を更新します。

SDO_GEOM

表 1-4 SDO_GEOM パッケージのサブプログラム

サブプログラム	説明
RELATE	2 つの空間オブジェクト間のトポロジ的な関係を判断します。
VALIDATE_GEOMETRY	空間オブジェクトの形状整合性制約を評価します。
WITHIN_DISTANCE	2 つの空間オブジェクトが、互いに任意のユークリッド距離内にあるかどうかを判断します。

SDO_MIGRATE

表 1-5 SDO_MIGRATE パッケージのサブプログラム

サブプログラム	説明
TO_81x	空間データを、7.3.3 または 7.3.4 のデータベースから 8.1.x データベースに移行します。

SDO_TUNE

表 1-6 SDO_TUNE パッケージのサブプログラム

サブプログラム	説明
ESTIMATE_TILING_LEVEL	空間索引の解決方法を定めるパラメータの値を提示します。

TimeScale

表 1-7 TimeScale パッケージのサブプログラム

サブプログラム	説明
Scaledown Interpolate	入力時系列上の値の間にデータ値が補間された時系列を戻します。
ScaledownRepeat	入力時系列のデータ値が繰り返される時系列を戻します。
ScaledownSplit	入力時系列のデータ値を、結果の時系列にある関連するタイムスタンプ数で除算した結果を反映する時系列を戻します。
ScaleupAvg	値の各スケール・グループの平均値を反映する時系列を戻します。
ScaleupAvgX	値の各スケール・グループに、直前のソース期間を加算した値の平均値を反映する時系列を戻します。
ScaleupCount	値の各スケール・グループにある NULL 以外のタイムスタンプの件数を反映する時系列を戻します。
ScaleupFirst	値の各スケール・グループの NULL 以外の最初の値を反映する時系列を戻します。
ScaleupGMean	値の各スケール・グループの形状平均を反映する時系列を戻します。
ScaleupLast	値の各スケール・グループの NULL 以外の最後の値を反映する時系列を戻します。
ScaleupMax	値の各スケール・グループの最大値を反映する時系列を戻します。
ScaleupMin	値の各スケール・グループの最小値を反映する時系列を戻します。
ScaleupSum	値の各スケール・グループの合計を反映する時系列を戻します。
ScaleupSumAnnual	値の各スケール・グループの合計値（年率で表現）を反映する時系列を戻します。

TimeSeries

表 1-8 TimeSeries パッケージのサブプログラム

サブプログラム	説明
Cavg	入力 ORDTNumSeries 内の対応する要素とその時点までの累積平均を含んだ各要素とともに ORDTNumSeries を戻します。
Cmax	入力 ORDTNumSeries 内の対応する要素とその時点までの累積最大値を含んだ各要素とともに ORDTNumSeries を戻します。
Cmin	入力 ORDTNumSeries 内の対応する要素とその時点までの累積最小値を含んだ各要素とともに ORDTNumSeries を戻します。
Cprod	入力 ORDTNumSeries 内の対応する要素とその時点までの乗算の累積結果を含んだ各要素とともに ORDTNumSeries を戻します。
Csum	入力 ORDTNumSeries 内の対応する要素とその時点までの累積合計値を含んだ各要素とともに ORDTNumSeries を戻します。
DeriveExceptions	時系列、カレンダーと日付表、または 2 つの時系列からカレンダー例外を導出します。
Display	DBMS_OUTPUT ルーチンを使用して、様々な情報を表示します。
DisplayValTS	ValidateTS ファンクションが戻した結果を表示します。
ExtractCal	時系列が基準単位としているカレンダーと同じカレンダーを戻します。
ExtractDate	日付を戻します。
ExtractTable	時系列に関連付けられている時系列表 (ORDTNumTab または ORDTVvarchar2Tab) を戻します。
ExtractValue	時系列内の指定された要素に格納されている値を戻します。
Fill	欠落している日付の値が挿入される時系列を戻します。
First	指定された時系列内の最初の要素を戻します。
FirstN	指定された時系列内の最初の NumValues 要素を戻します。
GetDatedElement	指定された日付に対する時系列要素を戻します。
GetNthElement	指定された時系列内、または日付範囲が指定されている場合はその範囲内にある N 番目の要素 (target_index に対応する位置にある要素) を戻します。
GetSeries	時系列インスタンス (ORDTNumSeries または ORDTVvarchar2Series) を戻します。
IsValidTS	時系列が有効な場合は 1、無効な場合は 0 (ゼロ) を戻します。

表 1-8 TimeSeries パッケージのサブプログラム

サブプログラム	説明
Lag	適切な数のタイムスタンプによって入力時系列を遅らせる、または（数値が負の場合は）進ませる時系列を戻します。
Last	指定された時系列内の最後の要素を戻します。
LastN	指定された時系列内の最後の NumValues 要素を戻します。
Lead	適切な数のタイムスタンプによって入力時系列を進ませる、または（数値が負の場合は）遅らせる時系列を戻します。
Mavg	指定した時系列、または日付範囲が指定された場合はその範囲に対する移動平均を戻します。
Msum	時系列、または日付範囲が指定された場合はその範囲に対する移動合計を戻します。
TrimSeries	指定された日付範囲外のデータをすべて削除した同じ型の ORDT 時系列を戻します。
TSAdd	2つの入力時系列または時系列と定数、およびオプションで開始日付と終了日付が指定されると、最初の2つのパラメータの加算結果を反映する時系列を戻します。
TSAvg	入力 ORDTNumSeries、およびオプションで開始日付と終了日付が指定されると、NULL 以外のすべての時系列エントリの平均に対応する数値を戻します。
TSCount	入力 ORDTNumSeries、およびオプションで開始日付と終了日付が指定されると、NULL 以外のすべての時系列エントリの件数に対応する数値を戻します。
TSDivide	2つの入力時系列または時系列と定数、およびオプションで開始日付と終了日付が指定されると、最初のパラメータを2番目のパラメータで除算した結果を反映する時系列を戻します。
TSMax	入力 ORDTNumSeries、およびオプションで開始日付と終了日付が指定されると、NULL 以外のすべての時系列エントリの最高（最大）値に対応する数値を戻します。
TSMaxN	指定した数（NumValues）の最高値を含む ORDTNumTab を戻します。
TSMedian	NULL 以外のすべての時系列エントリの中央値に対応する数値を戻します。
TSMin	NULL 以外のすべての時系列エントリの最低（最小）値に対応する数値を戻します。
TSMinN	指定した数（NumValues）の最低値を含む ORDTNumTab を戻します。

表 1-8 TimeSeries パッケージのサブプログラム

サブプログラム	説明
TSMultiply	2つの入力時系列または時系列と定数、およびオプションで開始日付と終了日付が指定されると、最初のパラメータに2番目のパラメータを乗算した結果を反映する時系列を戻します。
TSProd	NULL 以外のすべての時系列エントリの積（乗算の結果）に対応する数値を戻します。
TSStdDev	NULL 以外のすべての時系列エントリの標準偏差に対応する数値を戻します。
TSSubtract	2つの入力時系列または時系列と定数、およびオプションで開始日付と終了日付が指定されると、最初のパラメータから2番目のパラメータを減算した結果を反映する時系列を戻します。
TSSum	NULL 以外のすべての時系列エントリの合計に対応する数値を戻します。
TSVariance	NULL 以外のすべての時系列エントリの平方偏差に対応する数値を戻します。
ValidateTS	時系列が有効かどうかをチェックし、無効な場合は、診断メッセージと無効の原因となっているタイムスタンプを持つ表を出力します。

TSTools

表 1-9 TSTools パッケージのサブプログラム

サブプログラム	説明
Add_Existing_Column	既存のフラット表の列属性を時系列に追加します。
Add_Integer_Column	継続中のフラット時系列作成仕様に整数列の属性を追加します。
Add_Number_Column	継続中のフラット時系列作成仕様に整数列の属性を追加します。
Add_Varchar2_Column	継続中のフラット時系列の作成仕様に VARCHAR2 列属性を追加します。
Begin_Create_TS_Schema	時系列用のスキーマ・オブジェクトを作成するために、コンテキストを開始します。
Cancel_Create_TS_Schema	時系列スキーマの作成を取り消します。
Close_Log	Open_Log プロシージャがオープンしたログ・ファイルをクローズします。
Display_Attributes	作成している時系列スキーマに関する情報を表示します。
Drop_TS_Schema	時系列スキーマ定義とそれに関連付けられているすべてのビューを削除します。
Drop_TS_Schema_All	時系列スキーマ定義とそれに関連付けられているすべての表、ビュー、索引、制約およびトリガーを削除します。
End_Create_TS_Schema	Begin_Create_TS_Schema プロシージャで設定されたコンテキストをクローズし、適切なスキーマ・オブジェクトをすべて作成します。
Get_Flat_Attributes	フラット時系列の属性を取り出します。
Get_Object_Attributes	オブジェクト・モデル時系列の属性を取り出します。
Get_Status	時系列作成処理が進行中かどうかをチェックします。
Open_Log	Time Series 管理ツール・プロシージャで生成されたデータ定義言語 (data definition language: DDL) を含むログ・ファイルをオープンします。
Set_Flat_Attributes	フラット時系列の属性を設定します。
Set_Object_Attributes	オブジェクト・モデル時系列の属性を設定します。
Trace_Off	Time Series カートリッジ管理ツール・プロシージャに関するデバッグを無効にします。

表 1-9 TSTools パッケージのサブプログラム

サブプログラム	説明
Trace_On	Time Series カートリッジ管理ツール・プロシージャに関するデバッグを有効にします。

UTL_PG

表 1-10 UTL_PG パッケージのサブプログラム

サブプログラム	説明
MAKE_NUMBER_TO_RAW_FORMAT	宣言した精度とスケールの Oracle の数値型をリモート・ホストの内部フォーマットの RAW バイト文字列に変換するために使用する number_to_raw フォーマット変換仕様を作成します。
MAKE_RAW_TO_NUMBER_FORMAT	RAW バイト列を、リモート・ホストの内部フォーマットから対応する精度とスケールの Oracle の数値型に変換するために使用する raw_to_number フォーマット変換仕様を作成します。
NUMBER_TO_RAW	宣言した精度とスケールの Oracle の数値型をリモート・ホストの内部フォーマットの RAW バイト列に変換します。
NUMBER_TO_RAW_FORMAT	number_to_raw 変換フォーマット n2rfmt に基づいて、宣言した精度とスケールの Oracle の数値型 numval をリモート・ホストの内部フォーマットの RAW バイト列に変換します。
RAW_TO_NUMBER	リモート・ホストの内部フォーマットの RAW バイト列を Oracle の数値型に変換します。
RAW_TO_NUMBER_FORMAT	raw_to_number 変換フォーマット r2nfmt に基づいて、リモート・ホストの内部フォーマットの RAW バイト列 rawval を Oracle の数値型に変換します。
WMSG	wmsgitem で指定された警告メッセージを wmsgblk から抽出します。
WMSGCNT	wmsgblk をテストして警告の数を判断します。

Vir_Pkg

表 1-11 Vir_Pkg パッケージのサブプログラム

サブプログラム	説明
Analyze	イメージ BLOB または BFILE を分析して、可視属性に関連する情報を取り出し、イメージ署名を作成します。
Convert	イメージ署名をホスト・マシンで使用可能なフォーマットに変換するか、または Viisage face-recognition に使用するフォーマットから Score と Similar 演算子を使用できる署名に変換します。
Score	2 つのイメージ署名を比較し、可視属性の距離の加重合計を表す数値を戻します。
Similar	2 つのイメージが一致しているかどうかを判断します。

DBMS_ALERT

DBMS_ALERT パッケージは、データベース・イベント（アラート）の非同期通知をサポートします。このパッケージとデータベース・トリガーを適切に使用することによって、アプリケーションは、データベース内で関連する値が変更されるたびに通知を受け取ることができます。

たとえば、グラフィック・ツールで、データベース表のデータをグラフ表示する場合を想定します。グラフィック・ツールは、データを読み込んでグラフ表示した後、読み込んだデータに関連するデータベース・アラート（WAITONE）を待機させることができます。他のユーザーがデータを変更すると、ツールは自動的に起動されます。必要となるのは、データベース表にトリガーを設定することのみです。この設定によってトリガーが起動されると必ず信号（SIGNAL）が送信されます。

アラートは、トランザクション単位で処理されます。つまり、アラートを通知するトランザクションがコミットされるまで、待機中セッションはアラートを受け取りません。したがって、特定のアラートに対する同時実行の送信と待機がいくつか発生する場合があります。

待機中のアプリケーションはデータベース内でブロックされるため、別の処理は実行できません。

注意： データベース・アラートはコミットを発行するため、Oracle Forms では使用できません。Oracle Forms がアクティブの状態ですトアド・プロシージャをコールする場合の制限については、Oracle Forms のドキュメントを参照してください。

セキュリティ

このパッケージのセキュリティは、選択したユーザーまたはロールにこのパッケージの EXECUTE 権限を付与することで制御できます。このパッケージの先頭部分に、使用するアラート名を制限するためのカバールパッケージを記述することもできます。この場合は、パッケージではなく、このカバールパッケージの EXECUTE 権限を付与できます。

定数

```
maxwait constant integer := 86400000; -- 1000 days
```

アラートの最大待機時間です（実質的には無期限）。

エラー

DBMS_ALERT では、エラー状態のときにアプリケーション・エラー -20000 が発生します。次の表は、エラー・メッセージとそのエラーが発生する可能性のあるプロシージャの一覧です。

表 2-1 DBMS_ALERT エラー・メッセージ

エラー・メッセージ	プロシージャ
ORU-10001 lock request error, status: N	SIGNAL
ORU-10015 error: N waiting for pipe status	WAITANY
ORU-10016 error: N sending on pipe 'X'	SIGNAL
ORU-10017 error: N receiving on pipe 'X'	SIGNAL
ORU-10019 error: N on lock request	WAIT
ORU-10020 error: N on lock request	WAITANY
ORU-10021 lock request error; status: N	REGISTER
ORU-10022 lock request error, status: N	SIGNAL
ORU-10023 lock request error; status N	WAITONE
ORU-10024 there are no alerts registered	WAITANY
ORU-10025 lock request error; status N	REGISTER
ORU-10037 attempting to wait on uncommitted signal from same session	WAITONE

アラートの使用方法

アプリケーションは、複数のイベントに対して登録でき、WAITANY プロシージャを使用して、すべてのイベントの発生を待機できます。

WAITONE または WAITANY プロシージャに対して、オプションの timeout パラメータを指定することもできます。timeout を 0（ゼロ）に設定すると、保留中のアラートが存在しない場合はすぐに戻されます。

通知セッションは、待機中セッションが受け取るメッセージをオプションで渡すことができます。

アラートは、対応するアプリケーションの待機コールよりも頻繁に通知されます。この場合、古いアラートは廃棄されます。アプリケーションは、トランザクションのコミットごとに、最新のアラートを取得します。

アプリケーションで、トランザクション単位のアラートが必要ない場合は、DBMS_PIPE パッケージを使用してください。

関連項目： [第 31 章「DBMS_PIPE」](#)

SIGNAL のコール後にトランザクションがロールバックされた場合、アラートは発生しません。

アラートを受け取り、データを読み込んでも、データが変更されていない場合があります。これは、先行したアラートの後にデータが変更されたが、その前に先行したアラートに関するデータが読み込まれているためです。

アラートのチェック

通常、Oracle はイベント・ドリブンであるため、ポーリング・ループは発生しません。ポーリング・ループは、次の 2 つの場合に発生する可能性があります。

- 共有モード。データベースを共有モードで実行している場合は、別のインスタンスからのアラートをチェックするためにポーリング・ループが必要です。ポーリング・ループのデフォルト値は 1 秒で、SET_DEFAULTS プロシージャで設定できます。
- WAITANY プロシージャ。WAITANY プロシージャを使用していて、通知セッションが通知の 1 秒以内にコミットしない場合は、このコミットされていないアラートが別のアラートをカムフラージュしないようにするために、ポーリング・ループが必要です。ポーリング・ループは 1 秒間隔で始まり、30 秒間隔まで段階的に変化します。

サブプログラムの要約

表 2-2 DBMS_ALERT パッケージのサブプログラム

サブプログラム	説明
2-4 ページの REGISTER プロシージャ	アラートからメッセージを受け取ります。
2-5 ページの REMOVE プロシージャ	アラートからの通知を無効にします。
2-5 ページの REMOVEALL プロシージャ	このセッションに対するアラートを登録リストからすべて削除します。
2-6 ページの SET_DEFAULTS プロシージャ	ポーリング間隔を設定します。
2-6 ページの SIGNAL プロシージャ	アラートを通知します（登録セッションにメッセージを送信します）。
2-7 ページの WAITANY プロシージャ	セッションに登録したアラートからのメッセージの受取りを、timeout 秒待機します。
2-8 ページの WAITONE プロシージャ	名前を設定したアラートからのメッセージの受取りを、timeout 秒待機します。

REGISTER プロシージャ

セッションは、このプロシージャによってアラートを登録します。アラートの名前が IN パラメータになります。セッションは、必要な数のアラートを登録できます。アラートとの関連が不要になった場合は、REMOVE をコールしてそのアラートの登録を削除する必要があります。

構文

```
DBMS_ALERT.REGISTER (  
    name IN VARCHAR2);
```

パラメータ

表 2-3 REGISTER プロシージャのパラメータ

パラメータ	説明
name	このセッションに関連しているアラート名

注意： 'ORA\$' で始まるアラート名は、オラクル社の製品用に予約されています。アラート名は、30 バイト以内で設定してください。大文字と小文字は区別されません。

REMOVE プロシージャ

アラートとの関連が不要になったセッションは、このプロシージャによって登録リストからそのアラートを削除できます。アラートを削除すると、アラートの通知側が行う処理量が削減されます。

アラートの削除は、アラートの通知側が行う処理量を削減するための重要な処理です。セッションがアラートを削除せずに異常終了した場合、そのアラートは（即時ではありませんが）結果的に消去されます。

構文

```
DBMS_ALERT.REMOVE (  
    name IN VARCHAR2);
```

パラメータ

表 2-4 REMOVE プロシージャのパラメータ

パラメータ	説明
name	登録リストから削除するアラートの名前（大 / 小文字区別なし）

REMOVEALL プロシージャ

このプロシージャによって、このセッションに関連するアラートを登録リストからすべて削除します。セッションですべてのアラートが不要になったときは、必ずこのプロシージャを使用してください。

このプロシージャは、セッションの中でこのパッケージを初めて参照すると、自動的にコールされます。したがって、前のセッションが異常終了した場合でも、そのセッションに関連するアラートが現在のセッションに影響を与えることはありません。

このプロシージャは常にコミットを実行します。

構文

```
DBMS_ALERT.REMOVEALL;
```

パラメータ

なし

SET_DEFAULTS プロシージャ

ポーリング・ループが必要な場合は、この SET_DEFAULTS プロシージャを使用してポーリング間隔を設定します。

構文

```
DBMS_ALERT.SET_DEFAULTS (  
    polling_interval IN NUMBER);
```

パラメータ

表 2-5 SET_DEFAULTS プロシージャのパラメータ

パラメータ	説明
polling_interval	ポーリング間のスリープ時間（秒）。 デフォルトは 5 秒です。

SIGNAL プロシージャ

このプロシージャはアラートを通知します。SIGNAL コールは、コールしたトランザクションがコミットされたときのみ有効になります。トランザクションがロールバックされると、SIGNAL は無効になります。

このアラートを登録したすべてのセッションに通知されます。関連しているセッションが現在待機中の場合は、そのセッションが起動されます。関連しているセッションが現在待機中でない場合は、次にそのセッションが待機コールを行ったときに通知されます。

複数のセッションが、同時に同じアラートの通知を受けることができます。各セッションは、アラートを通知するとき、コミットするまでその他の同時セッションをすべてブロックします。この結果、トランザクションはシリアライズ化されます。

構文

```
DBMS_ALERT.SIGNAL (  
    name      IN VARCHAR2,  
    message   IN VARCHAR2);
```

パラメータ

表 2-6 SIGNAL プロシージャのパラメータ

パラメータ	説明
name	通知するアラートの名前。
message	このアラートに関連付けるメッセージ（1800 バイト以下）。 このメッセージが待機中のセッションに渡されます。待機中のセッションは、メッセージ内の情報を使用して、アラート発生後のデータベースの読み込みを中止することもできます。

WAITANY プロシージャ

現行のセッションが登録されているすべてのアラートを対象として、そのいずれかの発生を待機する場合に WAITANY をコールします。同一のセッションで、最初にアラートを通知して、その後アラートを待機する場合があります。この場合は、通知の後と待機の前に必ずコミットしてください。この処理を行わないと、DBMS_LOCK.REQUEST（DBMS_ALERT によってコールされます）からステータス 4 が戻されます。

構文

```
DBMS_ALERT.WAITANY (  
    name      OUT  VARCHAR2,  
    message   OUT  VARCHAR2,  
    status     OUT  INTEGER,  
    timeout    IN   NUMBER DEFAULT MAXWAIT);
```

パラメータ

表 2-7 WAITANY プロシージャのパラメータ

パラメータ	説明
name	発生したアラートの名前を戻します。
message	アラートに関連付けられたメッセージを戻します。 これは、SIGNAL コールが提供するメッセージです。WAITANY の前に、このアラートで複数の通知が発生した場合は、最新の SIGNAL コールに対応するメッセージが戻されます。それ以前の SIGNAL コールのメッセージは廃棄されます。
status	戻される値。 0 - アラート発生。 1 - タイムアウト発生。
timeout	アラートの最大待機時間。 timeout 秒以内にアラートが発生しない場合は、ステータス 1 が戻されます。

エラー

-20000, ORU-10024: there are no alerts registered.

原因： 待機前にアラートを登録する必要があります。

WAITONE プロシージャ

このプロシージャは、特定のアラートの発生を待機します。同一のセッションで、最初のアラートを通知して、その後のトランザクションでアラートを待機する場合があります。この場合、通知の後と待機の前に必ずコミットしてください。この処理を行わないと、DBMS_LOCK.REQUEST (DBMS_ALERT によってコールされます) からステータス 4 が戻されます。

構文

```
DBMS_ALERT.WAITONE (  
    name      IN   VARCHAR2,  
    message   OUT  VARCHAR2,  
    status     OUT  INTEGER,  
    timeout    IN   NUMBER DEFAULT MAXWAIT);
```

パラメータ

表 2-8 WAITONE プロシージャのパラメータ

パラメータ	説明
name	待機するアラートの名前。
message	アラートに関連付けられたメッセージを戻します。 これは、SIGNAL コールが提供するメッセージです。WAITONE の前に、このアラートで複数の通知が発生した場合は、最新の SIGNAL コールに対応するメッセージが戻されます。それ以前の SIGNAL コールのメッセージは廃棄されます。
status	戻される値。 0 - アラート発生。 1 - タイムアウト発生。
timeout	アラートの最大待機時間。 名前を設定したアラートが timeout 秒以内に発生しない場合は、ステータス 1 が戻されます。

例

全従業員について、部門ごとの平均給与のグラフを作成するとします。アプリケーションは、EMP の変更を常に認識しておく必要があります。アプリケーションのコードは次のようになります。

```
DBMS_ALERT.REGISTER('emp_table_alert');
  readagain:
/* ... read the emp table and graph it */
  DBMS_ALERT.WAITONE('emp_table_alert', :message, :status);
  if status = 0 then goto readagain; else
/* ... error condition */
```

EMP 表のトリガーは次のようになります。

```
CREATE TRIGGER emptrig AFTER INSERT OR UPDATE OR DELETE ON emp
BEGIN
  DBMS_ALERT.SIGNAL('emp_table_alert', 'message_text');
END;
```

アラートが不要になると、アプリケーションは次の要求を作成します。

```
DBMS_ALERT.REMOVE('emp_table_alert');
```

この要求によって、アラートの通知側の処理量が削減されます。登録したアラートが存在している間にセッションが終了（または異常終了）した場合、そのアラートは結果的に、このパッケージの次のユーザーによって消去されます。

前述の例では、アプリケーションが中間値をすべて参照するとは限りませんが、常に最新のデータを参照することが保証されます。

DBMS_APPLICATION_INFO

アプリケーション開発者は、Oracle Trace と SQL トレース機能を持つ DBMS_APPLICATION_INFO パッケージを使用して、実行しているモジュール名またはトランザクションをデータベースに記録できます。この記録は、後で行う様々なモジュールのパフォーマンスを追跡するときに使用されます。

アプリケーションを登録することによって、システム管理者とパフォーマンス・チューニング担当者は、パフォーマンスをモジュール別に追跡できます。システム管理者は、モジュール別のリソース使用率をこの情報から追跡することもできます。アプリケーションをデータベースに登録すると、その名前とアクションが V\$SESSION と V\$SQLAREA ビューに記録されます。

ユーザーがモジュールを入力するたびに、アプリケーションがモジュール名とアクション名を自動的に設定するようにしてください。モジュール名は、Oracle Forms アプリケーションのフォーム名または Oracle プリコンパイラ・アプリケーションのコード・セグメント名である場合があります。アクション名は、通常、モジュール内の現行トランザクションの名前または説明にしてください。

モジュールに基づいて独自の統計情報を収集する場合は、最初に統計を収集する別のスキーマにこのパッケージのバージョンを記述してこのパッケージにラッパーをかけてから、SYS バージョンのパッケージをコールすることができます。このようにして、DBMS_APPLICATION_INFO のパブリック・シノニムを、DBA バージョンのパッケージまで変更できます。

注意： DBMS_APPLICATION_INFO のパブリック・シノニムが作成前に削除されることはありません。これは、ユーザーがパブリック・シノニムをユーザー独自のパッケージまでリダイレクトできるようにするためです。

権限

このパッケージを使用する前に、DBMSUTL.SQL スクリプトを実行して DBMS_APPLICATION_INFO パッケージを作成する必要があります。

サブプログラムの要約

表 3-1 DBMS_APPLICATION_INFO パッケージのサブプログラム

サブプログラム	説明
3-2 ページの SET_MODULE プロシージャ	現在実行中のモジュール名を新規モジュールに設定します。
3-3 ページの SET_ACTION プロシージャ	現行モジュール内の現行アクション名を設定します。
3-4 ページの READ_MODULE プロシージャ	現行セッションのモジュールとアクションのフィールド値を読み込みます。
3-5 ページの SET_CLIENT_INFO プロシージャ	セッションのクライアント情報フィールドを設定します。
3-6 ページの READ_CLIENT_INFO プロシージャ	現行セッションの client_info フィールドの値を読み込みます。
3-6 ページの SET_SESSION_LONGOPS プロシージャ	V\$SESSION_LONGOP 表に行を設定します。

SET_MODULE プロシージャ

このプロシージャは、現行のアプリケーションまたはモジュールの名前を設定します。モジュール名には、プロシージャ名（ストアド・プロシージャを使用している場合）またはアプリケーション名を設定してください。アクション名には、実行されるアクションを説明する名前を設定してください。

構文

```
DBMS_APPLICATION_INFO.SET_MODULE (  
    module_name IN VARCHAR2,  
    action_name IN VARCHAR2);
```


パラメータ

表 3-2 SET_MODULE プロシージャのパラメータ

パラメータ	説明
module_name	現在実行中のモジュール名。現行のモジュールが終了したときは、新規モジュールがある場合はその名前で、ない場合は NULL でこのプロシージャをコールします。48 バイトを超える名前は切り捨てられます。
action_name	現行モジュール内の現行アクション名。アクションを指定しない場合は、この値を NULL に設定してください。32 バイトを超える名前は切り捨てられます。

例

```
CREATE or replace PROCEDURE add_employee(
    name VARCHAR2,
    salary NUMBER,
    manager NUMBER,
    title VARCHAR2,
    commission NUMBER,
    department NUMBER) AS
BEGIN
    DBMS_APPLICATION_INFO.SET_MODULE(
        module_name => 'add_employee',
        action_name => 'insert into emp');
    INSERT INTO emp
        (ename, empno, sal, mgr, job, hiredate, comm, deptno)
        VALUES (name, emp_seq.nextval, salary, manager, title, SYSDATE,
            commission, department);
    DBMS_APPLICATION_INFO.SET_MODULE('', '');
END;
```

SET_ACTION プロシージャ

このプロシージャは、現行モジュール内の現行アクション名を設定します。アクション名には、実行されている現行のアクションを説明する名前を設定してください。アクション名は、すべてのトランザクションの開始前に設定することをお勧めします。

構文

```
DBMS_APPLICATION_INFO.SET_ACTION (
    action_name IN VARCHAR2);
```

パラメータ

表 3-3 SET_ACTION プロシージャのパラメータ

パラメータ	説明
action_name	現行モジュール内の現行アクション名。現行アクションが終了したときは、次のアクションがある場合はその名前で、ない場合は NULL でこのプロシージャをコールします。32 バイトを超える名前は切り捨てられます。

使用上の注意

後続のトランザクションのログが正しく記録されるように、トランザクション完了後はトランザクション名を NULL に設定してください。トランザクション名を NULL に設定しないと、後続のトランザクションのログがその前のトランザクション名で記録される可能性があります。

例

次のコードは、登録プロシージャを使用するトランザクションの例です。

```
CREATE OR REPLACE PROCEDURE bal_tran (amt IN NUMBER(7,2)) AS
BEGIN

-- balance transfer transaction

    DBMS_APPLICATION_INFO.SET_ACTION(
        action_name => 'transfer from chk to sav');
    UPDATE chk SET bal = bal + :amt
        WHERE acct# = :acct;
    UPDATE sav SET bal = bal - :amt
        WHERE acct# = :acct;
    COMMIT;
    DBMS_APPLICATION_INFO.SET_ACTION('');

END;
```

READ_MODULE プロシージャ

このプロシージャは、現行セッションのモジュールとアクション・フィールドの値を読み込みます。

構文

```
DBMS_APPLICATION_INFO.READ_MODULE (
    module_name OUT VARCHAR2,
    action_name OUT VARCHAR2);
```

パラメータ

表 3-4 READ_MODULE プロシージャのパラメータ

パラメータ	説明
module_name	SET_MODULE のコールによってモジュール名に設定された最後の値
action_name	SET_ACTION または SET_MODULE のコールによってアクション名に設定された最後の値

使用上の注意

登録アプリケーションのモジュール名とアクション名は、V\$SQLAREA を問い合わせるか、または READ_MODULE プロシージャをコールして取り出すことができます。クライアント情報は、V\$SESSION ビューを問い合わせるか、または READ_CLIENT_INFO プロシージャをコールして取り出すことができます。

例

次の問合せのサンプルは、V\$SQLAREA の MODULE と ACTION 列の使用方法的例です。

```
SELECT sql_text, disk_reads, module, action
FROM v$sqlarea
WHERE module = 'add_employee';

SQL_TEXT DISK_READS MODULE ACTION
-----
INSERT INTO emp 1 add_employee insert into emp
(ename, empno, sal, mgr, job, hiredate, comm, deptno)
VALUES
(name, next.emp_seq, manager, title, SYSDATE, commission, department)

1 row selected.
```

SET_CLIENT_INFO プロシージャ

このプロシージャは、クライアント・アプリケーションに関する追加情報を提供します。

構文

```
DBMS_APPLICATION_INFO.SET_CLIENT_INFO (
    client_info IN VARCHAR2);
```

パラメータ

表 3-5 SET_CLIENT_INFO プロシージャのパラメータ

パラメータ	説明
client_info	クライアント・アプリケーションに関するあらゆる追加情報を提供します。この情報は、V\$SESSIONS ビューに格納されています。64 バイトを超える情報は切り捨てられます。

注意： CLIENT_INFO は、ユーザーによる読み込みと書き込みが可能です。保護アプリケーション属性の格納には、アプリケーション・コンテキスト機能を使用できます。

READ_CLIENT_INFO プロシージャ

このプロシージャでは、現行セッションの client_info フィールドの値を読み込みます。

構文

```
DBMS_APPLICATION_INFO.READ_CLIENT_INFO (  
    client_info OUT VARCHAR2);
```

パラメータ

表 3-6 READ_CLIENT_INFO プロシージャのパラメータ

パラメータ	説明
client_info	SET_CLIENT_INFO プロシージャに提供された最新のクライアント情報の値

SET_SESSION_LONGOPS プロシージャ

このプロシージャは、V\$SESSION_LONGOPS ビューに行を設定します。このビューは、長時間にわたって実行する操作の進行状況を示すために使用されます。パラレル実行や Server Managed Recovery などの一部の Oracle 機能は、このビューの行を使用してデータベース・バックアップなどの状態を示します。

アプリケーション固有の長時間実行タスクの進行状況に関する情報を通知するために、アプリケーションで set_session_longops プロシージャを使用することができます。この結果、V\$SESSION_LONGOPS ビューで進行状況をモニターできます。

構文

```
DBMS_APPLICATION_INFO.SET_SESSION_LONGOPS (  
  rindex      IN OUT PLS_INTEGER,  
  slno        IN OUT PLS_INTEGER,  
  op_name     IN      VARCHAR2    DEFAULT NULL,  
  target      IN      PLS_INTEGER DEFAULT 0,  
  context     IN      PLS_INTEGER DEFAULT 0,  
  sofar       IN      NUMBER       DEFAULT 0,  
  totalwork   IN      NUMBER       DEFAULT 0,  
  target_desc IN      VARCHAR2    DEFAULT 'unknown target',  
  units       IN      VARCHAR2    DEFAULT NULL)
```

```
set_session_longops_nohint constant pls_integer := -1;
```

プラグマ

```
pragma TIMESTAMP('1998-03-12:12:00:00');
```

パラメータ

表 3-7 SET_SESSION_LONGOPS プロシージャのパラメータ

パラメータ	説明
rindex	更新する v\$session_longops 行を示すトークン。新規行を使用するには、このトークンを set_session_longops_nohint に設定します。行を再使用する場合は、先行するコールの戻り値を使用します。
slno	set_session_longops へのコール全体の情報を保存します。内部使用のためのパラメータであるため、コール側で修正しないでください。
op_name	長時間実行タスクの名前を指定します。v\$session_longops の OPNAME 列に表示されます。最大長は 64 バイトです。
target	長時間実行操作中に処理されるオブジェクトを指定します。たとえば、ソートされる表の IDなどを指定します。v\$session_longops の TARGET 列に表示されます。
context	クライアントが格納する数。v\$session_longops の CONTEXT 列に表示されます。
sofar	クライアントが格納する数。v\$session_longops の SOFAR 列に表示されます。これは通常、その時点までに処理した作業量です。
totalwork	クライアントが格納する数。v\$session_longops の TOTALWORK 列に表示されます。これは通常、この長時間実行操作で行う必要がある推定合計作業量です。
target_desc	この長時間操作で操作されるオブジェクトの説明を指定します。この結果、target パラメータにキャプションが提供されます。この値は、v\$session_longops の TARGET_DESC フィールドに表示されます。最大長は 32 バイトです。
units	sofar と totalwork を表す単位を指定します。v\$session_longops の UNITS フィールドに表示されます。最大長は 32 バイトです。

例

この例では、ループ内で 10 個のオブジェクトに対してタスクを実行します。各オブジェクトの処理が完了するたびに、プロシージャの進行状況に関する V\$SESSION_LONGOPS が更新されます。

```
DECLARE
    rindex    pls_integer;
    slno       pls_integer;
    totalwork number;
    sofar      number;
    obj        pls_integer;

BEGIN
    rindex := dbms_application_info.set_session_longops_nohint;
    sofar := 0;
    totalwork := 10;

    WHILE sofar < 10 LOOP
        -- update obj based on sofar
        -- perform task on object target

        sofar := sofar + 1;
        dbms_application_info.set_session_longops(rindex, slno,
            "Operation X", obj, 0, sofar, totalwork, "table", "tables");
    END LOOP;
END;
```


DBMS_AQ パッケージは、Oracle のアドバンスト・キューイングへのインタフェースを提供します。

関連項目： DBMS_AQ の詳細は、『Oracle8i アプリケーション開発者ガイド アドバンスト・キューイング』を参照してください。

Java クラス

DBMS_AQ と DBMS_AQADM には、Java インタフェースを使用できます。Java インタフェースは、\$ORACLE_HOME/rdbms/jlib/aqapi.jar にあります。このリリースでは、Java API は RAW 型のペイロードがあるキューに対してのみ使用できます。これらのインタフェースを使用するには、ユーザーに DBMS_AQIN パッケージの EXECUTE 権限が必要です。

列挙定数

BROWSE、LOCKED、REMOVE などの列挙定数を使用するときは、それを定義しているパッケージの範囲内で、PL/SQL 定数を指定する必要があります。操作インタフェースに関連付けられているすべての型に、DBMS_AQ を付加する必要があります。たとえば、次のようになります。

DBMS_AQ.BROWSE

表 4-1 列挙定数

パラメータ	オプション
visibility	IMMEDIATE, ON_COMMIT
dequeue mode	BROWSE, LOCKED, REMOVE, REMOVE_NODATA
navigation	FIRST_MESSAGE, NEXT_MESSAGE, NEXT_TRANSACTION
state	WAITING, READY, PROCESSED, EXPIRED
sequence_deviation	BEFORE, TOP
wait	FOREVER, NO_WAIT
delay	NO_DELAY
expiration	NEVER

データ構造

DBMS_AQ と DBMS_AQADM では、次のデータ構造が使用されています。

- [オブジェクト名](#)
- [タイプ名](#)
- [エージェント](#)
- [エンキュー・オプション・タイプ](#)
- [デキュー・オプション・タイプ](#)
- [メッセージ・プロパティ・タイプ](#)
- [AQ 受信者リスト・タイプ](#)

- AQ エージェント・リスト・タイプ
- AQ サブスクライバ・リスト・タイプ

オブジェクト名

データベース・オブジェクトに名前を設定します。この命名規則は、キュー、キュー表、エージェント名およびオブジェクト型に適用されます。

構文

```
object_name := VARCHAR2;  
object_name := [<schema_name>.<name>];
```

使用方法 オブジェクト名は、オプションのスキーマ名と名前で指定します。スキーマ名が指定されない場合は、現在のスキーマ名が使用されます。名前は、予約語に関して、『Oracle8i SQL リファレンス』のオブジェクト名ガイドラインに従う必要があります。スキーマ名、エージェント名およびオブジェクト型名の最大長は、それぞれ 30 バイトです。ただし、キュー名とキュー表名の最大長は、24 バイトです。

タイプ名

キュー・タイプを定義します。

構文

```
type_name := VARCHAR2;  
type_name := <object_type> | "RAW";
```

使用方法

表 4-2 タイプ名

パラメータ	説明
<object_types>	オブジェクト型内の属性の最大数は 900 です。
関連項目：『Oracle8i 概要』	

表 4-2 タイプ名

パラメータ	説明
"RAW"	<p>RAW 型のペイロードを格納するために、AQ は、ペイロード・リポジトリとして LOB 列を含むキュー表を作成します。メッセージ・ペイロードの理論上の最大サイズは、LOB 列に格納可能な最大データ量です。ただし、ペイロードの最大サイズは、AQ にアクセスするときに使用するプログラム環境によって決まります。PL/SQL、Java およびプリコンパイラの場合は 32K、OCI の場合は 4G です。PL/SQL のエンキューとデキューのインタフェースは、RAW バッファをペイロード・パラメータとして受け入れるため、32KB に制限されます。OCI では、ユーザーの RAW データの最大サイズは、OCI オブジェクト・キャッシュによる割当てが可能な最大連続メモリー量（OCIRaw が単純にバイト配列の場合）に制限されます。通常は、最低 32KB ですが、ほとんどの場合それ以上になります。</p> <p>LOB 列は RAW ペイロードの格納に使用されるため、AQ 管理者は、LOB 表領域を選択して、キュー表作成時に storage_clause パラメータに LOB 記憶域文字列を設定することによって、LOB 記憶域を構成できます。</p>

エージェント

メッセージのプロデューサまたはコンシューマを識別します。

構文

```
TYPE sys.aq$_agent IS OBJECT (  
    name      VARCHAR2(30),  
    address   VARCHAR2(1024),  
    protocol  NUMBER);
```

使用方法

表 4-3 エージェント

パラメータ	説明
name	メッセージのプロデューサまたはコンシューマの名前。名前は、予約語に関して、『Oracle8i SQL リファレンス』のオブジェクト名ガイドラインに従う必要があります。
address	受信者のプロトコル固有のアドレス。プロトコルが 0（デフォルト）の場合、アドレスは [schema.]queue[@dblink] の形式になります。
protocol	アドレスを解釈し、メッセージを伝播するためのプロトコル。デフォルトは 0（ゼロ）です。

エンキュー・オプション・タイプ

エンキュー操作に使用できるオプションを指定します。

構文

```
TYPE enqueue_options_t IS RECORD (  
    visibility          BINARY_INTEGER DEFAULT ON_COMMIT,  
    relative_msgid      RAW(16)          DEFAULT NULL,  
    sequence_deviation  BINARY_INTEGER DEFAULT NULL);
```

使用方法

表 4-4 エンキュー・オプション・タイプ

パラメータ	説明
visibility	エンキュー要求のトランザクション動作を指定します。 ON_COMMIT: エンキューは、現行トランザクションの一部です。操作は、トランザクションがコミットされると完了します。これがデフォルトです。 IMMEDIATE: エンキューは、現行トランザクションの一部ではありません。操作は独自のトランザクションを構成します。非永続キューにエンキューするときは、この値のみ許可されます。
relative_msgid	順序逸脱操作で参照されるメッセージのメッセージ ID を指定します。このフィールドは、sequence_deviation に BEFORE が指定されている場合のみ有効です。順序逸脱が指定されていない場合、このパラメータは無視されます。
sequence_deviation	エンキューされるメッセージが、すでにキューにある他のメッセージより前にデキューされる必要があるかどうかを指定します。 BEFORE: メッセージは、relative_msgid で指定されたメッセージより前にエンキューされます。 TOP: メッセージは、他のどのメッセージよりも前にエンキューされます。 NULL: デフォルトです。

デキュー・オプション・タイプ

デキュー操作に使用できるオプションを指定します。

構文

```
TYPE dequeue_options_t IS RECORD (  
    consumer_name    VARCHAR2(30)    DEFAULT NULL,  
    dequeue_mode     BINARY_INTEGER DEFAULT REMOVE,  
    navigation       BINARY_INTEGER DEFAULT NEXT_MESSAGE,  
    visibility       BINARY_INTEGER DEFAULT ON_COMMIT,  
    wait             BINARY_INTEGER DEFAULT FOREVER,  
    msgid            RAW(16)         DEFAULT NULL,  
    correlation       VARCHAR2(128)  DEFAULT NULL);
```

使用方法

表 4-5 デキュー・オプション・タイプ

パラメータ	説明
consumer_name	コンシューマの名前。コンシューマ名が一致するメッセージのみアクセスされます。キューが複数コンシューマ用に設定されていない場合、このフィールドは NULL に設定してください。
dequeue_mode	<p>デキューに関連付けるロック動作を指定します。</p> <p>BROWSE: メッセージのロックを取得せずにメッセージを読み込みます。SELECT 文と同じです。</p> <p>LOCKED: メッセージを読み込み、その書込みロックを取得します。ロックはトランザクションの継続中有効です。UPDATE 文に対する SELECT と同じです。</p> <p>REMOVE: メッセージを読み込み、そのメッセージを更新または削除します。これがデフォルトです。メッセージは保存プロパティに基づいてキュー表に保存されます。</p> <p>REMOVE_NODATA: メッセージに更新または削除のマークを設定します。メッセージは保存プロパティに基づいてキュー表に保存されます。</p>

表 4-5 デキュー・オプション・タイプ

パラメータ	説明
navigation	<p>取り出されるメッセージの位置を指定します。最初に、位置が判断されます。次に、検索基準が適用されます。最後に、メッセージが取り出されます。</p> <p>NEXT_MESSAGE: 使用可能で検索基準と一致する次のメッセージを取り出します。前のメッセージがメッセージ・グループに属している場合は、検索基準と一致し、そのメッセージ・グループに属している次の使用可能メッセージが取り出されます。これがデフォルトです。</p> <p>NEXT_TRANSACTION: 現行トランザクション・グループの残り（ある場合）をスキップし、次のトランザクション・グループの最初のメッセージを取り出します。このオプションは、現行キューに対してメッセージ・グループが使用可能な場合のみ使用できます。</p> <p>FIRST_MESSAGE: 使用可能で検索基準と一致する最初のメッセージを取り出します。これによって、位置がキューの先頭に再設定されます。</p>
visibility	<p>新規メッセージを現行トランザクションの一部としてデキューするかどうかを指定します。BROWSE モードの使用時は無視されます。</p> <p>ON_COMMIT: デキューは現行トランザクションの一部になります。これがデフォルトです。</p> <p>IMMEDIATE: デキュー・メッセージは、現行トランザクションの一部ではありません。独自のトランザクションを構成します。</p>
wait	<p>検索基準と一致する使用可能なメッセージが存在していない場合の待機時間を指定します。</p> <p>FOREVER: 使用可能なメッセージが発生するまで待機します。これがデフォルトです。</p> <p>NO_WAIT: 待機しません。</p> <p>数値: 待機時間（秒）。</p>
msgid	<p>デキューするメッセージのメッセージ ID を指定します。</p>
correlation	<p>デキューするメッセージの相関 ID を指定します。パーセント符号（%）やアンダースコア（_）などの特殊なパターン一致文字を使用できます。パターンを満たすメッセージが複数ある場合、デキュー順序は未定義です。</p>

メッセージ・プロパティ・タイプ

個々のメッセージを管理するために AQ が使用する情報を記述します。これらはエンキュー時に設定され、デキュー時にその値が戻されます。

構文

```
TYPE message_properties_t IS RECORD (  
    priority          BINARY_INTEGER DEFAULT 1,  
    delay             BINARY_INTEGER DEFAULT NO_DELAY,  
    expiration        BINARY_INTEGER DEFAULT NEVER,  
    correlation       VARCHAR2(128)  DEFAULT NULL,  
    attempts          BINARY_INTEGER,  
    recipient_list    aq$_recipient_list_t,  
    exception_queue   VARCHAR2(51)   DEFAULT NULL,  
    enqueue_time      DATE,  
    state             BINARY_INTEGER,  
    sender_id         aq$_agent      DEFAULT NULL,  
    original_msgid    RAW(16)        DEFAULT NULL);  
  
TYPE aq$_recipient_list_t IS TABLE OF sys.aq$_agent  
    INDEX BY BINARY_INTEGER;
```

使用方法

表 4-6 メッセージ・プロパティ・タイプ

パラメータ	説明
priority	メッセージの優先順位を指定または戻します。数値が小さいほど優先順位は高くなります。優先順位には、負数も含めてあらゆる数値を使用できます。
delay	エンキュー・メッセージの遅延を指定または戻します。遅延は、メッセージがデキュー可能になるまでの秒数を表します。msgid によるデキューの場合、delay 操作は上書きされます。遅延設定をしてエンキューされたメッセージは WAITING 状態になり、遅延期間が終了すると READY 状態になります。DELAY 処理では、キュー・モニターを起動する必要があります。delay は、メッセージをエンキューするプロデューサが設定する点に注意してください。 NO_DELAY: メッセージは、すぐにデキュー可能になります。 数値: メッセージの遅延秒数。

表 4-6 メッセージ・プロパティ・タイプ

パラメータ	説明
expiration	<p>メッセージの時間切れまでの時間を指定または戻します。メッセージのデキュー操作可能期間を秒数で指定します。このパラメータは、delay からオフセットされます。時間切れ処理では、キュー・モニターが実行されている必要があります。</p> <p>NEVER: メッセージは時間切れになりません。</p> <p>数値: メッセージを READY 状態にしておく時間 (秒数)。時間切れまでにメッセージがデキューされない場合、メッセージは EXPIRED 状態で例外キューに移されます。</p>
correlation	<p>エンキュー時にメッセージのプロデューサが提供した ID を戻します。</p>
attempts	<p>このメッセージのデキューの試行回数を戻します。このパラメータはエンキュー時には設定できません。</p>
recipient_list	<p>型定義の詳細は、4-4 ページの「エージェント」を参照してください。</p> <p>このパラメータは、複数コンシューマを許可しているキューに対してのみ有効です。デフォルトの受信者は、キューのサブスクライバです。このパラメータは、デキュー時にコンシューマに戻されません。</p>
exception_queue	<p>メッセージが正常に処理されなかった場合のメッセージの移動先キュー名を指定または戻します。メッセージが移されるのは、デキューの試行失敗回数が <i>max_retries</i> を超えた場合、またはメッセージが時間切れになった場合です。例外キューのメッセージは、すべて EXPIRED 状態になります。</p> <p>デフォルトは、キュー表に関連付けられている例外キューです。指定した例外キューが移動時に存在しない場合、そのメッセージは、キュー表に関連付けられているデフォルトの例外キューに移され、アラート・ファイルに警告ログが記録されます。デフォルトの例外キューが使用されると、デキュー時に NULL 値が戻されます。</p>
enqueue_time	<p>メッセージがエンキューされた時刻を戻します。この値はシステムによって設定され、ユーザーは設定できません。このパラメータはデキュー時には設定できません。</p>
state	<p>デキュー時にメッセージの状態を戻します。このパラメータはエンキュー時には設定できません。</p> <p>0: メッセージは処理可能な状態です。</p> <p>1: メッセージ遅延の指定秒数に達していません。</p> <p>2: メッセージは処理され、保存されています。</p> <p>3: メッセージは例外キューに移されました。</p>

表 4-6 メッセージ・プロパティ・タイプ

パラメータ	説明
sender_id	アプリケーション指定の送信者 ID を指定または戻します。 DEFAULT: NULL
original_msgid	このパラメータは、Oracle AQ がメッセージを伝播するために使用します。 DEFAULT: NULL

AQ 受信者リスト・タイプ

メッセージを受信するエージェントのリストを識別します。この構造体は、キューが複数デキュー可能な場合のみ使用されます。

構文

```
TYPE aq$_recipient_list_t IS TABLE OF sys.aq$_agent
INDEX BY BINARY_INTEGER;
```

AQ エージェント・リスト・タイプ

DBMS_AQ.LISTEN がリスニングするエージェントのリストを識別します。

構文

```
TYPE aq$_agent_list_t IS TABLE of sys.aq$_agent
INDEX BY BINARY_INTEGER;
```

AQ サブスクライバ・リスト・タイプ

このキューにサブスクライブするサブスクライバのリストを識別します。

構文

```
TYPE aq$_subscriber_list_t IS TABLE OF sys.aq$_agent
INDEX BY BINARY_INTEGER;
```

サブプログラムの要約

表 4-7 DBMS_AQ パッケージのサブプログラム

サブプログラム	説明
4-11 ページの ENQUEUE プロシージャ	指定したキューにメッセージを追加します。
4-13 ページの DEQUEUE プロシージャ	指定したキューからメッセージをデキューします。
4-15 ページの LISTEN プロシージャ	エージェントのリストにかわって1つ以上のキューをリスニングします。

注意： DBMS_AQ パッケージには、純粹さレベルが定義されていません。したがって、RNDS、WNDS、RNPS または WNPS 制約が定義されている他のプロシージャからこのパッケージ内のプロシージャをコールすることはできません。

ENQUEUE プロシージャ

このプロシージャは、指定したキューにメッセージを追加します。

構文

```
DBMS_AQ.ENQUEUE (  
  queue_name          IN      VARCHAR2,  
  enqueue_options     IN      enqueue_options_t,  
  message_properties  IN      message_properties_t,  
  payload             IN      "<type_name>",  
  msgid              OUT     RAW);
```

パラメータ

表 4-8 ENQUEUE プロシージャのパラメータ

パラメータ	説明
queue_name	このメッセージをエンキューするキュー名を指定します。例外キューは指定できません。
enqueue_options	4-5 ページの「エンキュー・オプション・タイプ」を参照してください。
message_properties	4-8 ページの「メッセージ・プロパティ・タイプ」を参照してください。
payload	Oracle AQ では解釈されません。 ペイロードは、関連するキュー表の仕様に基づいて指定する必要があります。NULL は受け入れ可能なパラメータの 1 つです。 <type_name> の定義の詳細は、4-3 ページの「タイプ名」を参照してください。
msgid	システムが生成するメッセージ ID。 これは、デキュー時にメッセージを識別するために使用するグローバルな一意の ID です。

使用上の注意

順序逸脱の使用方法 enqueue_options の sequence_deviation パラメータを使用すると、2 つのメッセージ間の処理順序を変更できます。参照されるメッセージの ID は、enqueue_options のパラメータ relative_msgid で指定できます。関係は、sequence_deviation パラメータによって識別されます。

メッセージに sequence_deviation を指定すると、このメッセージに指定できる遅延と優先の順位値が一部制限されます。遅延の値は、このメッセージより後にエンキューされるメッセージの遅延以下に設定する必要があります。優先順位は、このメッセージより後にエンキューされるメッセージの優先順位以上に設定する必要があります。

受信者がいない場合のメッセージの送信 メッセージが受信者のいない複数コンシューマ・キューにエンキューされ、かつそのキューにサブスクライバが存在しない（またはこのメッセージと一致するルールベースのサブスクライバが存在しない）場合は、Oracle エラー ORA 24033 が発生します。これは、配信可能な受信者またはサブスクライバが存在しないために、そのメッセージが廃棄されることを示す警告です。

DEQUEUE プロシージャ

このプロシージャは、指定したキューからメッセージをデキューします。

構文

```
DBMS_AQ.DEQUEUE (
    queue_name          IN      VARCHAR2,
    dequeue_options     IN      dequeue_options_t,
    message_properties   OUT     message_properties_t,
    payload             OUT     "<type_name>",
    msgid               OUT     RAW);
```

パラメータ

表 4-9 DEQUEUE プロシージャのパラメータ

パラメータ	説明
queue_name	キュー名を指定します。
dequeue_options	4-6 ページの「 デキュー・オプション・タイプ 」を参照してください。
message_properties	4-8 ページの「 メッセージ・プロパティ・タイプ 」を参照してください。
payload	Oracle AQ では解釈されません。ペイロードは、関連するキュー表の仕様に基づいて指定する必要があります。 <type_name> の定義の詳細は、4-3 ページの「 タイプ名 」を参照してください。
msgid	システムが生成するメッセージ ID。

使用上の注意

メッセージの検索基準とデキュー順序 デキューされるメッセージの検索基準は、dequeue_options の consumer_name、msgid および correlation パラメータによって判断されます。msgid はデキューされるメッセージを一意に識別します。相関識別子は、AQ によって解釈されないアプリケーション定義の識別子です。

msgid が指定されていない限り、READY 状態のメッセージのみがデキューされます。

デキュー順序は、dequeue_options の msgid と相関 ID で上書きされない限り、キュー表の作成時に指定した値によって判断されます。

キュー操作には、データベース一貫読込みメカニズムが適用されます。たとえば、BROWSE コールは、ブラウズ・トランザクションの開始後にエンキューされたメッセージを参照しない場合があります。

キューの移動 デキュー時のデフォルトの NAVIGATION パラメータは、NEXT_MESSAGE です。この場合、後続のデキューは、最初のデキューで取得したスナップショットに基づいて、キューからメッセージを取り出します。特に、最初のデキュー・コマンド後にエンキューされたメッセージは、キュー内の残りのメッセージがすべて処理されるまで処理されません。これは、すべてのメッセージがすでにキューにエンキューされている場合、またはキューに優先順位が設定されていない場合は、通常問題ありません。ただし、デキュー・コマンドのたびにキュー内の先頭メッセージを処理する必要があるときには、アプリケーションで FIRST_MESSAGE ナビゲーション・オプションを使用する必要があります。この処理は、すでにエンキューされたメッセージの処理中に優先順位の高いメッセージがキューに登録された場合に必要になります。

注意： 同時にエンキューされている複数のメッセージがある場合は、FIRST_MESSAGE ナビゲーション・オプションを使用すると効率が向上します。FIRST_MESSAGE オプションが指定されていない場合、AQ は最初のデキュー・コマンド時のスナップショットを生成し続けるためパフォーマンスが低下します。FIRST_MESSAGE オプションが指定されている場合、AQ はすべてのデキュー・コマンドに対して新しいスナップショットを使用します。

メッセージのグループ化によるデキュー 同一トランザクションでメッセージのグループ化に対応しているキューにエンキューされたメッセージは、グループを形成します。そのトランザクションでエンキューされたメッセージが1つのみの場合は、実質的に1つのメッセージでグループを形成します。1つのトランザクションでグループ化できるメッセージの数に上限はありません。

メッセージのグループ化に対応していないキューでは、LOCKED または REMOVE モードのデキューは、1つのメッセージのみロックします。これに対して、グループの一部のメッセージをデキューしようとするデキュー操作は、グループ全体をロックします。これは、グループ内のすべてのメッセージを基本単位で処理する必要がある場合に有効です。

グループ内のすべてのメッセージがデキューされている場合、そのデキューはグループ内のすべてのメッセージが処理済であることを示すエラーを戻します。この場合、アプリケーションは NEXT_TRANSACTION を使用して、次に使用可能なグループからメッセージのデキューを開始します。使用可能なグループがない場合、デキューは指定した WAIT 期間後にタイムアウトします。

LISTEN プロシージャ

このプロシージャは、エージェントのリストのかわりに 1 つ以上のキューをリスニングします。エージェントの 'address' フィールドは、エージェントがモニターするキューを示します。ローカル・キューのみアドレスとしてサポートされています。将来使用するためにプロトコルが予約されています。

エージェント・アドレスが複数コンシューマ・キューの場合、エージェント名は必須です。単一コンシューマ・キューの場合は、エージェント名を指定する必要はありません。

これは、リスト内のエージェントに対してコンシューマ用のメッセージが準備されているときに戻されるブロック・コールです。待機時間が期限切れしてもメッセージが見つからない場合は、エラーが発生します。

構文

```
DBMS_AQ.LISTEN (  
    agent_list IN      aq$_agent_list_t,  
    wait       IN      BINARY_INTEGER DEFAULT DBMS_AQ.FOREVER,  
    agent      OUT     sys.aq$_agent);  
  
TYPE aq$_agent_list_t IS TABLE of aq$_agent INDEXED BY BINARY_INTEGER;
```

パラメータ

表 4-10 LISTEN プロシージャのパラメータ

パラメータ	説明
agent_list	リスニングするエージェントのリスト。
wait	リスニング・コールのタイムアウト（秒数）。デフォルトでは、コールは永続的にブロックします。
agent	コンシューマ用のメッセージがあるエージェント。

使用上の注意

このプロシージャは、引数としてエージェントのリストを使用します。リストされた各エージェントのアドレス・フィールドに、モニターするキューを指定します。複数コンシューマ・キューをモニターするときは、エージェントの名前も指定する必要があります。単一コンシューマ・キューの場合は、エージェント名を指定する必要はありません。ローカル・キューのみアドレスとしてサポートされています。将来使用するためにプロトコルが予約されています。

これは、リスト内のエージェントに対してコンシューマ用のメッセージが準備されているときに戻されるブロック・コールです。複数のエージェントに対するメッセージがある場合は、リストの最初のエージェントのみ戻されます。待機時間が期限切れしてもメッセージが見つからない場合は、エラーが発生します。

リスニング・コールからの正常な戻りは、指定したキューの中に、リストされたエージェントの 1 つに対するメッセージがあることを示しているにすぎません。対象となっているエージェントは、関連メッセージを継続してデキューする必要があります。

非永続キューでは、リスニングはコールできない点に注意してください。

DBMS_AQADM

DBMS_AQADM パッケージは、アドバンスト・キューイングの構成と管理情報を管理するプロシージャを提供します。

関連項目： DBMS_AQADM の詳細は、『Oracle8i アプリケーション開発者ガイド アドバンスト・キューイング』を参照してください。

列挙定数

INFINITE、TRANSACTIONAL、NORMAL_QUEUE などの列挙定数を使用するときは、それを定義するパッケージの範囲内で、その記号を指定する必要があります。管理インタフェースに関連付けられているすべての型に、DBMS_AQADM を付加する必要があります。たとえば、次のようになります。

```
DBMS_AQADM.NORMAL_QUEUE
```

表 5-1 管理インタフェース内の列挙型

パラメータ	オプション
retention	0,1,2...INFINITE
message_grouping	TRANSACTIONAL, NONE
queue_type	NORMAL_QUEUE, EXCEPTION_QUEUE, NON_PERSISTENT_QUEUE

関連項目： DBMS_AQ と DBMS_AQADM で使用される Java クラスとデータ構造の詳細は、第 4 章「DBMS_AQ」を参照してください。

サブプログラムの要約

表 5-2 DBMS_AQADM パッケージのサブプログラム

サブプログラム	説明
5-4 ページの CREATE_QUEUE_TABLE プロシージャ	事前定義型のメッセージのキュー表を作成します。
5-7 ページの ALTER_QUEUE_TABLE プロシージャ	既存のキュー表を変更します。
5-8 ページの DROP_QUEUE_TABLE プロシージャ	既存のキュー表を削除します。
5-9 ページの CREATE_QUEUE プロシージャ	指定したキュー表にキューを作成します。
5-11 ページの CREATE_NP_QUEUE プロシージャ	非永続の RAW キューを作成します。
5-12 ページの ALTER_QUEUE プロシージャ	キューの既存のプロパティを変更します。
5-13 ページの DROP_QUEUE プロシージャ	既存のキューを削除します。

表 5-2 DBMS_AQADM パッケージのサブプログラム

サブプログラム	説明
5-14 ページの START_QUEUE プロシージャ	指定したキューに対するエンキューまたはデキュー（あるいはその両方）を使用可能にします。
5-15 ページの STOP_QUEUE プロシージャ	指定したキューに対するエンキューまたはデキュー（あるいはその両方）を使用禁止にします。
5-16 ページの GRANT_SYSTEM_PRIVILEGE プロシージャ	ユーザーとロールに AQ システム権限を付与します。
5-17 ページの REVOKE_SYSTEM_PRIVILEGE プロシージャ	ユーザーとロールから AQ システム権限を取り消します。
5-17 ページの GRANT_QUEUE_PRIVILEGE プロシージャ	ユーザーとロールにキューの権限を付与します。
5-18 ページの REVOKE_QUEUE_PRIVILEGE プロシージャ	ユーザーとロールからキューの権限を取り消します。
5-19 ページの ADD_SUBSCRIBER プロシージャ	キューにデフォルトのサブスクライバを追加します。
5-20 ページの ALTER_SUBSCRIBER プロシージャ	指定したキューに対するサブスクライバの既存プロパティを変更します。
5-20 ページの REMOVE_SUBSCRIBER プロシージャ	キューからデフォルトのサブスクライバを削除します。
5-21 ページの SCHEDULE_PROPAGATION プロシージャ	キューから特定の DB リンクで指定された宛先へのメッセージ伝播をスケジュールします。
5-23 ページの UNSCHEDULE_PROPAGATION プロシージャ	キューから、特定の DB リンクで指定された宛先へのメッセージ伝播の旧スケジュールを取り消します。
5-23 ページの VERIFY_QUEUE_TYPES プロシージャ	ソース・キューと宛先キューの型が同じであるかどうかを検証します。
5-24 ページの ALTER_PROPAGATION_SCHEDULE プロシージャ	伝播スケジュールのパラメータを変更します。
5-26 ページの ENABLE_PROPAGATION_SCHEDULE プロシージャ	以前使用禁止にした伝播スケジュールを使用可能にします。
5-26 ページの DISABLE_PROPAGATION_SCHEDULE プロシージャ	伝播スケジュールを使用禁止にします。

表 5-2 DBMS_AQADM パッケージのサブプログラム

サブプログラム	説明
5-27 ページの MIGRATE_QUEUE_TABLE プロシージャ	8.0 互換キュー表から 8.1 互換キュー表へのアップグレード、または 8.1 互換キュー表から 8.0 互換キュー表へのダウングレードを実行します。

CREATE_QUEUE_TABLE プロシージャ

このプロシージャは、事前定義型のメッセージ用のキュー表を作成します。デキュー順序のソート・キーがある場合は、表作成時に定義する必要があります。このときに次のオブジェクトが作成されます。

- キュー表に関連付けられているデフォルトの例外キュー。aq\$_<queue_table_name>_e という名前になります。
- AQ アプリケーションがキュー・データの間合せに使用する読取り専用ビュー。aq\$_<queue_table_name> という名前になります。
- 索引、またはキュー・モニター操作の複数コンシューマ・キューの場合の索引構成表 (IOT)。aq\$_<queue_table_name>_t という名前になります。
- 索引、またはデキュー操作の複数コンシューマ・キューの場合の索引構成表。aq\$_<queue_table_name>_i という名前になります。

Oracle8i 互換のキュー表の場合は、次の索引構成表が作成されます。

- aq\$_<queue_table_name>_s という名前の表。この表には、サブスライバに関する情報が格納されます。
- aq\$_<queue_table_name>_r という名前の表。この表には、サブスライバのルールに関する情報が格納されます。
- aq\$_<queue_table_name>_h という名前の索引構成表。この表には、デキュー履歴データが格納されます。

構文

```
DBMS_AQADM.CREATE_QUEUE_TABLE (
    queue_table           IN      VARCHAR2,
    queue_payload_type    IN      VARCHAR2,
    storage_clause        IN      VARCHAR2      DEFAULT NULL,
    sort_list             IN      VARCHAR2      DEFAULT NULL,
    multiple_consumers    IN      BOOLEAN       DEFAULT FALSE,
    message_grouping      IN      BINARY_INTEGER DEFAULT NONE,
    comment               IN      VARCHAR2      DEFAULT NULL,
    auto_commit           IN      BOOLEAN       DEFAULT TRUE,
    primary_instance      IN      BINARY_INTEGER DEFAULT 0,
    secondary_instance    IN      BINARY_INTEGER DEFAULT 0,
    compatible            IN      VARCHAR2      DEFAULT NULL);
```

パラメータ

表 5-3 CREATE_QUEUE_TABLE プロシージャのパラメータ

パラメータ	説明
queue_table	作成するキュー表の名前。
queue_payload_type	格納されるユーザー・データの型。このパラメータの有効値については、4-3 ページの「 タイプ名 」を参照してください。
storage_clause	<p>記憶域パラメータ。</p> <p>記憶域パラメータは、キュー表の作成時に、CREATE TABLE 文に組み込まれます。記憶域パラメータは、次のパラメータの任意の組合せで作成できます。PCTFREE、PCTUSED、INITRANS、MAXTRANS、TABLESPACE、LOB およびテーブルのストレージ句。</p> <p>ここで表領域が指定されない場合は、キュー表とそのすべての関連オブジェクトが、デフォルトのユーザー表領域に作成されます。ここで表領域が指定されると、キュー表とそのすべての関連オブジェクトは、テーブルのストレージ句で指定された表領域に作成されます。</p> <p>これらのパラメータの使用方法については、『Oracle8i SQL リファレンス』を参照してください。</p>
sort_list	<p>昇順ソート・キーに使用する列。</p> <p>Sort_list の書式は次のとおりです。</p> <pre>'<sort_column_1>,<sort_column_2>'</pre> <p>許可される列名は、priority と enq_time です。両方の列を指定する場合は、<sort_column_1> で最も重要な順序を定義します。</p> <p>特定の順序付けメカニズムでキュー表が作成されると、キュー表内のすべてのキューが同じデフォルトを使用します。キュー表の順序は、キュー表の作成後には変更できません。</p> <p>ソート・リストが指定されていない場合、このキュー表内のキューはすべてエンキュー時に昇順ソートされます。この順序は FIFO 順と同じです。</p> <p>デフォルトの順序が定義されている場合でも、デキュー側は、msgid または correlation を指定して、デキューするメッセージを選択できます。msgid、correlation および sequence_deviation が指定されている場合は、デフォルトのデキュー順序よりも優先されます。</p>
multiple_consumers	<p>FALSE: 表内で作成されたキューには、1 つのメッセージに対して 1 つのコンシューマしか設定できません。これがデフォルトです。</p> <p>TRUE: 表内で作成されたキューには、1 つのメッセージに対して複数のコンシューマを設定できます。</p>

表 5-3 CREATE_QUEUE_TABLE プロシージャのパラメータ

パラメータ	説明
message_grouping	表内で作成されたキューのメッセージ・グループ化に関する動作。 NONE: 各メッセージは個々に処理されます。 TRANSACTIONAL: あるトランザクションの一部としてエンキューされた複数のメッセージは、同じグループの一部とみなされ、関連するメッセージのグループとしてデキューできます。
comment	キュー表に関するユーザー指定の説明。このユーザー・コメントは、キュー・カタログに追加されます。
auto_commit	TRUE: 現行トランザクションがある場合は、CREATE_QUEUE_TABLE 操作が実行される前にコミットされます。CREATE_QUEUE_TABLE 操作は、コールから戻ると持続されます。これがデフォルトです。 FALSE: 操作は現行トランザクションの一部で、コール側がコミットを入力したときのみ持続されます。 注意: このパラメータは使用しないことをお勧めします。
primary_instance	キュー表のプライマリ所有者。キュー表内のキューに対するキュー・モニターのスケジューリングと伝播は、このインスタンス内で行われます。 プライマリ・インスタンスのデフォルト値は 0 (ゼロ) で、キュー・モニターのスケジューリングと伝播は、使用可能なすべてのインスタンス内で行われます。
secondary_instance	プライマリ・インスタンスが使用不可の場合、キュー表はセカンダリ・インスタンスにフェイルオーバーします。デフォルト値は 0 (ゼロ) で、キュー表は使用可能なすべてのインスタンスにフェイルオーバーします。
compatible	キューの互換性があるデータベースの最下位バージョン。現在有効な値は、'8.0' または '8.1' です。デフォルトは '8.0' です。

使用上の注意

CLOB、BLOB および BFILE は、AQ オブジェクト型ペイロードに対する有効な属性です。ただし、Oracle8i リリース 8.1.6 で AQ を使用して伝播できるのは、CLOB と BLOB のみです。詳細は、『Oracle8i アプリケーション開発者ガイド アドバンスト・キューイング』を参照してください。

primary_instance と secondary_instance は、8.1 互換モードでのみ指定および変更できます。

プライマリ・インスタンスが存在しないと、セカンダリ・インスタンスは指定できません。

ALTER_QUEUE_TABLE プロシージャ

このプロシージャは、キュー表の既存のプロパティを変更します。

構文

```
DBMS_AQADM.ALTER_QUEUE_TABLE (  
    queue_table      IN   VARCHAR2,  
    comment          IN   VARCHAR2      DEFAULT NULL,  
    primary_instance IN   BINARY_INTEGER DEFAULT NULL,  
    secondary_instance IN  BINARY_INTEGER DEFAULT NULL);
```

パラメータ

表 5-4 ALTER_QUEUE_TABLE プロシージャのパラメータ

パラメータ	説明
queue_table	作成するキュー表の名前。
comment	キュー表に関するユーザー指定の説明を変更します。このユーザー・コメントは、キュー・カタログに追加されます。デフォルト値は NULL で、値が変更されないことを意味します。
primary_instance	キュー表のプライマリ所有者。キュー表内のキューに対するキュー・モニターのスケジューリングと伝播は、このインスタンス内で行われます。 デフォルト値は NULL で、現行の値が変更されないことを意味します。
secondary_instance	プライマリ・インスタンスが使用不可の場合、キュー表はセカンダリ・インスタンスにフェイルオーバーします。 デフォルト値は NULL で、現行の値が変更されないことを意味します。

DROP_QUEUE_TABLE プロシージャ

このプロシージャは、既存のキュー表を削除します。キュー表を削除するには、キュー表内のすべてのキューを停止し、削除しておく必要があります。この処理を自動的に実行する force オプションを使用しない場合は、明示的に実行する必要があります。

構文

```
DEMS_AQADM.DROP_QUEUE_TABLE (  
    queue_table      IN    VARCHAR2,  
    force            IN    BOOLEAN DEFAULT FALSE,  
    auto_commit      IN    BOOLEAN DEFAULT TRUE);
```

パラメータ

表 5-5 DROP_QUEUE_TABLE プロシージャのパラメータ

パラメータ	説明
queue_table	削除するキュー表の名前。
force	FALSE: 表内にキューが存在する場合、操作は成功しません。これがデフォルトです。 TRUE: 表内のすべてのキューが自動的に停止および削除されます。
auto_commit	TRUE: 現行トランザクションがある場合は、DROP_QUEUE_TABLE 操作が実行される前にコミットされます。DROP_QUEUE_TABLE 操作は、コールから戻ると持続されます。これがデフォルトです。 FALSE: 操作は現行トランザクションの一部で、コール側がコミットを入力したときのみ持続されます。 注意: このパラメータは使用しないことをお勧めします。

CREATE_QUEUE プロシージャ

このプロシージャは、指定したキュー表にキューを作成します。

構文

```
DBMS_AQADM.CREATE_QUEUE (
    queue_name      IN          VARCHAR2,
    queue_table     IN          VARCHAR2,
    queue_type      IN          BINARY_INTEGER DEFAULT NORMAL_QUEUE,
    max_retries     IN          NUMBER          DEFAULT NULL,
    retry_delay     IN          NUMBER          DEFAULT 0,
    retention_time  IN          NUMBER          DEFAULT 0,
    dependency_tracking IN      BOOLEAN          DEFAULT FALSE,
    comment         IN          VARCHAR2       DEFAULT NULL,
    auto_commit     IN          BOOLEAN          DEFAULT TRUE);
```

パラメータ

表 5-6 CREATE_QUEUE プロシージャのパラメータ

パラメータ	説明
queue_name	作成するキューの名前。名前はスキーマ内で重複しないようにし、予約語については、『Oracle8i SQL リファレンス』のオブジェクト名ガイドラインに従う必要があります。
queue_table	キューを格納するキュー表の名前。
queue_type	作成されるキューが例外キューか標準キューかを指定します。 NORMAL_QUEUE: キューは標準キューです。これがデフォルトです。 EXCEPTION_QUEUE: キューは例外キューです。例外キューでは、デキュー操作のみ許可されます。
max_retries	REMOVE モードのデキューがメッセージ上で試行される回数を制限します。 デキュー実行後、アプリケーションがロールバックを発行するたびにカウントが増加します。指定した max_retries に達すると、メッセージは例外キューに移されます。 max_retries はすべての単一コンシューマ・キューと 8.1 互換の複数コンシューマ・キューでサポートされていますが、8.0 互換の複数コンシューマ・キューではサポートされていないことに注意してください。

表 5-6 CREATE_QUEUE プロシージャのパラメータ

パラメータ	説明
retry_delay	<p>アプリケーションのロールバック後、このメッセージの再処理をスケジュールするまでの遅延時間（秒数）。</p> <p>デフォルトは 0（ゼロ）で、メッセージを最も迅速に取り出すことができます。このパラメータは、max_retries が 0（ゼロ）に設定されている場合は無効です。rety_delay は、単一コンシューマ・キューと 8.1 互換の複数コンシューマ・キューでサポートされていますが、8.0 互換の複数コンシューマ・キューではサポートされていないことに注意してください。</p>
retention_time	<p>キューからデキューされた後に、メッセージがキュー表に保持される秒数。</p> <p>INFINITE: メッセージは無期限で保持されます。</p> <p>数値：メッセージを保持する秒数。デフォルトは 0（ゼロ）で、保持されません。</p>
dependency_tracking	<p>将来使用のために確保。</p> <p>FALSE: これがデフォルトです。</p> <p>TRUE: このリリースでは指定できません。</p>
comment	<p>キューに関するユーザー指定の説明。このユーザー・コメントは、キュー・カタログに追加されます。</p>
auto_commit	<p>TRUE: 現行トランザクションがある場合は、CREATE_QUEUE 操作が実行される前にコミットされます。CREATE_QUEUE 操作は、コールから戻ると持続されます。これがデフォルトです。</p> <p>FALSE: 操作は現行トランザクションの一部で、コール側がコミットを入力したときのみ持続されます。</p> <p>注意：このパラメータは使用しないことをお勧めします。</p>

使用上の注意

すべてのキュー名はスキーマ内で重複しないようにしてください。CREATE_QUEUE でキューが作成された後、START_QUEUE をコールするとキューが使用可能になります。デフォルトでは、キューはエンキューとデキューともに使用禁止で作成されます。

CREATE_NP_QUEUE プロシージャ

非永続の RAW キューを作成します。

構文

```
DBMS_AQADM.CREATE_NP_QUEUE (
    queue_name          IN          VARCHAR2,
    multiple_consumers  IN          BOOLEAN DEFAULT FALSE,
    comment              IN          VARCHAR2 DEFAULT NULL);
```

パラメータ

表 5-7 CREATE_NP_QUEUE プロシージャのパラメータ

パラメータ	説明
queue_name	作成する非永続キューの名前。名前はスキーマ内で重複しないようにし、予約語については、『Oracle8i SQL リファレンス』のオブジェクト名ガイドラインに従う必要があります。
multiple_consumers	FALSE: 表内で作成されたキューには、1 つのメッセージに対して 1 つのコンシューマしか設定できません。これがデフォルトです。 TRUE: 表内で作成されたキューには、1 つのメッセージに対して複数のコンシューマを設定できます。 非永続キューは、ユーザーが作成したキュー表からこの特性を継承しないため、このパラメータはキュー・レベルで識別されることに注意してください。
comment	キューに関するユーザー指定の説明。このユーザー・コメントは、キュー・カタログに追加されます。

使用上の注意

キューは、単一コンシューマ・キューまたは複数コンシューマ・キューのいずれかです。すべてのキュー名はスキーマ内で重複しないようにしてください。キューは、キュー名が指定している同じスキーマ内の 8.1 互換のシステム作成キュー表 (AQ\$_MEM_SC または AQ\$_MEM_MC) に作成されます。

キュー名がスキーマ名を指定していない場合、キューはログイン・ユーザーのスキーマ内に作成されます。CREATE_NP_QUEUE でキューが作成された後、START_QUEUE をコールするとそのキューが使用可能になります。デフォルトでは、キューはエンキューとデキューともに使用禁止で作成されます。

非永続キューからはデキューできません。非永続キューからメッセージを取り出す唯一の方法は、OCI 通知メカニズムを使用することです。

非永続キューでは、listen コールは起動できません。

ALTER_QUEUE プロシージャ

このプロシージャは、キューの既存プロパティを変更します。パラメータ `max_retries`、`retention_time` および `retry_delay` は、非永続キューではサポートされていません。

構文

```
DBMS_AQADM.ALTER_QUEUE (
    queue_name      IN    VARCHAR2,
    max_retries     IN    NUMBER    DEFAULT NULL,
    retry_delay     IN    NUMBER    DEFAULT NULL,
    retention_time  IN    NUMBER    DEFAULT NULL,
    auto_commit     IN    BOOLEAN  DEFAULT TRUE,
    comment         IN    VARCHAR2 DEFAULT NULL);
```

パラメータ

表 5-8 ALTER_QUEUE プロシージャのパラメータ

パラメータ	説明
queue_name	変更するキューの名前。
max_retries	REMOVE モードのデキューがメッセージ上で試行される回数を制限します。 デキュー実行後、アプリケーションがロールバックを発行するたびにカウントが増加します。試行回数内で時間切れとなった場合、その後の再試行は行われません。デフォルトは NULL で、値が変更されないことを意味します。 max_retries はすべての単一コンシューマ・キューと 8.1 互換の複数コンシューマ・キューでサポートされていますが、8.0 互換の複数コンシューマ・キューではサポートされていないことに注意してください。
retry_delay	アプリケーションのロールバック後、このメッセージの再処理をスケジュールするまでの遅延時間（秒数）。デフォルトは NULL で、値が変更されないことを意味します。 retry_delay は単一コンシューマ・キューと 8.1 互換の複数コンシューマ・キューでサポートされていますが、8.0 互換の複数コンシューマ・キューではサポートされていないことに注意してください。
retention_time	デキュー後、メッセージがキュー表内に保持される時間（秒数）。デフォルトは NULL で、値が変更されないことを意味します。

表 5-8 ALTER_QUEUE プロシージャのパラメータ

パラメータ	説明
auto_commit	TRUE: 現行トランザクションがある場合は、ALTER_QUEUE 操作が実行される前にコミットされます。ALTER_QUEUE 操作は、コールから戻ると持続されます。これがデフォルトです。 FALSE: 操作は現行トランザクションの一部で、コール側がコミットを入力したときのみ持続されます。 注意: このパラメータは使用しないことをお勧めします。
comment	キューに関するユーザー指定の説明。このユーザー・コメントは、キュー・カタログに追加されます。デフォルト値は NULL で、値が変更されないことを意味します。

DROP_QUEUE プロシージャ

このプロシージャは、既存のキューを削除します。DROP_QUEUE は、STOP_QUEUE をコールして、キューに対するエンキューとデキューを使用禁止にしないと許可されません。すべてのキュー・データが、削除操作の一部として削除されます。

構文

```
DBMS_AQADM.DROP_QUEUE (  
    queue_name      IN    VARCHAR2,  
    auto_commit     IN    BOOLEAN DEFAULT TRUE);
```

パラメータ

表 5-9 DROP_QUEUE プロシージャのパラメータ

パラメータ	説明
queue_name	削除するキューの名前。
auto_commit	TRUE: 現行トランザクションがある場合は、DROP_QUEUE 操作が実行される前にコミットされます。DROP_QUEUE 操作は、コールから戻ると持続されます。これがデフォルトです。 FALSE: 操作は現行トランザクションの一部で、コール側がコミットを入力したときのみ持続されます。 注意: このパラメータは使用しないことをお勧めします。

START_QUEUE プロシージャ

このプロシージャは、指定したキューに対するエンキューまたはデキュー（あるいはその両方）を使用可能にします。

キューの作成後に、管理者は START_QUEUE を使用してキューを使用可能にする必要があります。デフォルトでは、ENQUEUE と DEQUEUE の両方が使用可能になります。例外キューでは、デキュー操作のみ許可されます。この操作はコールが完了すると有効になり、トランザクションの特性はありません。

構文

```
DBMS_AQADM.START_QUEUE (
    queue_name      IN      VARCHAR2,
    enqueue         IN      BOOLEAN DEFAULT TRUE,
    dequeue         IN      BOOLEAN DEFAULT TRUE);
```

パラメータ

表 5-10 START_QUEUE プロシージャのパラメータ

パラメータ	説明
queue_name	使用可能にするキューの名前。
enqueue	このキューで ENQUEUE を使用可能にするかどうかを指定します。 TRUE: ENQUEUE を使用可能にします。これがデフォルトです。 FALSE: 現在の設定を変更しません。
dequeue	このキューで DEQUEUE を使用可能にするかどうかを指定します。 TRUE: DEQUEUE を使用可能にします。これがデフォルトです。 FALSE: 現在の設定を変更しません。

STOP_QUEUE プロシージャ

このプロシージャは、指定したキューに対するエンキューまたはデキュー（あるいはその両方）を使用禁止にします。

デフォルトでは、このプロシージャをコールすると ENQUEUE と DEQUEUE の両方が使用禁止になります。キューに対する未処理のトランザクションがある場合、キューは停止できません。この操作はコールが完了すると有効になり、トランザクションの特性はありません。

構文

```
DBMS_AQADM.STOP_QUEUE (
    queue_name      IN   VARCHAR2,
    enqueue         IN   BOOLEAN DEFAULT TRUE,
    dequeue         IN   BOOLEAN DEFAULT TRUE,
    wait            IN   BOOLEAN DEFAULT TRUE);
```

パラメータ

表 5-11 STOP_QUEUE プロシージャのパラメータ

パラメータ	説明
queue_name	使用禁止にするキューの名前。
enqueue	このキューで ENQUEUE を使用禁止にするかどうかを指定します。 TRUE: ENQUEUE を使用禁止にします。これがデフォルトです。 FALSE: 現在の設定を変更しません。
dequeue	このキューで DEQUEUE を使用禁止にするかどうかを指定します。 TRUE: DEQUEUE を使用禁止にします。これがデフォルトです。 FALSE: 現在の設定を変更しません。
wait	未処理のトランザクションの完了を待つかどうかを指定します。 TRUE: 未処理のトランザクションがある場合は待機します。この状態では、新規トランザクションは、このキューへのエンキューまたはこのキューからのデキューを許可されません。 FALSE: 正常終了かエラーかをすぐに戻します。

GRANT_SYSTEM_PRIVILEGE プロシージャ

このプロシージャは、ユーザーとロールに AQ システム権限を付与します。権限には、ENQUEUE_ANY、DEQUEUE_ANY および MANAGE_ANY があります。初期設定では、SYS と SYSTEM のみこのプロシージャを正常に使用できます。

構文

```
DBMS_AQADM.GRANT_SYSTEM_PRIVILEGE (  
    privilege          IN    VARCHAR2,  
    grantee            IN    VARCHAR2,  
    admin_option       IN    BOOLEAN := FALSE);
```

パラメータ

表 5-12 GRANT_SYSTEM_PRIVILEGE プロシージャのパラメータ

パラメータ	説明
privilege	<p>付与する AQ システム権限。ENQUEUE_ANY、DEQUEUE_ANY および MANAGE_ANY から選択できます。</p> <p>各システム権限に許可される操作は次のとおりです。</p> <p>ENQUEUE_ANY: この権限を付与されたユーザーは、データベース内のすべてのキューにメッセージをエンキューできます。</p> <p>DEQUEUE_ANY: この権限を付与されたユーザーは、データベース内のすべてのキューからメッセージをデキューできます。</p> <p>MANAGE_ANY: この権限を付与されたユーザーは、データベース内のすべてのスキーマで DBMS_AQADM コールを実行できます。</p>
grantee	<p>権限受領者。権限受領者には、ユーザー、ロールまたは PUBLIC ロールを指定できます。</p>
admin_option	<p>システム権限を ADMIN オプション付きで付与するかどうかを指定します。</p> <p>ADMIN オプション付きで権限が付与されると、権限受領者はこのプロシージャを使用して、他のユーザーまたはロールにシステム権限を付与できます。デフォルトは FALSE です。</p>

REVOKE_SYSTEM_PRIVILEGE プロシージャ

このプロシージャは、ユーザーとロールから AQ システム権限を取り消します。権限には、ENQUEUE_ANY、DEQUEUE_ANY および MANAGE_ANY があります。システム権限の ADMIN オプションは、選択的に取り消すことはできません。

構文

```
DBMS_AQADM.REVOKE_SYSTEM_PRIVILEGE (  
    privilege      IN   VARCHAR2,  
    grantee        IN   VARCHAR2);
```

パラメータ

表 5-13 REVOKE_SYSTEM_PRIVILEGE プロシージャのパラメータ

パラメータ	説明
privilege	取り消す AQ システム権限。ENQUEUE_ANY、DEQUEUE_ANY および MANAGE_ANY から選択できます。 システム権限の ADMIN オプションは、選択的に取り消すことはできません。
grantee	権限受領者。権限受領者には、ユーザー、ロールまたは PUBLIC ロールを指定できます。

GRANT_QUEUE_PRIVILEGE プロシージャ

このプロシージャは、ユーザーとロールにキューの権限を付与します。権限は、ENQUEUE または DEQUEUE です。初期設定では、キュー表の所有者のみこのプロシージャを使用してそのキューの権限を付与できます。

構文

```
DBMS_AQADM.GRANT_QUEUE_PRIVILEGE (  
    privilege      IN   VARCHAR2,  
    queue_name     IN   VARCHAR2,  
    grantee        IN   VARCHAR2,  
    grant_option    IN   BOOLEAN := FALSE);
```

パラメータ

表 5-14 GRANT_QUEUE_PRIVILEGE プロシージャのパラメータ

パラメータ	説明
privilege	付与する AQ キュー権限。ENQUEUE、DEQUEUE および ALL から選択できます。ALL は、ENQUEUE と DEQUEUE 両方の権限を付与します。
queue_name	キュー名。
grantee	権限受領者。権限受領者には、ユーザー、ロールまたは PUBLIC ロールを指定できます。
grant_option	アクセス権限を GRANT オプション付きで付与するかどうかを指定します。 GRANT オプション付きで権限が付与されると、権限受領者はキュー表の所有権に関係なく、このプロシージャを使用して他のユーザーまたはロールにアクセス権限を付与できます。デフォルトは FALSE です。

REVOKE_QUEUE_PRIVILEGE プロシージャ

このプロシージャは、ユーザーとロールからキューの権限を取り消します。権限は、ENQUEUE または DEQUEUE です。権限を取り消すには、取消し実行者がその権限の付与者である必要があります。GRANT オプションを使用して付与された権限は、付与者の権限が取り消されると、同様に取り消されます。

構文

```
DBMS_AQADM.REVOKE_QUEUE_PRIVILEGE (  
    privilege      IN      VARCHAR2,  
    queue_name     IN      VARCHAR2,  
    grantee        IN      VARCHAR2);
```

パラメータ

表 5-15 REVOKE_QUEUE_PRIVILEGE プロシージャのパラメータ

パラメータ	説明
privilege	取り消す AQ キュー権限。ENQUEUE、DEQUEUE および ALL から選択できます。ALL は、ENQUEUE と DEQUEUE 両方の権限を付与します。
queue_name	キュー名。
grantee	権限受領者。権限受領者には、ユーザー、ロールまたは PUBLIC ロールを指定できます。権限が GRANT オプションを使用して付与された場合は、その権限も取り消されます。

ADD_SUBSCRIBER プロシージャ

このプロシージャは、キューにデフォルトのサブスクライバを追加します。

構文

```
DBMS_AQADM.ADD_SUBSCRIBER (  
    queue_name      IN      VARCHAR2,  
    subscriber      IN      sys.aq$_agent,  
    rule            IN      VARCHAR2 DEFAULT NULL);
```

パラメータ

表 5-16 ADD_SUBSCRIBER プロシージャのパラメータ

パラメータ	説明
queue_name	キュー名。
subscriber	かわりにサブスクリプションを定義しようとしているエージェント。
rule	メッセージ・プロパティ、メッセージ・データ・プロパティおよび PL/SQL ファンクションに基づいた条件式。 ルールは SQL 問合せの WHERE 句と同様の構文を使用して、ブール式で指定されます。このブール式には、メッセージ・プロパティの状態、ユーザー・データの プロパティ（オブジェクト・ペイロードのみ）、および PL/SQL または SQL ファンクション（SQL 問合せの WHERE 句で指定）の状態を組み込むことができます。現在サポートされているメッセージ・プロパティは、priority と corrid です。 メッセージ・ペイロード（オブジェクト・ペイロード）のルールを指定するには、句にオブジェクト型の属性を使用します。各属性の前に修飾子として tab.user_data を付加して、ペイロードを格納するキュー表の特定の列を示します。ルールのパラメータは 4000 バイト以内です。

使用上の注意

プログラムは、特定の受信者リストまたはデフォルトのサブスクライバ・リストに対してメッセージをエンキューできます。この操作は、複数コンシューマが許可されているキューでのみ成功します。この操作はすぐに有効となり、含まれるトランザクションがコミットされます。このコール後に実行されたエンキュー要求は、新しい動作を反映します。

ルール内の任意の文字列を次に示します。

```
rule    => 'PRIORITY <= 3 AND CORRID =  ''FROM JAPAN'''
```

すべて一重引用符が使用されていることに注意してください。

ALTER_SUBSCRIBER プロシージャ

このプロシージャは、指定したキューに対するサブスクライバの既存プロパティを変更します。変更できるのはルールのみです。

構文

```
DBMS_AQADM.ALTER_SUBSCRIBER (  
    queue_name      IN      VARCHAR2,  
    subscriber      IN      sys.aq$_agent,  
    rule            IN      VARCHAR2);
```

パラメータ

表 5-17 ALTER_SUBSCRIBER プロシージャのパラメータ

パラメータ	説明
queue_name	キュー名。
subscriber	サブスクリプションを変更するエージェント。4-4 ページの「 エー ジェント 」を参照してください。
rule	メッセージ・プロパティ、メッセージ・データ・プロパティおよび PL/SQL ファンクションに基づいた条件式。 注意： ルールのパラメータは 4000 バイト以内です。ルールを削除す るには、ルール・パラメータを NULL に設定します。

REMOVE_SUBSCRIBER プロシージャ

このプロシージャは、キューからデフォルトのサブスクライバを取り消します。この操作は
すぐに有効となり、含まれるトランザクションがコミットされます。既存のメッセージ内の
サブスクライバへの参照は、この操作の一部としてすべて削除されます。

構文

```
DBMS_AQADM.REMOVE_SUBSCRIBER (  
    queue_name      IN      VARCHAR2,  
    subscriber      IN      sys.aq$_agent);
```

パラメータ

表 5-18 REMOVE_SUBSCRIBER プロシージャのパラメータ

パラメータ	説明
queue_name	キュー名。
subscriber	削除するエージェント。4-4 ページの「 エージェント 」を参照してください。

SCHEDULE_PROPAGATION プロシージャ

このプロシージャは、キューから特定の DB リンクによって識別される宛先へのメッセージ伝播をスケジュールします。

宛先に NULL を指定すると、同じデータベース内の他のキューにもメッセージを伝播できます。同じ宛先にメッセージの受信者が複数存在する場合は、同じキューにあるか、異なるキューにあるかに関係なく、そのすべてに同時に伝播されます。

構文

```
DBMS_AQADM.SCHEDULE_PROPAGATION (  
    queue_name      IN      VARCHAR2,  
    destination     IN      VARCHAR2 DEFAULT NULL,  
    start_time      IN      DATE      DEFAULT SYSDATE,  
    duration        IN      NUMBER   DEFAULT NULL,  
    next_time       IN      VARCHAR2 DEFAULT NULL,  
    latency         IN      NUMBER   DEFAULT 60);
```

パラメータ

表 5-19 SCHEDULE_PROPAGATION プロシージャのパラメータ

パラメータ	説明
queue_name	伝播対象のメッセージがあるソース・キューの名前。スキーマ名を含みます。 スキーマ名が指定されない場合は、デフォルトで管理ユーザーのスキーマ名に設定されます。

表 5-19 SCHEDULE_PROPAGATION プロシージャのパラメータ

パラメータ	説明
destination	宛先の DB リンク。 この宛先の受信者に対するソース・キュー内のメッセージが伝播されます。宛先が NULL の場合は、ローカル・データベースが宛先となり、メッセージはローカル・データベース内の他のキューに伝播されます。このフィールドの長さは 128 バイトに制限されており、名前が完全に修飾されていない場合は、デフォルトのドメイン名が使用されます。
start_time	ソース・キューから宛先へのメッセージに対する伝播枠の初期起動時間。
duration	伝播枠の継続期間（秒数）。 NULL 値の場合、伝播枠は無期限か、または伝播スケジュールが取り消されるまで継続します。
next_time	現在の伝播枠の終了から次の伝播枠の開始を計算する日付関数。 この値が NULL の場合は、現在の枠が終了すると伝播は停止されます。たとえば、毎日同時刻に枠を起動するには、next_time に 'SYSDATE + 1 - duration/86400' と指定します。
latency	伝播枠内にエンキュー後、メッセージが伝播されるまでの最大待機時間（秒数）。 例：待ち時間が 60 秒で伝播枠にある場合は、伝播するメッセージがないと、その宛先に対するそのキューのメッセージは、最低 60 秒間伝播されません。 指定した宛先に対してメッセージを伝播するためにキューを再度チェックするまでの時間は、最短で 60 秒です。待ち時間が 600 秒の場合、キューは 10 分間チェックされず、待ち時間が 0（ゼロ）の場合は、宛先に対するメッセージがエンキューされるまでジョブ・キュー・プロセスが待機することになります。メッセージは、エンキューされるとすぐに伝播されます。

UNSCHEDULE_PROPAGATION プロシージャ

このプロシージャは、キューから特定の DB リンクで指定された宛先へのメッセージの伝播の旧スケジュールを取り消します。

構文

```
DBMS_AQADM.UNSCHEDULE_PROPAGATION (  
    queue_name      IN    VARCHAR2,  
    destination     IN    VARCHAR2 DEFAULT NULL);
```

パラメータ

表 5-20 UNSCHEDULE_PROPAGATION プロシージャのパラメータ

パラメータ	説明
queue_name	伝播対象のメッセージがあるソース・キューの名前。スキーマ名を含みます。 スキーマ名が指定されない場合は、デフォルトで管理ユーザーのスキーマ名に設定されます。
destination	宛先の DB リンク。 この宛先の受信者に対するソース・キュー内のメッセージが伝播されます。宛先が NULL の場合は、ローカル・データベースが宛先となり、メッセージはローカル・データベース内の他のキューに伝播されます。このフィールドの長さは 128 バイトに制限されており、名前が完全に修飾されていない場合は、デフォルトのドメイン名が使用されます。

VERIFY_QUEUE_TYPES プロシージャ

このプロシージャは、ソース・キューと宛先キューの型が同じであるかどうかを検証します。検証結果は、sys.aq\$_message_types 表に格納され、このコマンドの以前の出力がすべて上書きされます。

構文

```
DBMS_AQADM.VERIFY_QUEUE_TYPES (  
    src_queue_name  IN    VARCHAR2,  
    dest_queue_name IN    VARCHAR2,  
    destination     IN    VARCHAR2 DEFAULT NULL,  
    rc              OUT   BINARY_INTEGER);
```

パラメータ

表 5-21 VERIFY_QUEUE_TYPES プロシージャのパラメータ

パラメータ	説明
src_queue_name	伝播対象のメッセージがあるソース・キューの名前。スキーマ名を含みます。 スキーマ名が指定されていない場合は、デフォルトでユーザーのスキーマ名が設定されます。
dest_queue_name	メッセージが伝播される宛先キューの名前。スキーマ名を含みます。 スキーマ名が指定されていない場合は、デフォルトでユーザーのスキーマ名が設定されます。
destination	宛先の DB リンク。 この宛先の受信者に対するソース・キュー内のメッセージが伝播されます。宛先が NULL の場合は、ローカル・データベースが宛先となり、メッセージはローカル・データベース内の他のキューに伝播されます。このフィールドの長さは 128 バイトに制限されており、名前が完全に修飾されていない場合は、デフォルトのドメイン名が使用されます。
rc	プロシージャの結果を示すリターン・コード。 エラーがなく、ソースと宛先のキュー・タイプが一致している場合、結果は 1 になります。一致していない場合、結果は 0（ゼロ）になります。Oracle エラーが発生した場合は、rc に戻されます。

ALTER_PROPAGATION_SCHEDULE プロシージャ

このプロシージャは、伝播スケジュールのパラメータを変更します。

構文

```
DBMS_AQADM.ALTER_PROPAGATION_SCHEDULE (  
  queue_name      IN      VARCHAR2,  
  destination     IN      VARCHAR2 DEFAULT NULL,  
  duration        IN      NUMBER  DEFAULT NULL,  
  next_time       IN      VARCHAR2 DEFAULT NULL,  
  latency         IN      NUMBER  DEFAULT 60);
```


パラメータ

表 5-22 ALTER_PROPAGATION_SCHEDULE プロシージャのパラメータ

パラメータ	説明
queue_name	<p>伝播対象のメッセージがあるソース・キューの名前。スキーマ名を含みます。</p> <p>スキーマ名が指定されていない場合は、デフォルトでユーザーのスキーマ名が設定されます。</p>
destination	<p>宛先の DB リンク。</p> <p>この宛先の受信者に対するソース・キュー内のメッセージが伝播されます。宛先が NULL の場合は、ローカル・データベースが宛先となり、メッセージはローカル・データベース内の他のキューに伝播されます。このフィールドの長さは 128 バイトに制限されており、名前が完全に修飾されていない場合は、デフォルトのドメイン名が使用されます。</p>
duration	<p>伝播枠の継続期間（秒数）。</p> <p>NULL 値の場合、伝播枠は無期限か、または伝播スケジュールが取り消されるまで継続します。</p>
next_time	<p>現在の伝播枠の終了から次の伝播枠の開始を計算する日付ファンクション。</p> <p>この値が NULL の場合は、現在の枠が終了すると伝播は停止されます。たとえば、毎日同時刻に枠を起動するには next_time に 'SYSDATE + 1 - duration/86400' と指定します。</p>
latency	<p>伝播枠内にエンキュー後、メッセージが伝播されるまでの最大待機時間（秒数）。</p> <p>デフォルト値は 60 です。</p> <p>注意：このコールに対して待ち時間が指定されないと、待ち時間は既存の値をデフォルト値で上書きします。</p> <p>たとえば、待ち時間が 60 秒で伝播枠にある場合は、伝播するメッセージがないと、その宛先に対するそのキューのメッセージは、最低 60 秒間伝播されません。指定した宛先に対してメッセージを伝播するためにキューを再度チェックするまでの時間は、最短で 60 秒です。待ち時間が 600 の場合、キューは 10 分間チェックされず、待ち時間が 0（ゼロ）の場合は、宛先に対するメッセージがエンキューされるまでジョブ・キュー・プロセスが待機することになります。メッセージは、エンキューされるとすぐに伝播されます。</p>

ENABLE_PROPAGATION_SCHEDULE プロシージャ

このプロシージャは、以前に使用禁止にした伝播スケジュールを使用可能にします。

構文

```
DBMS_AQADM.ENABLE_PROPAGATION_SCHEDULE (  
    queue_name      IN      VARCHAR2,  
    destination     IN      VARCHAR2 DEFAULT NULL);
```

パラメータ

表 5-23 ENABLE_PROPAGATION_SCHEDULE プロシージャのパラメータ

パラメータ	説明
queue_name	伝播対象のメッセージがあるソース・キューの名前。スキーマ名を含みます。 スキーマ名が指定されていない場合は、デフォルトでユーザーのスキーマ名が設定されます。
destination	宛先の DB リンク。 この宛先の受信者に対するソース・キュー内のメッセージが伝播されます。宛先が NULL の場合は、ローカル・データベースが宛先となり、メッセージはローカル・データベース内の他のキューに伝播されます。このフィールドの長さは 128 バイトに制限されており、名前が完全に修飾されていない場合は、デフォルトのドメイン名が使用されます。

DISABLE_PROPAGATION_SCHEDULE プロシージャ

このプロシージャは、伝播スケジュールを無効にします。

構文

```
DBMS_AQADM.DISABLE_PROPAGATION_SCHEDULE (  
    queue_name      IN      VARCHAR2,  
    destination     IN      VARCHAR2 DEFAULT NULL);
```

パラメータ

表 5-24 DISABLE_PROPAGATION_SCHEDULE プロシージャのパラメータ

パラメータ	説明
queue_name	伝播対象のメッセージがあるソース・キューの名前。スキーマ名を含みます。 スキーマ名が指定されていない場合は、デフォルトでユーザーのスキーマ名が設定されます。
destination	宛先の DB リンク。 この宛先の受信者に対するソース・キュー内のメッセージが伝播されます。宛先が NULL の場合は、ローカル・データベースが宛先となり、メッセージはローカル・データベース内の他のキューに伝播されます。このフィールドの長さは 128 バイトに制限されており、名前が完全に修飾されていない場合は、デフォルトのドメイン名が使用されます。

MIGRATE_QUEUE_TABLE プロシージャ

このプロシージャは、8.0 互換キュー表から 8.1 互換キュー表へのアップグレード、または 8.1 互換キュー表から 8.0 互換キュー表へのダウングレードを実行します。

構文

```
DBMS_AQADM.MIGRATE_QUEUE_TABLE (  
    queue_table IN VARCHAR2,  
    compatible IN VARCHAR2);
```

パラメータ

表 5-25 MIGRATE_QUEUE_TABLE プロシージャのパラメータ

パラメータ	説明
queue_table	移行するキュー表の名前を指定します。
compatible	8.0 互換キュー表を 8.1 互換キュー表にアップグレードするには '8.1' を設定します。8.1 互換キュー表を 8.0 互換キュー表にダウングレードするには '8.0' に設定します。

DBMS_BACKUP_RESTORE

DBMS_BACKUP_RESTORE パッケージには、Windows NT プラットフォーム上でファイル名を正規化する PL/SQL プロシージャがあります。

注意： このプロシージャは、8.1.6 以前の Oracle リリース、または UNIX ベースの Oracle インストレーションでは使用しないでください。

Windows NT プラットフォームの Oracle に対するファイル名の正規化

リリース 8.1.6 以降の Oracle では、ファイル名は正しく正規化されます。ただし、以前のリリースの制御ファイルおよびリカバリ・カタログのファイル名を正規化するためには、このプロシージャを使用する必要があります。

8.1.6 以前のリリースでは、ファイル名の正規化方法に欠陥があったため、2つの異なるファイル名で1つの物理ファイルを参照することがありました。DBMS_BACKUP_RESTORE は、これを修正し、制御ファイルおよびリカバリ・カタログで参照されたすべての物理ファイルを、Oracle が正確に識別できるようになりました。

このパッケージの詳細と詳しい実行手順は、『Oracle8i 移行ガイド』を参照してください。

このパッケージは、ストアド・プロシージャから一部の SQL データ定義言語 (Data Definition Language: DDL) 文へのアクセスを提供します。DDL では使用できない特殊な管理操作も提供します。

ALTER_COMPILE と ANALYZE_OBJECT プロシージャは、現行トランザクションをコミットし、操作を実行してから再度コミットします。

要件

このパッケージは、パッケージ所有者 SYS ではなく、コール・ユーザーの権限で実行されます。

サブプログラムの要約

表 7-1 DBMS_DDL パッケージのサブプログラム

サブプログラム	説明
7-2 ページの ALTER_COMPILE プロシージャ	PL/SQL オブジェクトをコンパイルします。
7-3 ページの ANALYZE_OBJECT プロシージャ	データベース・オブジェクトの統計情報を提供します。

ALTER_COMPILE プロシージャ

このプロシージャは、次の SQL 文と同じです。

```
ALTER PROCEDURE|FUNCTION|PACKAGE [<schema>.] <name> COMPILER [BODY]
```

名前を指定したオブジェクトがこのパッケージであるか、または依存するパッケージ（現在は STANDARD または DBMS_STANDARD）の場合、プロシージャは単に戻され、これらのパッケージは正常にコンパイルされます。

構文

```
DBMS_DDL.ALTER_COMPILE (  
    type    VARCHAR2,  
    schema  VARCHAR2,  
    name    VARCHAR2);
```

パラメータ

表 7-2 ALTER_COMPILE プロシージャのパラメータ

パラメータ	説明
type	PROCEDURE、FUNCTION、PACKAGE、PACKAGE BODY または TRIGGER のいずれかを設定します。
schema	スキーマ名。 NULL の場合は、現行スキーマ（大 / 小文字区別）が使用されます。
name	オブジェクトの名前（大 / 小文字区別）。

例外

表 7-3 ALTER_COMPILE プロシージャの例外

例外	説明
ORA-20000:	権限が不十分であるか、またはオブジェクトが存在しません。
ORA-20001:	リモート・オブジェクトであるためコンパイルできません。
ORA-20002:	オブジェクト型の値が正しくありません。 PACKAGE、PACKAGE BODY、PROCEDURE、FUNCTION または TRIGGER のいずれかを設定してください。

ANALYZE_OBJECT プロシージャ

このプロシージャは、指定された表、索引またはクラスタに関する統計情報を提供します。
このプロシージャは、次の SQL 文と同じです。

```
ANALYZE TABLE|CLUSTER|INDEX [<schema>.<name> [<method>] STATISTICS [SAMPLE <n>
[ROWS|PERCENT]]
```

構文

```
DBMS_DDL.ANALYZE_OBJECT (
    type          VARCHAR2,
    schema        VARCHAR2,
    name          VARCHAR2,
    method        VARCHAR2,
    estimate_rows NUMBER   DEFAULT NULL,
    estimate_percent NUMBER DEFAULT NULL,
    method_opt    VARCHAR2 DEFAULT NULL,
    partname      VARCHAR2 DEFAULT NULL);
```

パラメータ

表 7-4 ANALYZE_OBJECT プロシージャのパラメータ

パラメータ	説明
type	TABLE、CLUSTER または INDEX のいずれかを設定します。 いずれかを指定しないと、プロシージャは戻されるのみです。
schema	分析するオブジェクトのスキーマ。NULL の場合は現行スキーマ（大 / 小文字区別）が使用されます。
name	分析するオブジェクトの名前（大 / 小文字区別）。

表 7-4 ANALYZE_OBJECT プロシージャのパラメータ

パラメータ	説明
method	ESTIMATE、COMPUTE または DELETE のいずれかを設定します。 ESTIMATE を設定した場合は、estimate_rows または estimate_percent のいずれかを 0（ゼロ）以外に設定する必要があります。
estimate_rows	推定する行数。
estimate_percent	推定する行のパーセント。 estimate_rows が指定されている場合、このパラメータは無視されます。
method_opt	次の書式の分析方法オプション。 [FOR TABLE] [FOR ALL [INDEXED] COLUMNS] [SIZE n] [FOR ALL INDEXES]
partname	分析する特定のパーティション。

例外

表 7-5 ANALYZE_OBJECT プロシージャの例外

例外	説明
ORA-20000:	権限が不十分であるか、またはオブジェクトが存在しません。
ORA-20001:	オブジェクト型の値が正しくありません。 TABLE、INDEX または CLUSTER のいずれかを設定してください。
ORA-20002:	METHOD には、COMPUTE、ESTIMATE または DELETE のいずれかを設定する必要があります。

DBMS_DEBUG

DBMS_DEBUG は、Oracle Server における PL/SQL デバッガ・レイヤー、プローブへの PL/SQL の API です。

この API は、主にサーバー側のデバッガをインプリメントすることを目的としており、サーバー側の PL/SQL プログラム・ユニットをデバッグする方法を提供します。

注意： プログラム・ユニットという用語は、各種の PL/SQL プログラム（プロシージャ、ファンクション、パッケージ、パッケージ本体、トリガー、無名ブロック、オブジェクト型またはオブジェクト型本体）のことを指します。

DBMS_DEBUG の使用方法

サーバー側のコードをデバッグするには、2つのデータベース・セッションが必要です。1つはコードをデバッグ・モードで実行するセッション（ターゲット・セッション）、他の1つはそのターゲット・セッションを監視するセッション（デバッグ・セッション）です。

ターゲット・セッションは、DBMS_DEBUG で初期化コールを行うことでデバッグ可能になります。この結果、そのセッションにマークが付けられるため、PL/SQL インタプリタがデバッグ・モードで実行され、デバッグ・イベントが生成されます。デバッグ・イベントが生成されると、それらはセッションから転記されます。多くの場合、デバッグ・イベントには戻り通知が必要です。インタプリタは応答があるまで一時停止します。

この間に、デバッグ・セッション自体は DBMS_DEBUG を使用して初期化する必要があります。この結果、監視するターゲット・セッションが識別されます。次に、デバッグ・セッションは DBMS_DEBUG のエントリポイントをコールして、ターゲット・セッションから転記されたイベントを読み込み、ターゲット・セッションと通信します。

関連項目： [図 8-1](#) と [図 8-2](#) は、デバッグ対象のセッションとデバッグ・セッションにおける操作の例です。

DBMS_DEBUG は、PL/SQL コンパイラへのインタフェースは提供しませんが、コンパイラがオプションで生成するデバッグ情報には依存します。デバッグ情報がないと、パラメータまたは変数の値の検証や変更が実行できません。デバッグ情報を確実に生成する方法には、セッション・スイッチの設定または個別再コンパイルの2通りの方法があります。

セッション・スイッチを設定するには、次の文を入力します。

```
ALTER SESSION SET PLSQL_DEBUG = true;
```

この文によって、コンパイラはセッションの残りの部分に関するデバッグ情報を生成します。既存の PL/SQL は再コンパイルしません。

既存の PL/SQL コードのデバッグ情報を生成するには、次の文のいずれかを使用します（2番目の文はパッケージまたは型の本体を再コンパイルします）。

```
ALTER [PROCEDURE | FUNCTION | PACKAGE | TRIGGER | TYPE] <name> COMPILE DEBUG;  
ALTER [PACKAGE | TYPE] <name> COMPILE DEBUG BODY;
```

図 8-1 ターゲット・セッション

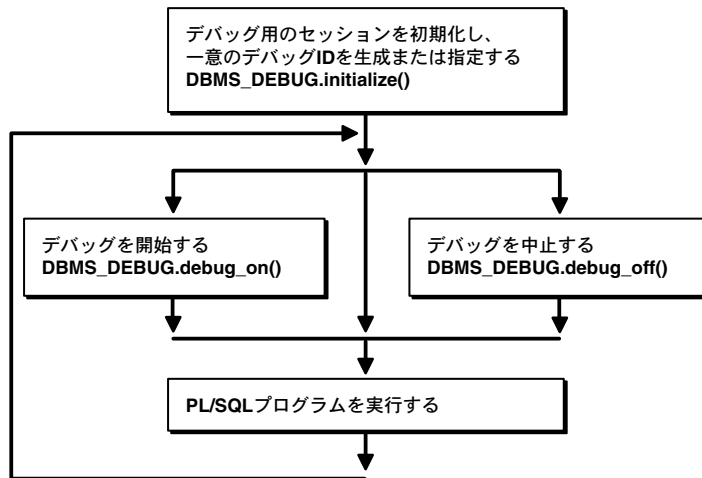


図 8-2 デバッグ・セッション

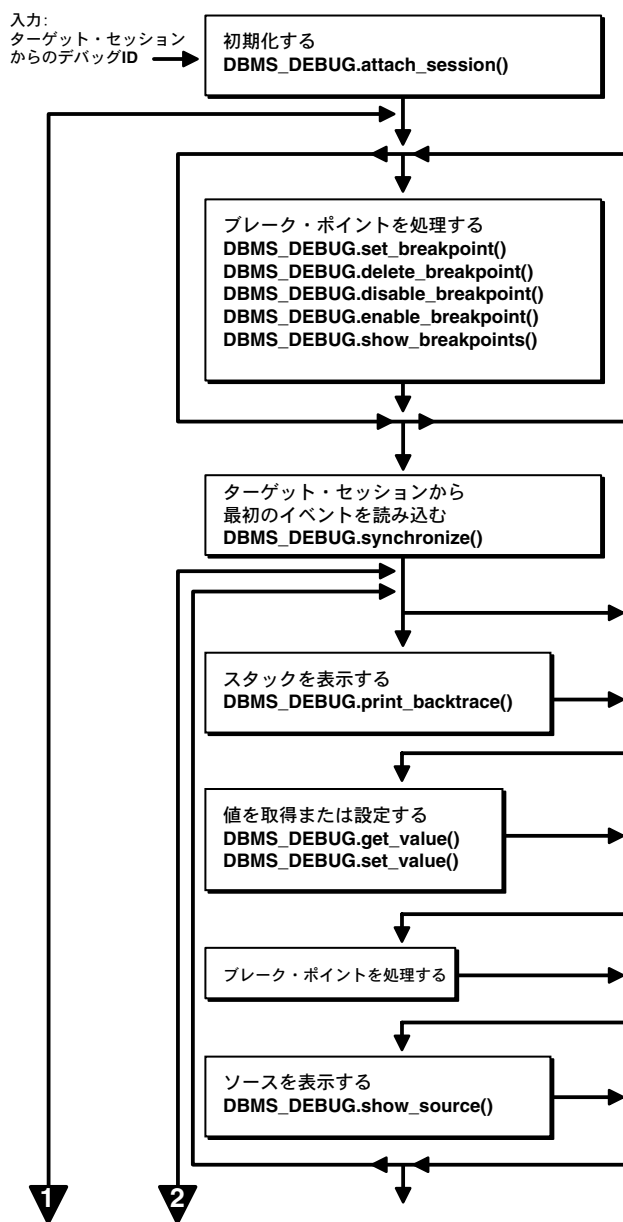
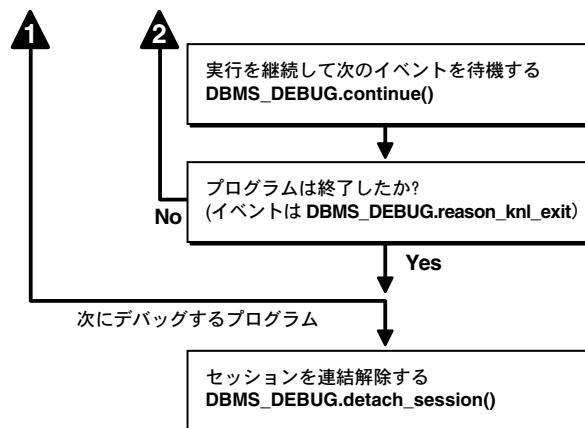


図 8-3 デバッグ・セッション（続き）



インタプリタの管理

インタプリタは、次の場合に実行を一時停止します。

1. インタプリタの起動時。実行前に、遅延ブレーク・ポイントをインストールできるようにするためです。
2. 使用可能なブレーク・ポイントを含んだ行に達したとき。
3. 関連のあるイベントが発生した行に達したとき。関連イベントのセットは、breakflags パラメータの DBMS_DEBUG.CONTINUE に渡されるフラグで指定されます。

使用上の注意

セッションの終了

セッション終了のイベントはありません。したがって、ターゲット・セッションが終了していないことを、デバッグ・セッションでチェックして確認する必要があります。ターゲット・セッションが終了した後に DBMS_DEBUG.SYNCHRONIZE をコールすると、タイムアウトするまでデバッグ・セッションがハングアップします。

遅延操作

図では、ターゲット・セッションの前にブレーク・ポイントを設定できることが示されています。これは確かに可能です。この場合、プローブはブレーク・ポイント要求をキャッシュして、最初の同期でターゲット・セッションに送信します。ただし、ブレーク・ポイント要求がこのように遅延した場合は次のようになります。

- `SET_BREAKPOINT` はブレーク・ポイント番号を設定しません（必要に応じて後で `SHOW_BREAKPOINTS` から取得できます）。
- `SET_BREAKPOINT` はブレーク・ポイント要求を検証しません。要求されたソース行が存在しない場合は、同期時にエラーが単的に発生し、ブレーク・ポイントは設定されません。

診断出力

プローブをデバッグするために、`DBMS_DEBUG` のコールの一部に対して *diagnostics* パラメータが用意されています。これらのパラメータは、`RDBMS` トレース・ファイルに診断出力を格納するかどうかを指定します。`RDBMS` トレース・ファイルに出力できない場合、このパラメータは無効になります。

型

PROGRAM_INFO この型はプログラムの位置を指定します。プログラム・ユニットの中の行番号を使用します。これは、スタックのバックトレース用とブレーク・ポイントの設定と検査用に使用されます。読取り専用フィールドは、ブレーク・ポイント操作に関してプローブでは現在無視されています。スタックのバックトレース用のみにプローブが設定します。

EntrypointName	ネストされたプロシージャまたはファンクション以外は NULL です。
LibunitType	同じ名前領域を共有するオブジェクト（プロシージャやパッケージ仕様部など）を一義化します。 詳細は、8-9 ページの「 Libunit 型 」を参照してください。

```

TYPE program_info IS RECORD
(
    -- The following fields are used when setting a breakpoint
    Namespace      BINARY_INTEGER, -- See 'NAMESPACES' section below.
    Name            VARCHAR2(30),    -- name of the program unit
    Owner           VARCHAR2(30),    -- owner of the program unit
    DbLink          VARCHAR2(30),    -- database link, if remote
    Line#           BINARY_INTEGER,
    -- Read-only fields (set by Probe when doing a stack backtrace)
    LibunitType     BINARY_INTEGER,
    EntrypointName  VARCHAR2(30)
);

```

RUNTIME_INFO この型は、実行プログラムに関するコンテキスト情報を提供します。

```

TYPE runtime_info IS RECORD
(
    Line#           BINARY_INTEGER, -- (duplicate of program.line#)
    Terminated     BINARY_INTEGER, -- has the program terminated?
    Breakpoint       BINARY_INTEGER, -- breakpoint number
    StackDepth       BINARY_INTEGER, -- number of frames on the stack
    InterpreterDepth BINARY_INTEGER, -- <reserved field>
    Reason           BINARY_INTEGER, -- reason for suspension
    Program          program_info    -- source location
);

```

BREAKPOINT_INFO この型は、ブレーク・ポイントに関して、現在の状態や配置されたプログラム・ユニットなどの情報を提供します。

```

TYPE breakpoint_info IS RECORD
(
    -- These fields are duplicates of 'program_info':
    Name            VARCHAR2(30),
    Owner           VARCHAR2(30),
    DbLink          VARCHAR2(30),
    Line#           BINARY_INTEGER,
    LibunitType     BINARY_INTEGER,
    Status          BINARY_INTEGER -- see breakpoint_status_* below
);

```

INDEX_TABLE この型は、索引表で使用可能な索引を戻すために、GET_INDEXES で使用されます。

TYPE index_table IS table of BINARY_INTEGER INDEX BY BINARY_INTEGER;

BACKTRACE_TABLE この型は、PRINT_BACKTRACE で使用されます。

TYPE backtrace_table IS TABLE OF program_info INDEX BY BINARY_INTEGER;

BREAKPOINT_TABLE この型は、SHOW_BREAKPOINTS で使用されます。

TYPE breakpoint_table IS TABLE OF breakpoint_info INDEX BY BINARY_INTEGER;

VC2_TABLE この型は、SHOW_SOURCE で使用されます。

TYPE vc2_table IS TABLE OF VARCHAR2(90) INDEX BY BINARY_INTEGER;

定数

ブレーク・ポイントの状態には次の値があります。

breakpoint_status_unused ブレーク・ポイントは使用されていません。

ブレーク・ポイントが使用されている場合、状態は次の値のマスクになります。

breakpoint_status_active 行ブレーク・ポイント。

breakpoints_status_disabled ブレーク・ポイントは現在使用できません。

breakpoint_status_remote 'shadow' ブレーク・ポイント（リモート・ブレーク・ポイントのローカル表示）。

名前領域 サーバー上のプログラム・ユニットは、異なる名前領域に常駐しています。ブレーク・ポイントの設定時には、希望する名前領域を指定してください。

1. Namespace_cursor にはカーソル（無名ブロック）が含まれています。
2. Namespace_pgkspec_or_toplevel には次のものが含まれています。
 - パッケージ仕様部。
 - 他のパッケージ、プロシージャまたはファンクション内にネストされていないプロシージャとファンクション。
 - オブジェクト型。
3. Namespace_pkg_body にはパッケージ本体と型本体が含まれています。
4. Namespace_trigger にはトリガーが含まれています。

Libunit 型 この値は、特定の名前領域のオブジェクトを一義化するために使用されます。これらの定数は、プローブがスタックのバックトレースを提供しているときに、PROGRAM_INFO で使用されます。

LibunitType_cursor
LibunitType_procedure
LibunitType_function
LibunitType_package
LibunitType_package_body
LibunitType_trigger
LibunitType_Unknown

ブレイク・フラグ この値は、クライアントに関連のあるイベントをプローブに通知するために、CONTINUE に対する breakflags パラメータで使用されます。これらのフラグは結合できます。

break_next_line	次のソース行でブレイクします（コールをスキップ）。
break_any_call	次のソース行でブレイクします（コールを開始）。
break_any_return	現行エントリポイントから戻された後ブレイク（現行ルーチンからコールされたエントリポイントはすべてスキップ）します。
break_return	次回エントリポイントが戻し処理の準備ができた時点でブレイクします。（現行エントリポイントからコールされたエントリポイントが含まれます。インタプリタが Proc2 をコールする Proc1 を実行している場合、break_return は Proc2 の終了時に停止します。）
break_exception	例外が発生したときにブレイクします。
break_handler	例外ハンドラが実行されたときにブレイクします。
abort_execution	実行を停止し、DBMS_DEBUG.CONTINUE がコールされるとすぐに、'exit' イベントを強制的に実行します。

情報フラグ このフラグは、`info_requested` パラメータとして、`SYNCHRONIZE`、`CONTINUE` および `GET_RUNTIME_INFO` に渡されます。

<code>info_getStackDepth</code>	スタックの現在の深さを取得します。
<code>info_getBreakpoint</code>	ブレーク・ポイント数を取得します。
<code>info_getLineinfo</code>	プログラム・ユニット情報を取得します。

中断理由 `CONTINUE` の実行後、プログラムは最後まで実行されるか、または途中の行でブレークします。

<code>reason_none</code>	
<code>reason_interpreter_starting</code>	インタプリタは起動中です。
<code>reason_breakpoint</code>	ブレーク・ポイントに到達しました。
<code>reason_enter</code>	プロシージャ・エントリ。
<code>reason_return</code>	プロシージャが戻ります。
<code>reason_finish</code>	プロシージャが終了しました。
<code>reason_line</code>	改行に到達しました。
<code>reason_interrupt</code>	割込みが発生しました。
<code>reason_exception</code>	例外が発生しました。
<code>reason_exit</code>	インタプリタは終了処理中です（旧形式）。
<code>reason_knl_exit</code>	カーネルは終了処理中です。
<code>reason_handler</code>	例外ハンドラを起動します。
<code>reason_timeout</code>	タイムアウトが発生しました。
<code>reason_instantiate</code>	インスタンス化・ブロック。
<code>reason_abort</code>	インタプリタは異常終了中です。

エラー・コード

この値は、デバッグ・セッション（SYNCHRONIZE、CONTINUE、SET_BREAKPOINT など）でコールされる様々なファンクションによって戻されます。PL/SQL 例外がクライアント / サーバーおよびサーバー / サーバーの境界を越えて発生した場合は、すべて例外となり、エラー・コードは戻されません。

success	正常終了。
---------	-------

GET_VALUE と SET_VALUE が戻すステータスは次のとおりです。

error_bogus_frame	該当するエントリポイントがスタックにありません。
error_no_debug_info	プログラムがデバッグ記号なしにコンパイルされました。
error_no_such_object	該当する変数またはパラメータがありません。
error_unknown_type	デバッグ情報を読み取ることができません。
error_indexed_table	オブジェクトが表で、索引が提供されていない場合に GET_VALUE で戻されます。
error_illegal_index	該当する要素がコレクション内に存在しません。
error_nullcollection	表がアトミック NULL です。
error_nullvalue	値が NULL です。

SET_VALUE が戻すステータスは次のとおりです。

error_illegal_value	制約違反。
error_illegal_null	制約違反。
error_value_malformed	指定された値を解釈できません。
error_other	その他のエラー。
error_name_incomplete	名前をスカラーに変換できません。

ブレーク・ポイント・ファンクションが戻すステータスは次のとおりです。

error_no_such_breakpt	該当するブレーク・ポイントがありません。
error_idle_breakpt	未使用のブレーク・ポイントは使用可能または使用禁止にできません。
error_bad_handle	指定されたプログラムにブレーク・ポイントを設定できません（存在していないか、またはセキュリティ違反です）。

一般的なエラー・コード（多数の DBMS_DEBUG サブプログラムが戻す）は次のとおりです。

<code>error_unimplemented</code>	機能がインプリメントされていません。
<code>error_deferred</code>	プログラムが実行されていません。操作は延期されました。
<code>error_exception</code>	サーバー上の DBMS_DEBUG またはプローブ・パッケージで例外が発生しました。
<code>error_communication</code>	タイムアウト以外のエラーが発生しました。
<code>error_timeout</code>	タイムアウトが発生しました。

例外

<code>illegal_init</code>	INITIALIZE の前に DEBUG_ON がコールされました。
---------------------------	------------------------------------

次の例外は、プロシージャ SELF_CHECK によって発生します。

<code>pipe_creation_failure</code>	パイプを作成できませんでした。
<code>pipe_send_failure</code>	パイプにデータを書き込めませんでした。
<code>pipe_receive_failure</code>	パイプからデータを読み込めませんでした。
<code>pipe_datatype_mismatch</code>	パイプ内のデータ型が正しくありませんでした。
<code>pipe_data_error</code>	データがパイプ内で混同されていました。

変数

<code>default_timeout</code>	タイムアウトの値（両方のセッションが使用します）。最小許容値は 1 秒です。この値が 0（ゼロ）に設定された場合は、大きい値（3600）が使用されます。
------------------------------	--

サブプログラムの要約

表 8-1 DBMS_DEBUG パッケージのサブプログラム

サブプログラム	説明
8-14 ページの PROBE_VERSION プロシージャ	サーバー上の DBMS_DEBUG のバージョン番号を戻します。
8-15 ページの SELF_CHECK プロシージャ	内部一貫性チェックを実行します。
8-16 ページの SET_TIMEOUT ファンクション	タイムアウト値を設定します。
8-17 ページの INITIALIZE ファンクション	ターゲット・セッションのデバッグ ID を設定します。
8-17 ページの DEBUG_ON プロシージャ	デバッグ・モードをオンにします。
8-18 ページの DEBUG_OFF プロシージャ	デバッグ・モードをオフにします。
8-19 ページの ATTACH_SESSION プロシージャ	デバッグ・セッションにターゲット・デバッグ ID に関する情報を通知します。
8-20 ページの SYNCHRONIZE ファンクション	プログラムの実行開始を待機します。
8-21 ページの SHOW_SOURCE プロシージャ	プログラム・ソースを取り出します。
8-23 ページの PRINT_BACKTRACE プロシージャ	スタックのバックトレースを印刷します。
8-24 ページの CONTINUE ファンクション	ターゲット・プログラムの実行を継続します。
8-25 ページの SET_BREAKPOINT ファンクション	プログラム・ユニットにブレーク・ポイントを設定します。
8-26 ページの DELETE_BREAKPOINT ファンクション	ブレーク・ポイントを削除します。
8-27 ページの DISABLE_BREAKPOINT ファンクション	ブレーク・ポイントを使用禁止にします。
8-28 ページの ENABLE_BREAKPOINT ファンクション	既存のブレーク・ポイントをアクティブ化します。
8-28 ページの SHOW_BREAKPOINTS プロシージャ	現行ブレーク・ポイントのリストを戻します。

表 8-1 DBMS_DEBUG パッケージのサブプログラム

サブプログラム	説明
8-29 ページの GET_VALUE ファンクション	現在実行中のプログラムから値を取得します。
8-32 ページの SET_VALUE ファンクション	現在実行中のプログラムに値を設定します。
8-34 ページの DETACH_SESSION プロシージャ	ターゲット・プログラムのデバッグを停止します。
8-34 ページの GET_RUNTIME_INFO ファンクション	現行プログラムに関する情報を戻します。
8-35 ページの GET_INDEXES ファンクション	索引表に対する一連の索引を戻します。
8-36 ページの EXECUTE プロシージャ	ターゲット・セッションで SQL または PL/SQL を実行します。

共通セクション

次に示すサブプログラムは、ターゲットまたはデバッグ・セッションのいずれでもコールできます。

- [PROBE_VERSION](#) プロシージャ
- [SELF_CHECK](#) プロシージャ
- [SET_TIMEOUT](#) ファンクション

PROBE_VERSION プロシージャ

このプロシージャは、サーバー上の DBMS_DEBUG のバージョン番号を戻します。

構文

```
DBMS_DEBUG.PROBE_VERSION (  
    major out BINARY_INTEGER,  
    minor out BINARY_INTEGER);
```


パラメータ

表 8-2 PROBE_VERSION プロシージャのパラメータ

パラメータ	説明
major	バージョン番号。
minor	リリース番号。機能が追加されるたびに増加します。

SELF_CHECK プロシージャ

このプロシージャは、内部一貫性チェックを実行します。SELF_CHECK は、プローブ・プロセスが通信可能かどうかを確認するために、通信テストも実行します。

SELF_CHECK が正常に終了しなかった場合は、このサーバーにインストールされている DBMS_DEBUG のバージョンが適切ではない可能性があります。解決方法は、正しいバージョンをインストールすることです (pbload.sql を実行すると、DBMS_DEBUG とその他の関連パッケージがロードされます)。

構文

```
DBMS_DEBUG.SELF_CHECK (  
    timeout IN binary_integer := 60);
```

パラメータ

表 8-3 SELF_CHECK プロシージャのパラメータ

パラメータ	説明
timeout	通信テストに使用するタイムアウト時間。デフォルトは 60 秒です。

例外

表 8-4 SELF_CHECK プロシージャの例外

例外	説明
OER-6516	プローブのバージョンに一貫性がありません。
pipe_creation_failure	パイプを作成できませんでした。
pipe_send_failure	パイプにデータを書き込めませんでした。
pipe_receive_failure	パイプからデータを読み込めませんでした。
pipe_datatype_mismatch	パイプ内のデータ型が正しくありませんでした。

表 8-4 SELF_CHECK プロシージャの例外

例外	説明
pipe_data_error	データがパイプ内で混同されていました。

これらはすべて致命的な例外です。プローブの正常な実行を妨げる重大な問題であることを示しています。

SET_TIMEOUT ファンクション

このファンクションは、タイムアウト値を設定し、新しいタイムアウト値を戻します。

構文

```
DBMS_DEBUG.SET_TIMEOUT (  
    timeout BINARY_INTEGER)  
RETURN BINARY_INTEGER;
```

パラメータ

表 8-5 SET_TIMEOUT ファンクションのパラメータ

パラメータ	説明
timeout	ターゲットとデバッグ・セッション間の通信に使用されるタイムアウト時間

TARGET SESSION セクション

次のサブプログラムは、ターゲット・セッション（デバッグ対象のセッション）で実行されます。

- [INITIALIZE ファンクション](#)
- [DEBUG_ON プロシージャ](#)
- [DEBUG_OFF プロシージャ](#)

INITIALIZE ファンクション

このファンクションは、デバッグ用にターゲット・セッションを初期化します。

構文

```
DBMS_DEBUG.INITIALIZE (  
    debug_session_id  IN VARCHAR2          := NULL,  
    diagnostics       IN BINARY_INTEGER := 0)  
RETURN VARCHAR2;
```

パラメータ

表 8-6 INITIALIZE ファンクションのパラメータ

パラメータ	説明
debug_session_id	セッション ID の名前。NULL の場合は、一意の ID が生成されます。
diagnostics	診断出力をトレースファイルにダンプするかどうかを示します。 0 = (デフォルト) 診断出力なし。 1 = 診断出力あり。

戻り値

新たに登録されたデバッグ・セッション ID (デバッグ ID)

DEBUG_ON プロシージャ

このプロシージャは、すべての PL/SQL がデバッグ・モードで実行されるように、ターゲット・セッションにマークを設定します。この処理は、デバッグの開始前に実行する必要があります。

構文

```
DBMS_DEBUG.DEBUG_ON (  
    no_client_side_plsql_engine BOOLEAN := TRUE,  
    immediate                   BOOLEAN := FALSE);
```

パラメータ

表 8-7 DEBUG_ON プロシージャのパラメータ

パラメータ	説明
no_client_side_plsql_engine	デバッグ・セッションがクライアント側の PL/SQL エンジンから起動されていない限り、デフォルト値のままにしてください。
immediate	TRUE の場合、インタプリタは標準モードで処理を継続せずに、コール中にすぐにデバッグ・モードに切り替わります。

注意： immediate が TRUE の場合、デバッグ・セッションは待機する必要があります。

DEBUG_OFF プロシージャ

このプロシージャは、そのセッションでデバッグを起動する必要がなくなったことをターゲット・セッションに通知します。セッションの終了前にこのファンクションをコールする必要はありません。

構文

```
DBMS_DEBUG.DEBUG_OFF;
```

パラメータ

なし

使用上の注意

サーバーは、このエントリポイントを特別には処理しません。したがって、このエントリポイントをデバッグしようとはします。

デバッグ・セッション・セクション

次のサブプログラムは、デバッグ・セッションでのみ実行してください。

- [ATTACH_SESSION](#) プロシージャ
- [SYNCHRONIZE](#) ファンクション
- [SHOW_SOURCE](#) プロシージャ
- [PRINT_BACKTRACE](#) プロシージャ
- [CONTINUE](#) ファンクション

- SET_BREAKPOINT ファンクション
- DELETE_BREAKPOINT ファンクション
- DISABLE_BREAKPOINT ファンクション
- ENABLE_BREAKPOINT ファンクション
- SHOW_BREAKPOINTS プロシージャ
- GET_VALUE ファンクション
- SET_VALUE ファンクション
- DETACH_SESSION プロシージャ
- GET_RUNTIME_INFO ファンクション
- GET_INDEXES ファンクション
- EXECUTE プロシージャ

ATTACH_SESSION プロシージャ

このプロシージャは、ターゲット・プログラムに関する情報をデバッグ・セッションに通知します。

構文

```
DBMS_DEBUG.ATTACH_SESSION (  
    debug_session_id  IN VARCHAR2,  
    diagnostics       IN BINARY_INTEGER := 0);
```

パラメータ

表 8-8 ATTACH_SESSION プロシージャのパラメータ

パラメータ	説明
debug_session_id	ターゲット・セッションの INITIALIZE コールで取得したデバッグ ID。
diagnostics	0（ゼロ）以外の場合に診断出力を生成します。

SYNCHRONIZE ファンクション

このファンクションは、ターゲット・プログラムがイベントを通知するまで待機します。
info_requested が NULL でない場合は、GET_RUNTIME_INFO がコールされます。

構文

```
DBMS_DEBUG.SYNCHRONIZE (  
    run_info          OUT  runtime_info,  
    info_requested IN   BINARY_INTEGER := NULL)  
RETURN BINARY_INTEGER;
```

パラメータ

表 8-9 SYNCHRONIZE ファンクションのパラメータ

パラメータ	説明
run_info	プログラムに関する情報を書き込むための構造体。デフォルトでは、実行中のプログラムと一時停止している行に関する情報が含まれます。
info_requested	デフォルト (info_getStackDepth+ info_getLineInfo) 以外の情報を要求するためのオプションのビット・フィールド。0 (ゼロ) は、情報を要求しないことを意味します。 8-10 ページの「 情報フラグ 」を参照してください。

戻り値

表 8-10 SYNCHRONIZE ファンクションの戻り値

戻り値	説明
success	
error_timeout	プログラムが実行を開始する前にタイムアウトしました。
error_communication	その他の通信エラー。

SHOW_SOURCE プロシージャ

(実行されているプログラムの) ソース・コードを取得する最適な方法は、SQL を使用することです。たとえば、次のようになります。

```
DECLARE
    info DBMS_DEBUG.runtime_info;
BEGIN
    -- call DBMS_DEBUG.SYNCHRONIZE, CONTINUE,
    -- or GET_RUNTIME_INFO to fill in 'info'
    SELECT text INTO <buffer> FROM all_source
    WHERE owner = info.Program.Owner
          AND name = info.Program.Name
          AND line = info.Line#;
END;
```

ただし、このコードは非永続プログラム（無名ブロックやトリガー起動ブロックなど）では機能しません。非永続プログラムの場合は、SHOW_SOURCE をコールしてください。2 通りの方法があり、1 つはソース行の索引表を戻し、他の 1 つはバック（およびフォーマット）されたバッファを戻します。

SHOW_SOURCE プロシージャは 2 種類、オーバーロードされています。

構文

```
DBMS_DEBUG.SHOW_SOURCE (
    first_line IN BINARY_INTEGER,
    last_line  IN BINARY_INTEGER,
    source     OUT vc2_table);
```

パラメータ

表 8-11 SHOW_SOURCE プロシージャのパラメータ

パラメータ	説明
first_line	取り出す最初の行の行番号。（PL/SQL プログラムは、常に行 1 から始まり、途中の行が抜けることはありません。）
last_line	取り出す最後の行の行番号。プログラムの行数を超えると行は取り出されません。
source	結果の表。行番号で索引が設定されている場合があります。

戻り値

ソース行の索引表。ソース行は、first_line から格納されます。エラーが発生した場合は、空の表が戻されます。

使用上の注意

次の 2 番目の SHOW_SOURCE の書式は、フォーマット済バッファに行番号の付いたソースを戻します。索引表を使用するより処理は高速ですが、すべてのソースが取り出されるとは限りません。

ソースがバッファ長 (buflen) にすべて格納できなかった場合は、GET_MORE_SOURCE プロシージャを使用して追加のピースを取り出すことができます (pieces は、取り出す必要のある追加ピースの数を戻します)。

構文

```
DEMS_DEBUG.SHOW_SOURCE (  
    first_line  IN    BINARY_INTEGER,  
    last_line   IN    BINARY_INTEGER,  
    window      IN    BINARY_INTEGER,  
    print_arrow IN    BINARY_INTEGER,  
    buffer      IN OUT VARCHAR2,  
    buflen      IN    BINARY_INTEGER,  
    pieces      OUT    BINARY_INTEGER);
```

パラメータ

表 8-12 SHOW_SOURCE プロシージャのパラメータ

パラメータ	説明
first_line	印刷を開始する行番号。
last_line	印刷を終了する行番号。
window	行の ' 枠 ' (現行ソース行の概数)。
print_arrow	0 (ゼロ) 以外の場合は、現在の行の前に矢印が印刷されます。
buffer	ソース・リストを格納するバッファ。
buflen	バッファの長さ。
pieces	指定したバッファにすべてのソースを格納できない可能性がある場合は、0 (ゼロ) 以外に設定してください。

PRINT_BACKTRACE プロシージャ

このプロシージャは、現在の実行スタックのバックトレース・リストを印刷します。これは、プログラムが実行中の場合のみコールしてください。

PRINT_BACKTRACE プロシージャは 2 種類、オーバーロードされています。

構文

```
DBMS_DEBUG.PRINT_BACKTRACE (  
    listing IN OUT VARCHAR2);
```

パラメータ

表 8-13 PRINT_BACKTRACE プロシージャのパラメータ

パラメータ	説明
listing	埋込み改行付きのフォーマット済文字バッファ

構文

```
DBMS_DEBUG.PRINT_BACKTRACE (  
    backtrace OUT backtrace_table);
```

パラメータ

表 8-14 PRINT_BACKTRACE プロシージャのパラメータ

パラメータ	説明
backtrace	バックトレース・エントリの 1 を基準とした索引表。現在実行中のプロシージャは、表の最終エントリです（つまり、フレーム番号は、GET_VALUE が使用しているものと同一です）。エントリ 1 は、スタック上で最も古いプロシージャです。

CONTINUE ファンクション

このファンクションは、所定のブレイク・フラグ（イベントのマスク）をターゲット・プロセスのプロープに渡します。プロープに、ターゲット・プロセスの実行を継続するように通知し、ターゲット・プロセスが実行を終了するか、またはイベントを通知するまで待機します。

info_requested が NULL でない場合は、GET_RUNTIME_INFO をコールします。

構文

```
DBMS_DEBUG.CONTINUE (  
    run_info          IN OUT runtime_info,  
    breakflags        IN      BINARY_INTEGER,  
    info_requested IN      BINARY_INTEGER := NULL)  
RETURN BINARY_INTEGER;
```

パラメータ

表 8-15 CONTINUE ファンクションのパラメータ

パラメータ	説明
run_info	プログラムの状態に関する情報。
breakflags	イベントのマスク。8-9 ページの「 ブレイク・フラグ 」を参照してください。
info_requested	プログラムが停止したときに、run_info に戻される必要のある情報。8-10 ページの「 情報フラグ 」を参照してください。

戻り値

表 8-16 CONTINUE ファンクションの戻り値

戻り値	説明
success	
error_timeout	プログラムが実行を開始する前にタイムアウトしました。
error_communication	その他の通信エラー。

SET_BREAKPOINT ファンクション

このファンクションは、現行セッションを持続するためのブレイク・ポイントをプログラム・ユニットに設定します。ターゲット・プログラムがブレイク・ポイントに到達すると、実行は一時停止します。

構文

```
DBMS_DEBUG.SET_BREAKPOINT (  
    program      IN  program_info,  
    line#        IN  BINARY_INTEGER,  
    breakpoint#  OUT BINARY_INTEGER,  
    fuzzy        IN  BINARY_INTEGER := 0,  
    iterations   IN  BINARY_INTEGER := 0)  
RETURN BINARY_INTEGER;
```

パラメータ

表 8-17 SET_BREAKPOINT ファンクションのパラメータ

パラメータ	説明
program	ブレイク・ポイントが設定されるプログラム・ユニットに関する情報。(バージョン 2.1 以降では、名前領域、名前、所有者および DB リンクを NULL に設定でき、この場合のブレイク・ポイントは、現在実行中のプログラム・ユニットに設定されます。)
line#	ブレイク・ポイントが設定される行。
breakpoint#	正常に完了すると、ブレイク・ポイントを参照するための一意のブレイク・ポイント番号が含まれます。
fuzzy	指定した行に実行可能コードがない場合にのみ適用されます。 0 (ゼロ) の場合は、error_illegal_line が戻されます。 1 の場合は、ブレイク・ポイントを設定する行が指定行から順方向に検索されます。 -1 の場合は、ブレイク・ポイントを設定する行が指定した行から逆方向に検索されます。
iterations	このブレイク・ポイントを通知するまでの待機回数。

注意： fuzzy と iterations パラメータは、まだインプリメントされていません。

戻り値

表 8-18 SET_BREAKPOINT ファンクションの戻り値

戻り値	説明
success	
error_illegal_line	この行にブレーク・ポイントは設定できません。
error_bad_handle	該当するプログラム・ユニットが存在しません。

DELETE_BREAKPOINT ファンクション

このファンクションはブレーク・ポイントを削除します。

構文

```
DEMS_DEBUG.DELETE_BREAKPOINT (  
    breakpoint IN BINARY_INTEGER)  
RETURN BINARY_INTEGER;
```

パラメータ

表 8-19 DELETE_BREAKPOINT ファンクションのパラメータ

パラメータ	説明
breakpoint	以前の SET_BREAKPOINT コールから戻されたブレーク・ポイント番号

戻り値

表 8-20 DELETE_BREAKPOINT ファンクションの戻り値

戻り値	説明
success	
error_no_such_breakpt	該当するブレーク・ポイントが存在しません。
error_idle_breakpt	未使用のブレーク・ポイントは削除できません。

表 8-20 DELETE_BREAKPOINT ファンクションの戻り値

戻り値	説明
error_stale_breakpt	ブレーク・ポイントが設定された後にプログラム・ユニットが再定義されました。

DISABLE_BREAKPOINT ファンクション

このファンクションは、既存のブレーク・ポイントを非アクティブにしますが、削除しないでそのまま残します。

構文

```
DBMS_DEBUG.DISABLE_BREAKPOINT (  
    breakpoint IN BINARY_INTEGER)  
RETURN BINARY_INTEGER;
```

パラメータ

表 8-21 DISABLE_BREAKPOINT ファンクションのパラメータ

パラメータ	説明
breakpoint	以前の SET_BREAKPOINT コールから戻されたブレーク・ポイント番号

戻り値

表 8-22 DISABLE_BREAKPOINT ファンクションの戻り値

戻り値	説明
success	
error_no_such_breakpt	該当するブレーク・ポイントが存在しません。
error_idle_breakpt	未使用のブレーク・ポイントは使用禁止にできません。

ENABLE_BREAKPOINT ファンクション

このファンクションは、使用禁止の逆の処理を実行します。以前に使用禁止にしたブレーク・ポイントを使用可能にします。

構文

```
DBMS_DEBUG.ENABLE_BREAKPOINT (  
    breakpoint IN BINARY_INTEGER)  
RETURN BINARY_INTEGER;
```

パラメータ

表 8-23 ENABLE_BREAKPOINT ファンクションのパラメータ

パラメータ	説明
breakpoint	以前の SET_BREAKPOINT コールから戻されたブレーク・ポイント番号

戻り値

表 8-24 ENABLE_BREAKPOINT ファンクションの戻り値

戻り値	説明
success	
error_no_such_breakpt	該当するブレーク・ポイントが存在しません。
error_idle_breakpt	未使用のブレーク・ポイントは使用可能にできません。

SHOW_BREAKPOINTS プロシージャ

このプロシージャは、現在のブレーク・ポイントのリストを戻します。SHOW_BREAKPOINTS プロシージャは 2 種類、オーバーロードされています。

構文

```
DBMS_DEBUG.SHOW_BREAKPOINTS (  
    listing    IN OUT VARCHAR2);
```

パラメータ

表 8-25 SHOW_BREAKPOINTS プロシージャのパラメータ

パラメータ	説明
listing	ブレーク・ポイントのフォーマット済バッファ（改行を含む）

構文

```
DBMS_DEBUG.SHOW_BREAKPOINTS (  
    listing OUT breakpoint_table);
```

パラメータ

表 8-26 SHOW_BREAKPOINTS プロシージャのパラメータ

パラメータ	説明
listing	ブレーク・ポイント・エントリの索引表。ブレーク・ポイント番号は、表に対する索引で示されます。ブレーク・ポイント番号は 1 から始まり、削除されると再使用されます。

GET_VALUE ファンクション

このファンクションは、現在実行中のプログラムから値を取得します。GET_VALUE ファンクションは 2 種類、オーバーロードされています。

構文

```
DBMS_DEBUG.GET_VALUE (  
    variable_name IN VARCHAR2,  
    frame#        IN BINARY_INTEGER,  
    scalar_value  OUT VARCHAR2,  
    format        IN VARCHAR2 := NULL)  
RETURN BINARY_INTEGER;
```

パラメータ

表 8-27 GET_VALUE ファンクションのパラメータ

パラメータ	説明
variable_name	変数またはパラメータの名前。
frame#	値が存在するフレーム。0（ゼロ）の場合は現行プロシージャです。
scalar_value	値。
format	使用するオプションの日付書式（指定する必要がある場合）。

戻り値

表 8-28 GET_VALUE ファンクションの戻り値

戻り値	説明
success	
error_bogus_frame	フレームが存在しません。
error_no_debug_info	エントリポイントにデバッグ情報がありません。
error_no_such_object	variable_name が frame# に存在しません。
error_unknown_type	デバッグ情報内の型情報が判読不能です。
error_nullvalue	値が NULL です。
error_indexed_table	オブジェクトは表ですが、索引が提供されていません。

次の書式の GET_VALUE は、パッケージ変数取出し用です。フレーム番号のかわりに、変数を含んだパッケージを説明するハンドルを使用します。

構文

```
DBMS_DEBUG.GET_VALUE (  
    variable_name  IN  VARCHAR2,  
    handle         IN  program_info,  
    scalar_value   OUT VARCHAR2,  
    format         IN  VARCHAR2 := NULL)  
RETURN BINARY_INTEGER;
```


パラメータ

表 8-29 GET_VALUE ファンクションのパラメータ

パラメータ	説明
variable_name	変数またはパラメータの名前
handle	変数を含んだパッケージの説明
scalar_value	値
format	使用するオプションの日付書式（指定する必要がある場合）

戻り値

表 8-30 GET_VALUE ファンクションの戻り値

戻り値	説明
error_no_such_object	次のいずれかです。 - パッケージが存在しません。 - パッケージがインスタンス化されていません。 - ユーザーにパッケージをデバッグする権限がありません。 - オブジェクトがパッケージ内に存在しません。
error_indexed_table	オブジェクトは表ですが、索引が提供されていません。

例

この例は、スキーマ SCOTT 内の変数 VAR を含んだ任意のパッケージ PACK の値を取得する方法を示しています。

```
DECLARE
  handle      dbms_debug.program_info;
  resultbuf   VARCHAR2(500);
  retval      BINARY_INTEGER;
BEGIN
  handle.Owner      := 'SCOTT';
  handle.Name       := 'PACK';
  handle.namespace := dbms_debug.namespace_pkgspec_or_toplevel;
  retval            := dbms_debug.get_value('VAR', handle, resultbuf, NULL);
END;
```

SET_VALUE ファンクション

このファンクションは、現在実行中のプログラムに値を設定します。SET_VALUE ファンクションは 2 種類、オーバーロードされています。

構文

```
DBMS_DEBUG.SET_VALUE (  
    frame#                IN binary_integer,  
    assignment_statement IN varchar2)  
RETURN BINARY_INTEGER;
```

パラメータ

表 8-31 SET_VALUE ファンクションのパラメータ

パラメータ	説明
frame#	値を設定するフレーム。0（ゼロ）は現在実行中のフレームを意味します。
assignment_statement	値を設定するために実行する代入文（有効な PL/SQL である必要があります）。たとえば、'x := 3;' のように指定します。 このリリースでは、スカラー値のみサポートされています。代入文の右辺はスカラーである必要があります。

戻り値

表 8-32 SET_VALUE ファンクションの戻り値

戻り値	説明
success	
error_illegal_value	指定した値を設定することができません。
error_illegal_null	オブジェクト型は 'NOT NULL' で指定されているため、NULL は設定できません。
error_value_malformed	値がスカラーではありません。
error_name_incomplete	代入文をスカラーに変換できません。たとえば、'x := 3;'（x はレコード）のように指定します。

次の書式の SET_VALUE は、パッケージ変数の値を設定します。

構文

```
DBMS_DEBUG.SET_VALUE (  
    handle                IN program_info,  
    assignment_statement IN VARCHAR2)  
RETURN BINARY_INTEGER;
```

パラメータ

表 8-33 SET_VALUE ファンクションのパラメータ

パラメータ	説明
handle	変数を含んだパッケージの説明。
assignment_statement	値を設定するために実行する代入文（有効な PL/SQL である必要があります）。たとえば、'x := 3;' のように指定します。 このリリースでは、スカラー値のみサポートされています。代入文の右辺はスカラーである必要があります。

表 8-34 SET_VALUE ファンクションの戻り値

戻り値	説明
error_no_such_object	次のいずれかです。 - パッケージが存在しません。 - パッケージがインスタンス化されていません。 - ユーザーにパッケージをデバッグする権限がありません。 - オブジェクトがパッケージ内に存在しません。

PL/SQL コンパイラが一時ファイルを使用してパッケージ変数にアクセスし、プローブがこのような一時ファイルの更新を保証しない場合があります。ほとんど発生しませんが、SET_VALUE を使用したパッケージ変数の変更が及ばない行があります。

例

SCOTT.PACK.var の値を 6 に設定する方法は次のとおりです。

```
DECLARE
    handle  dbms_debug.program_info;
    retval  BINARY_INTEGER;
BEGIN
    handle.Owner      := 'SCOTT';
    handle.Name       := 'PACK';
    handle.namespace := dbms_debug.namespace_pkgspec_or_toplevel;
    retval           := dbms_debug.set_value(handle, 'var := 6;');
END;
```

DETACH_SESSION プロシージャ

このプロシージャは、ターゲット・プログラムのデバッグを停止します。このプロシージャはいつでもコール可能ですが、デバッグ・セッションの連結が解除されたことはターゲット・セッションに通知されず、ターゲット・セッションの実行は中断されません。したがって、ターゲット・セッションが独自にハングアップしないように注意してください。

構文

```
DBMS_DEBUG.DETACH_SESSION;
```

パラメータ

なし

GET_RUNTIME_INFO ファンクション

このファンクションは、現行プログラムに関する情報を戻します。これは、SYNCHRONIZE または CONTINUE の info_requested パラメータが 0 に設定された場合のみ必要です。

注意： このファンクションは、現在クライアント側の PL/SQL でのみ使用されます。

構文

```
DBMS_DEBUG.GET_RUNTIME_INFO (
    info_requested IN BINARY_INTEGER,
    run_info       OUT runtime_info)
RETURN BINARY_INTEGER;
```

パラメータ

表 8-35 GET_RUNTIME_INFO ファンクションのパラメータ

パラメータ	説明
info_requested	プログラムが停止したときに、run_infoに戻される必要のある情報。8-10 ページの「 情報フラグ 」を参照してください。
run_info	プログラムの状態に関する情報。

GET_INDEXES ファンクション

変数またはパラメータの名前を指定すると、索引表の場合はその一連の索引を戻します。索引表以外の場合はエラーが戻されます。

構文

```
DBMS_DEBUG.GET_INDEXES (  
    varname    IN  VARCHAR2,  
    frame#     IN  BINARY_INTEGER,  
    handle     IN  program_info,  
    entries    OUT index_table)  
RETURN BINARY_INTEGER;
```

パラメータ

表 8-36 GET_INDEXES ファンクションのパラメータ

パラメータ	説明
varname	索引情報を取得する変数の名前。
frame#	変数またはパラメータが常駐しているフレームの番号。パッケージ変数の場合は NULL です。
handle	パッケージの説明（オブジェクトがパッケージ変数の場合）。
entries	1 が基準の索引表。NULL 以外の場合は、entries(1) にその行の 1 番目の索引が含まれ、entries(2) に 2 番目の索引、以下同様に索引が含まれます。

戻り値

表 8-37 GET_INDEXES ファンクションの戻り値

戻り値	説明
error_no_such_object	次のいずれかです。 - パッケージが存在しません。 - パッケージがインスタンス化されていません。 - ユーザーにパッケージをデバッグする権限がありません。 - オブジェクトがパッケージ内に存在しません。

EXECUTE プロシージャ

このプロシージャは、ターゲット・セッションで SQL または PL/SQL コードを実行します。ターゲット・セッションは、ブレイク・ポイント（またはその他のイベント）で待機中であるとみなされます。デバッグ・セッションで DBMS_DEBUG.EXECUTE がコールされ、ターゲット・セッションにコードの実行を要求します。

構文

```
DBMS_DEBUG.EXECUTE (  
  what          IN VARCHAR2,  
  frame#        IN BINARY_INTEGER,  
  bind_results  IN BINARY_INTEGER,  
  results       IN OUT NOCOPY dbms_debug_vc2coll,  
  erm           IN OUT NOCOPY VARCHAR2);
```

パラメータ

表 8-38 EXECUTE プロシージャのパラメータ

パラメータ	説明
what	実行する SQL または PL/SQL のソース。
frame#	コードを実行するコンテキスト。現在は -1（グローバル・コンテキスト）のみサポートされています。
bind_results	ターゲット・セッションから値を戻すために、ソースを results に結合するかどうかを指定します。 0 = 結合しない。 1 = 結合する。

表 8-38 EXECUTE プロシージャのパラメータ

パラメータ	説明
results	結果を格納するコレクション (bind_results が 0 (ゼロ) 以外の場合)。
errm	エラーが発生した場合はエラー・メッセージ、それ以外の場合は NULL です。

例 1:

この例は SQL 文の実行例です。結果は戻されません。

```
DECLARE
    coll sys.dbms_debug_vc2coll; -- results (unused)
    errm VARCHAR2(100);
BEGIN
    dbms_debug.execute('insert into emp(ename,empno,deptno) ' ||
        'values(''LJE'', 1, 1)',
        -1, 0, coll, errm);
END;
```

例 2:

この例は PL/SQL ブロックの実行例で、結果は戻されません。ブロックは自立型トランザクションで、表に挿入された値はデバッグ・セッションで参照できます。

```
DECLARE
    coll sys.dbms_debug_vc2coll;
    errm VARCHAR2(100);
BEGIN
    dbms_debug.execute(
        'DECLARE PRAGMA autonomous_transaction; ' ||
        'BEGIN ' ||
        '    insert into emp(ename, empno, deptno) ' ||
        '    values(''LJE'', 1, 1); ' ||
        ' COMMIT; ' ||
        'END;',
        -1, 0, coll, errm);
END;
```

例 3:

この例は PL/SQL ブロックの実行例で、結果がいくつか戻されます。

```

DECLARE
    coll sys.dbms_debug_vc2coll;
    errm VARCHAR2(100);
BEGIN
    dbms_debug.execute(
        'DECLARE ' ||
        '    pp SYS.dbms_debug_vc2coll := SYS.dbms_debug_vc2coll(); ' ||
        '    x PLS_INTEGER; ' ||
        '    i PLS_INTEGER := 1; ' ||
        'BEGIN ' ||
        '    SELECT COUNT(*) INTO x FROM emp; ' ||
        '    pp.EXTEND(x * 6); ' ||
        '    FOR c IN (SELECT * FROM emp) LOOP ' ||
        '        pp(i) := 'Ename: ' || c.ename; i := i+1; ' ||
        '        pp(i) := 'Empno: ' || c.empno; i := i+1; ' ||
        '        pp(i) := 'Job: ' || c.job; i := i+1; ' ||
        '        pp(i) := 'Mgr: ' || c.mgr; i := i+1; ' ||
        '        pp(i) := 'Sal: ' || c.sal; i := i+1; ' ||
        '        pp(i) := null; i := i+1; ' ||
        '    END LOOP; ' ||
        '    :1 := pp; ' ||
        'END;',
        -1, 1, coll, errm);
    each := coll.FIRST;
    WHILE (each IS NOT NULL) LOOP
        dosomething(coll(each));
        each := coll.NEXT(each);
    END LOOP;
END;
```


PRINT_INSTANTIATIONS プロシージャ

このプロシージャは、現行のセッションでインスタンス化されたパッケージのリストを戻します。

パラメータ

- pkgs: インスタンス化されたパッケージ (OUT)
- flags: オプションのビットマスク
 - 1 - 仕様部の表示。
 - 2 - 本体の表示。
 - 4 - ローカル・インスタンスエーションの表示。
 - 8 - リモート・インスタンスエーションの表示 (まだインプリメントされていません)。
 - 16 - 高速ジョブの実行。デバッグ情報が存在しているか、またはライブラリ・ユニットがシュリンクラップされているかどうかはテストされません。

例外

no_target_program: ターゲット・セッションは現在実行されていません。

使用上の注意

"pkgs" の戻り値には、各インスタンスエーションの program_info が含まれます。有効なフィールドは、名前領域、名前、所有者およびライブラリ・ユニット型です。

また、Line# には次のビットマスクが含まれます。

- 1 - ライブラリ・ユニットにデバッグ情報が含まれています。
- 2 - ライブラリ・ユニットはシュリンクラップされています。

PROCEDURE print_instantiations (pkgs IN OUT NOCOPY backtrace_table, flags IN BINARY_INTEGER);

TARGET_PROGRAM_RUNNING プロシージャ

ターゲット・セッションが現在ストアド・プロシージャを実行中の場合は TRUE、実行していない場合は FALSE を戻します。

FUNCTION target_program_running RETURN BOOLEAN;

PING プロシージャ

ターゲット・セッションがタイムアウトしないように ping します。ターゲット・セッションで実行が中断したとき、ブレーク・ポイントなどでこのプロシージャを使用します。

timeout_behavior が retry_on_timeout に設定されている場合、このプロシージャは不要です。

例外

ターゲット・プログラムがない場合、またはターゲット・セッションがデバッグ・セッションからの入力を待機していない場合は、no_target_program 例外が表示されます。

PROCEDURE ping;

タイムアウト・オプション

ターゲット・セッションのタイムアウト・オプションは、set_timeout_behavior をコールすると、ターゲット・セッションとともに登録されます。

- `retry_on_timeout`: 再試行します。タイムアウトの効果はありません。timeout に無限に大きい値を設定した場合と同じです。
- `continue_on_timeout`: 同じイベント・フラグを使用して、実行を継続します。
- `nodebug_on_timeout`: debug-mode をオフにして（つまり、`debug_off` をコール）、実行を継続します。`debug_on` をコールして初期化し直さない限り、これ以降、このターゲット・セッションでイベントは生成されません。
- `abort_on_timeout`: `abort_execution` フラグを使用して実行を継続します。プログラムが即時に異常終了します。セッションはデバッグ・モードのままです。

`retry_on_timeout` CONSTANT BINARY_INTEGER:= 0;

`continue_on_timeout` CONSTANT BINARY_INTEGER:= 1;

`nodebug_on_timeout` CONSTANT BINARY_INTEGER:= 2;

`abort_on_timeout` CONSTANT BINARY_INTEGER:= 3;

SET_TIMEOUT_BEHAVIOUR プロシージャ

このプロシージャは、タイムアウト発生時のターゲット・セッションの処理方法をプロンプトに通知します。（このコールは、ターゲット・セッションで行われます。）

パラメータ

- `behavior`: 次のいずれかです（前述の説明を参照してください）。

`retry_on_timeout`

`continue_on_timeout`

```
nodebug_on_timeout
```

```
abort_on_timeout
```

例外

unimplemented: 要求された動作が認識されていません。

使用上の注意

デフォルトの動作（このプロシージャがコールされない場合）は、`continue_on_timeout` です。この `continue_on_timeout` は、デバッガ・クライアントが（次のイベントで）制御の再確立ができるので、ターゲット・セッションが無期限にハングアップすることはありません。

PROCEDURE `set_timeout_behavior` (`behavior` IN `PLS_INTEGER`);

GET_TIMEOUT_BEHAVIOUR: 現行のタイムアウト動作を戻します。（このコールは、ターゲット・セッションで実行されます。）

FUNCTION `get_timeout_behavior` RETURN `BINARY_INTEGER`;

OER ブレーク・ポイント

PL/SQL プログラム内で宣言される例外は、ユーザー定義の例外として認識されています。この他に、Oracle カーネルから戻される Oracle エラー（OER）があります。この2つのメカニズムを結合するために、PL/SQL はユーザー定義の例外を OER に変換する "exception_init" プラグマを提供しています。この処理には PL/SQL ハンドラが使用され、PL/SQL エンジンには、OER を Oracle カーネルに戻すことができます。現行のリリースでは、OER に関する使用可能な情報はその番号のみです。2つのユーザー定義例外が同じ OER に `exception_init` されると、区別できません。

情報フラグ

```
info_getOerInfo    CONSTANT PLS_INTEGER:= 32;
```

理由

```
reason_oer_breakpoint  CONSTANT BINARY_INTEGER:= 26;
```

RUNTIME_INFO

Runtime_info は、実行プログラムに関するコンテキスト情報を提供します。

プローブ v2.4 では OER が追加されています。info_getOerInfo が設定されている場合は、設定を取得します。OER は正数です。1403 を 100 に、6510 を 1 に変換し、その他の値を無効にすると、SQLCODE に変換できます。

```
TYPE runtime_info IS RECORD
(
    Line#           BINARY_INTEGER, -- (duplicate of program.line#)
    Terminated     BINARY_INTEGER, -- has the program terminated?
    Breakpoint       BINARY_INTEGER, -- breakpoint number
    StackDepth       BINARY_INTEGER, -- number of frames on the stack
    InterpreterDepth BINARY_INTEGER, -- <reserved field>
    Reason           BINARY_INTEGER, -- reason for suspension
    Program          program_info,   -- source location
    -- Following fields were added in Probe v2.4
    oer              PLS_INTEGER     -- OER (exception), if any
);
```

OER_TABLE

show_breakpoints によって使用されます。

```
TYPE oer_table IS TABLE OF BINARY_INTEGER INDEX BY BINARY_INTEGER;
```

SET_OER_BREAKPOINT

OER にブレーク・ポイントを設定します。コード・ブレーク・ポイント同様、ブレーク・ポイントはセッションに対して（または削除されるまで）持続されます。

パラメータ

oer: OER (4 バイトの正数)

戻り値

SUCCESS

使用上の注意

OER ブレーク・ポイントでサポートされている機能性は、コード・ブレーク・ポイントと比較すると少なくなります。特に、次の点に注意してください。

- ブレーク・ポイント番号は戻されません。かわりに OER の番号が使用されます。したがって、指定した OER に複数のブレーク・ポイントを設定することはできません（操作できません）。

- OER ブレーク・ポイントは使用禁止にできません（ただし、クライアントは、これを削除して自由にシミュレートできます）。
- OER ブレーク・ポイントは、delete_oer_breakpoint を介して削除されます。

FUNCTION set_oer_breakpoint(oer IN PLS_INTEGER) RETURN PLS_INTEGER;

DELETE_OER_BREAKPOINT

OER ブレーク・ポイントを削除します。

パラメータ

oer: 削除する OER（4 バイトの正数）

戻り値

success

error_no_such_breakpt: 該当する OER ブレーク・ポイントが存在しません。

FUNCTION delete_oer_breakpoint(oer IN PLS_INTEGER) RETURN PLS_INTEGER;

SHOW_BREAKPOINTS

パラメータ

- code_breakpoints: ブレーク・ポイント番号で索引付けされた、ブレーク・ポイント・エントリの索引表
- oer_breakpoints: OER で索引付けされた、OER ブレーク・ポイントの索引表

PROCEDURE show_breakpoints (code_breakpoints OUT breakpoint_table, oer_breakpoints OUT oer_table);

DBMS_DEFER

DBMS_DEFER は、レプリケート・トランザクションの遅延リモート・プロシージャ・コール (RPC) 機能へのユーザー・インタフェースです。レプリケート・アプリケーションは、このインタフェース内のコールを使用して、後でトランザクションをリモート・ノードで実行するためにプロシージャ・コールをキューに登録します。

このルーチンは通常、AFTER 行トリガーまたはアプリケーションで指定した更新プロシージャからコールされます。

DBMS_DEFER パッケージ

サブプログラムの要約

表 9-1 DBMS_DEFER パッケージのサブプログラム

サブプログラム	説明
9-3 ページの CALL プロシージャ	リモート・プロシージャに対する遅延コールを作成します。
9-4 ページの COMMIT_WORK プロシージャ	遅延リモート・プロシージャ・コール（RPC）が適切な構成かどうかをチェックし、その後トランザクション・コミットを実行します。
9-5 ページの datatype_ARG プロシージャ	遅延リモート・プロシージャ・コール（RPC）に渡されるデータを提供します。
9-7 ページの TRANSACTION プロシージャ	新規遅延トランザクションの開始を指示します。

CALL プロシージャ

このプロシージャは、リモート・プロシージャに対する遅延コールを作成します。

構文

```
DBMS_DEFER.CALL (
  schema_name      IN   VARCHAR2,
  package_name     IN   VARCHAR2,
  proc_name        IN   VARCHAR2,
  arg_count        IN   NATURAL,
  { nodes          IN   node_list_t
  | group_name     IN   VARCHAR2 := '' });
```

注意： このプロシージャはオーバーロードされています。nodes パラメータと group_name パラメータは、両方同時には指定できません。

パラメータ

表 9-2 CALL プロシージャのパラメータ

パラメータ	説明
schema_name	ストアド・プロシージャが置かれているスキーマ名。
package_name	ストアド・プロシージャを含んでいるパッケージの名前。ストアド・プロシージャはパッケージの一部であることが必要です。スタンドアロン・プロシージャに対する遅延コールは、サポートされていません。
proc_name	コールを延期するリモート・プロシージャの名前。
arg_count	プロシージャのパラメータ数。これらの各パラメータに対しては、DBMS_DEFER.datatype_ARG のコールが 1 つずつ必要です。 注意： 一部のパラメータにデフォルトが設定されている場合でも、プロシージャに対してすべての引数を指定する必要があります。
nodes	遅延コールを伝播する先の完全修飾データベース名の PL/SQL 表。この表は、位置 1 から始まり、NULL エントリが検出されるか、または NO_DATA_FOUND 例外が発生するまで索引付けされています。表内のデータは、大文字と小文字が区別されません。この引数はオプションです。
group_name	内部的に使用されるパラメータです。

例外

表 9-3 CALL プロシージャの例外

例外	説明
ORA-23304 (malformedcall)	前のコールが正しく構成されませんでした。
ORA-23319	パラメータ値が不適正です。
ORA-23352	(nodes または前の DBMS_DEFER.TRANSACTION コールで指定された) 宛先リストの値が重複しています。

COMMIT_WORK プロシージャ

このプロシージャは、遅延リモート・プロシージャ・コール (RPC) が適切な構成かどうかをチェックし、その後トランザクション・コミットを実行します。

構文

```
DBMS_DEFER.COMMIT_WORK (  
    commit_work_comment IN VARCHAR2);
```

パラメータ

表 9-4 COMMIT_WORK プロシージャのパラメータ

パラメータ	説明
commit_work_ comment	SQL の COMMIT COMMENT 文と同じです。

例外

表 9-5 COMMIT_WORK プロシージャの例外

例外	説明
ORA-23304 (malformedcall)	トランザクションが正しく発行されなかったか、または終了しませんでした。

***datatype_ARG* プロシージャ**

このプロシージャは、遅延リモート・プロシージャ・コール (RPC) に渡されるデータを提供します。プロシージャに渡す必要があるデータの型によって、プロシージャの各引数ごとに次のプロシージャの 1 つをコールする必要があります。

プロシージャの各パラメータは、DBMS_DEFER.CALL の実行後に、*datatype_ARG* プロシージャを使用して指定してください。つまり、遅延リモート・プロシージャ・コール (RPC) にデフォルトのパラメータは使用できません。たとえば、次のプロシージャがあると想定します。

```
CREATE OR REPLACE PACKAGE my_pack AS
    PROCEDURE my_proc(a VARCHAR2, b VARCHAR2 DEFAULT 'SALES');
END;
/
```

DBMS_DEFER.CALL プロシージャを実行するときは、MY_PROC プロシージャの各パラメータ用に別の行を挿入する必要があります。

```
CREATE OR REPLACE PROCEDURE load_def_tx IS
    node DBMS_DEFER.NODE_LIST_T;
BEGIN
    node(1) := 'MYCOMPUTER.WORLD';
    node(2) := NULL;
    DBMS_DEFER.TRANSACTION(node);
    DBMS_DEFER.CALL('SCOTT', 'MY_PACK', 'MY_PROC', 2);
    DBMS_DEFER.VARCHAR2_ARG('TEST');
    DBMS_DEFER.VARCHAR2_ARG('SALES'); -- required, cannot omit to use default
END;
/
```

構文

```
DBMS_DEFER.NUMBER_ARG      (arg IN NUMBER);
DBMS_DEFER.DATE_ARG        (arg IN DATE);
DBMS_DEFER.VARCHAR2_ARG    (arg IN VARCHAR2);
DBMS_DEFER.CHAR_ARG        (arg IN CHAR);
DBMS_DEFER.ROWID_ARG       (arg IN ROWID);
DBMS_DEFER.RAW_ARG         (arg IN RAW);
DBMS_DEFER.BLOB_ARG        (arg IN BLOB);
DBMS_DEFER.CLOB_ARG        (arg IN CLOB);
DBMS_DEFER.NCLOB_ARG       (arg IN NCLOB);
DBMS_DEFER.NCHAR_ARG       (arg IN NCHAR);
DBMS_DEFER.NVARCHAR2_ARG   (arg IN NVARCHAR2);
DBMS_DEFER.ANY_CLOB_ARG    (arg IN CLOB);
DBMS_DEFER.ANY_VARCHAR2_ARG (arg IN VARCHAR2);
DBMS_DEFER.ANY_CHAR_ARG    (arg IN CHAR);
```

パラメータ

表 9-6 datatype_ARG プロシージャのパラメータ

パラメータ	説明
arg	事前に遅延コールを実行したリモート・プロシージャに渡すパラメータの値

例外

表 9-7 datatype_ARG プロシージャの例外

例外	説明
ORA-23323	引数の値が長すぎます。

TRANSACTION プロシージャ

このプロシージャは、新規遅延トランザクションの開始を指示します。このコールを省略すると、DBMS_DEFER.CALL の最初のコールが新規トランザクションの開始とみなされます。

構文

```
DBMS_DEFER.TRANSACTION (  
    nodes IN    node_list_t);
```

注意： このプロシージャはオーバーロードされています。入力パラメータが指定されていない場合も指定されている場合と同様に動作しますが、指定されていない場合は、nodes パラメータのノードを使用するかわりに、DEFDEFAULTDEST ビューの nodes が使用されます。

パラメータ

表 9-8 TRANSACTION プロシージャのパラメータ

パラメータ	説明
nodes	トランザクションの遅延コール伝播先の完全修飾データベース名の PL/SQL 表。この表は、位置 1 から始まり、NULL エントリが検出されるか、または NO_DATA_FOUND 例外が発生するまで索引付けされています。表内のデータは、大文字と小文字が区別されません。

例外

表 9-9 TRANSACTION プロシージャの例外

例外	説明
ORA-23304 (malformedcall)	前のトランザクションが正しく発行されなかったか、または終了していませんでした。
ORA-23319	パラメータ値が不適正です。
ORA-23352	ノード・リストの値が重複している場合に DBMS_DEFER.CALL で生成されます。

10

DBMS_DEFER_QUERY

DBMS_DEFER_QUERY によって、ビューでは参照できない遅延 RPC のキュー・データを問い合わせることができます。

DBMS_DEFER_QUERY パッケージ

サブプログラムの要約

表 10-1 DBMS_DEFER_QUERY パッケージのサブプログラム

サブプログラム	説明
10-3 ページの GET_ARG_FORM ファンクション	遅延コールの引数の形式を判別します。
10-4 ページの GET_ARG_TYPE ファンクション	遅延コールの引数の型を判別します。
10-6 ページの GET_CALL_ARGS プロシージャ	指定されたコールに対する様々な引数をテキストで戻します。
10-7 ページの GET_datatype_ARG ファンクション	遅延コールの引数の値を判別します。

GET_ARG_FORM ファンクション

このファンクションは、遅延コールの引数の形式を判別します。また、遅延コール・パラメータのキャラクタ・セット ID を戻します。

関連項目： Replication Manager における遅延トランザクションとエラー・トランザクションの表示方法は、Replication Manager のオンライン・ヘルプを参照してください。

構文

```
DBMS_DEFER_QUERY.GET_ARG_FORM (  
    callno             IN    NUMBER,  
    arg_no             IN    NUMBER,  
    deferred_tran_id   IN    VARCHAR2)  
RETURN NUMBER;
```

パラメータ

表 10-2 GET_ARG_FORM ファンクションのパラメータ

パラメータ	説明
callno	DEFCALL ビューからのコール識別子。
arg_no	コール引数リスト上の目的のパラメータの位置。パラメータの位置は、コール内のパラメータを 1 から順に数えます。
deferred_tran_id	遅延トランザクション ID。

例外

表 10-3 GET_ARG_FORM ファンクションの例外

例外	説明
NO_DATA_FOUND	入力パラメータが遅延コールのパラメータに対応していません。

戻り値

表 10-4 GET_ARG_Form ファンクションの戻り値

戻り値	対応するデータ型
1	CHAR、VARCHAR2、CLOB
2	NCHAR、NVARCHAR2、NCLOB

GET_ARG_TYPE ファンクション

このファンクションは、遅延コールの引数の型を判別します。遅延リモート・プロシージャ・コール（RPC）のパラメータの型が戻されます。

関連項目： Replication Manager における遅延トランザクションとエラー・トランザクションの表示方法は、Replication Manager のオンライン・ヘルプを参照してください。

構文

```
DBMS_DEFER_QUERY.GET_ARG_TYPE (  
    callno           IN    NUMBER,  
    arg_no           IN    NUMBER,  
    deferred_tran_id IN    VARCHAR2)  
RETURN NUMBER;
```

パラメータ

表 10-5 GET_ARG_TYPE ファンクションのパラメータ

パラメータ	説明
callno	遅延リモート・プロシージャ・コール（RPC）の DEFCALL ビューでの ID 番号。
arg_no	判別する型のコールに対する引数の数値的な位置。プロシージャに対する最初の引数の位置は 1 です。
deferred_tran_id	遅延トランザクションの ID。

例外

表 10-6 GET_ARG_TYPE ファンクションの例外

例外	説明
NO_DATA_FOUND	入力パラメータが遅延コールのパラメータに対応していません。

戻り値

表 10-7 GET_ARG_TYPE ファンクションの戻り値

戻り値	対応するデータ型
1	VARCHAR2
2	NUMBER
11	ROWID
12	DATE
23	RAW
96	CHAR
112	CLOB
113	BLOB

GET_CALL_ARGS プロシージャ

このプロシージャは、指定されたコールに対する様々な引数をテキストで戻します。テキストは最初の 2000 バイトまでに制限されています。

構文

```
DBMS_DEFER_QUERY.GET_CALL_ARGS (  
    callno      IN  NUMBER,  
    startarg    IN  NUMBER := 1,  
    argcnt      IN  NUMBER,  
    argsize     IN  NUMBER,  
    tran_id     IN  VARCHAR2,  
    date_fmt    IN  VARCHAR2,  
    types       OUT TYPE_ARG,  
    forms       OUT TYPE_ARG,  
    vals        OUT VAL_ARG);
```

パラメータ

表 10-8 GET_CALL_ARGS プロシージャのパラメータ

パラメータ	説明
callno	遅延 RPC の DEFCALL ビューでの ID 番号
startarg	記述する最初の引数の数値的な位置
argcnt	コールにある引数の数
argsize	戻される引数の最大サイズ
tran_id	遅延トランザクションの ID
date_fmt	日付を戻す書式
types	引数の型を含んでいる配列
forms	キャラクタ・セット形式の引数を含んでいる配列
vals	テキスト形式で引数の値を含んでいる配列

例外

表 10-9 GET_CALL_ARGS プロシージャの例外

例外	説明
NO_DATA_FOUND	入力パラメータが遅延コールのパラメータに対応していません。

GET_datatype_ARG ファンクション

このファンクションは、遅延コールの引数の値を判別します。

関連項目： Replication Manager における遅延トランザクションとエラー・トランザクションの表示方法は、Replication Manager のオンライン・ヘルプを参照してください。

構文

取り出す引数値の型に応じて、該当するファンクションの構文は次のように異なります。各ファンクションは、それぞれ指定された引数の値を戻します。

```
DBMS_DEFER_QUERY.GET_datatype_ARG (  
    callno           IN    NUMBER,  
    arg_no           IN    NUMBER,  
    deferred_tran_id IN    VARCHAR2 DEFAULT NULL)  
RETURN datatype;
```

datatype には次のいずれかを指定します。

```
{ NUMBER  
| VARCHAR2  
| CHAR  
| DATE  
| RAW  
| ROWID  
| BLOB  
| CLOB  
| NCLOB  
| NCHAR  
| NVARCHAR2 }
```

パラメータ

表 10-10 GET_datatype_ARG ファンクションのパラメータ

パラメータ	説明
callno	遅延リモート・プロシージャ・コール（RPC）の DEFCALL ビューでの ID 番号。
arg_no	判別する値のコールに対する引数の数値的な位置。プロシージャに対する最初の引数の位置は 1 です。
deferred_tran_id	遅延トランザクションの ID。GET_ARG_TYPE に渡された最後のトランザクション ID にデフォルト設定されます。デフォルトは NULL です。

例外

表 10-11 GET_datatype_ARG ファンクションの例外

例外	説明
NO_DATA_FOUND	入力パラメータが遅延コールのパラメータに対応していません。
ORA-26564	この位置の引数は指定された型ではありません。

DBMS_DEFER_SYS

DBMS_DEFER_SYS プロシージャは、デフォルトのレプリケーション・ノード・リストを管理します。

このパッケージは、レプリケート・トランザクションの遅延リモート・プロシージャ・コール機能へのシステム管理者用インタフェースです。管理者とレプリケーション・デーモンは、この機能を使用して、リモート・ノードのキューに入れられたトランザクションを実行できます。また、管理者はリモート・コールの宛先のノードを制御できます。

DBMS_DEFER_SYS パッケージ

サブプログラムの要約

表 11-1 DBMS_DEFER_SYS パッケージのサブプログラム

サブプログラム	説明
11-4 ページの ADD_DEFAULT_DEST プロシージャ	DEFDEFAULTDEST ビューに宛先データベースを追加します。
11-4 ページの DELETE_DEFAULT_DEST プロシージャ	DEFDEFAULTDEST ビューから宛先データベースを削除します。
11-5 ページの DELETE_DEF_DESTINATION プロシージャ	DEFSCHEDULE ビューから宛先データベースを削除します。
11-5 ページの DELETE_ERROR プロシージャ	DEFERROR ビューからトランザクションを削除します。
11-6 ページの DELETE_TRAN プロシージャ	DEFTRANDEST ビューからトランザクションを削除します。
11-7 ページの DISABLED ファンクション	現行サイトから指定サイトへの遅延トランザクション・キューの伝播が使用可能かどうかを判断します。
11-8 ページの EXCLUDE_PUSH プロシージャ	遅延トランザクションの PUSH を防止する排他ロックを取得します。
11-9 ページの EXECUTE_ERROR プロシージャ	トランザクションの元の受信者のセキュリティ・コンテキスト内で正常に完了しなかった遅延トランザクションを再実行します。
11-10 ページの EXECUTE_ERROR_AS_USER プロシージャ	このプロシージャを実行しているユーザーのセキュリティ・コンテキスト内で正常に完了しなかった遅延トランザクションを再実行します。
11-11 ページの PURGE ファンクション	現行マスター・サイトまたはスナップショット・サイトの遅延トランザクション・キューから、送信済トランザクションをバージします。
11-13 ページの PUSH ファンクション	現行マスター・サイトまたはスナップショット・サイトの遅延リモート・プロシージャ・コール (RPC) のキューを、別のマスター・サイトに強制的に送信します。
11-16 ページの REGISTER_PROPAGATOR プロシージャ	指定したユーザーをローカル・データベースのプロパゲータとして登録します。

表 11-1 DBMS_DEFER_SYS パッケージのサブプログラム

サブプログラム	説明
11-17 ページの SCHEDULE_PURGE プロシージャ	現行マスター・サイトまたはスナップショット・サイトの遅延トランザクション・キューから、送信済トランザクションをバージするジョブをスケジュールします。
11-19 ページの SCHEDULE_PUSH プロシージャ	遅延トランザクション・キューをリモート・マスター宛先に送信するジョブをスケジュールします。
11-21 ページの SET_DISABLED プロシージャ	現行サイトから指定宛先サイトへの遅延トランザクション・キューの伝播を使用禁止または使用可能にします。
11-22 ページの UNREGISTER_PROPAGATOR プロシージャ	ローカル・データベースからプロパゲータとしてのユーザーの登録を解除します。
11-23 ページの UNSCHEDULE_PURGE プロシージャ	スナップショット・サイトまたはマスター・サイトの遅延トランザクション・キューからの送信済トランザクションの自動バージを停止します。
11-23 ページの UNSCHEDULE_PUSH プロシージャ	スナップショット・サイトまたはマスター・サイトから別のマスター・サイトへの遅延トランザクション・キューの自動送信を停止します。

ADD_DEFAULT_DEST プロシージャ

このプロシージャは、DEFDEFAULTDEST ビューに宛先データベースを追加します。

構文

```
DBMS_DEFER_SYS.ADD_DEFAULT_DEST (  
    dblink    IN    VARCHAR2);
```

パラメータ

表 11-2 ADD_DEFAULT_DEST プロシージャのパラメータ

パラメータ	説明
dblink	DEFDEFAULTDEST ビューに追加するノードの完全修飾データベース名

例外

表 11-3 ADD_DEFAULT_DEST プロシージャの例外

例外	説明
ORA-23352	指定した dblink は、デフォルト・リストにすでに存在します。

DELETE_DEFAULT_DEST プロシージャ

このプロシージャは、DEFDEFAULTDEST ビューから宛先データベースを削除します。

構文

```
DBMS_DEFER_SYS.DELETE_DEFAULT_DEST (  
    dblink    IN    VARCHAR2);
```

パラメータ

表 11-4 DELETE_DEFAULT_DEST プロシージャのパラメータ

パラメータ	説明
dblink	DEFDEFAULTDEST ビューから削除するノードの完全修飾データベース名。ビューにこの DB リンクが見つからなかった場合は、何の処理も行われません。

DELETE_DEF_DESTINATION プロシージャ

このプロシージャは、DEFSCHEDULE ビューから宛先データベースを削除します。

構文

```
DBMS_DEFER_SYS.DELETE_DEF_DESTINATION (  
    destination    IN    VARCHAR2,  
    force          IN    BOOLEAN := FALSE);
```

パラメータ

表 11-5 DELETE_DEF_DESTINATION プロシージャのパラメータ

パラメータ	説明
destination	DEFSCHEDULE ビューから削除する宛先の完全修飾データベース名。 ビューにこの宛先が見つからなかった場合は、何の処理も行われません。
force	TRUE に設定すると、安全チェックはすべて無視され、宛先が削除されます。

DELETE_ERROR プロシージャ

このプロシージャは、DEFERROR ビューからトランザクションを削除します。

構文

```
DBMS_DEFER_SYS.DELETE_ERROR (  
    deferred_tran_id    IN    VARCHAR2,  
    destination         IN    VARCHAR2);
```

パラメータ

表 11-6 DELETE_ERROR プロシージャのパラメータ

パラメータ	説明
deferred_tran_id	DEFERROR ビューから削除する遅延トランザクションの DEFERROR ビューでの ID 番号。このパラメータが NULL の場合は、他のパラメータの要件と一致するすべてのトランザクションが削除されます。
destination	トランザクションが最初にキューに入れられたデータベースの DEFERROR ビューでの完全修飾データベース名。このパラメータが NULL の場合は、他のパラメータの要件と一致するすべてのトランザクションが DEFERROR ビューから削除されます。

DELETE_TRAN プロシージャ

このプロシージャは、DEFTRANDEST ビューからトランザクションを削除します。トランザクションに対する DEFTRANDEST または DEFERROR のエントリが他にない場合、そのトランザクションは、DEFTRAN と DEFCALL のビューからも削除されます。

構文

```
DBMS_DEFER_SYS.DELETE_TRAN (  
    deferred_tran_id      IN   VARCHAR2,  
    destination           IN   VARCHAR2);
```

パラメータ

表 11-7 DELETE_TRAN プロシージャのパラメータ

パラメータ	説明
deferred_tran_id	削除する遅延トランザクションの DEFTRAN ビューでの ID 番号。このパラメータが NULL の場合は、他のパラメータの要件と一致するすべてのトランザクションが削除されます。
destination	トランザクションが最初にキューに入れられたデータベースの DEFTRANDEST ビューでの完全修飾データベース名。このパラメータが NULL の場合は、他のパラメータの要件と一致するすべてのトランザクションが削除されます。

DISABLED ファンクション

このファンクションは、現行サイトから指定サイトへの遅延トランザクション・キューの伝播が使用可能かどうかを判断します。指定した宛先に対する遅延リモート・プロシージャ・コール（RPC）のキューが使用禁止の場合、DISABLED ファンクションは TRUE を戻します。

構文

```
DBMS_DEFER_SYS.DISABLED (  
    destination IN VARCHAR2)  
    RETURN BOOLEAN;
```

パラメータ

表 11-8 DISABLED ファンクションのパラメータ

パラメータ	説明
destination	伝播状態をチェックするノードの完全修飾データベース名

戻り値

表 11-9 DISABLED ファンクションの戻り値

戻り値	説明
TRUE	現行サイトから指定したサイトへの伝播は使用禁止です。
FALSE	現行サイトから指定したサイトへの伝播は使用可能です。

例外

表 11-10 DISABLED ファンクションの例外

例外	説明
NO_DATA_FOUND	指定した destination が DEFSCHEDULE ビューにありません。

EXCLUDE_PUSH プロシージャ

このファンクションは、遅延トランザクションの PUSH（シリアルまたはパラレル）を防止する排他ロックを取得します。ロックの取得時にコミットが実行されます。ロックは `RELEASE_ON_COMMIT => TRUE` で取得されるため、遅延トランザクション・キューの送信は、次のコミットの後に再開できます。

構文

```
DBMS_DEFER_SYS.EXCLUDE_PUSH (  
    timeout IN INTEGER)  
RETURN INTEGER;
```

パラメータ

表 11-11 EXCLUDE_PUSH ファンクションのパラメータ

パラメータ	説明
timeout	タイムアウト（秒数）。この時間内にロックを取得できない場合（エラーまたは PUSH が現在進行中であるため）は、値 1 が戻されます。タイムアウト値が DBMS_LOCK.MAXWAIT の場合は、無期限に待機します。

戻り値

表 11-12 EXCLUDE_PUSH ファンクションの戻り値

戻り値	説明
0	正常終了。ロックが取得されました。
1	タイムアウト。ロックは取得されていません。
2	デッドロック。ロックは取得されていません。
4	ロックはすでに取得されています。

EXECUTE_ERROR プロシージャ

このプロシージャは、トランザクションの元の受信者のセキュリティ・コンテキスト内で正常に完了しなかった遅延トランザクションを再実行します。

構文

```
DBMS_DEFER_SYS.EXECUTE_ERROR (
    deferred_tran_id IN    VARCHAR2,
    destination      IN    VARCHAR2);
```

パラメータ

表 11-13 EXECUTE_ERROR プロシージャのパラメータ

パラメータ	説明
deferred_tran_id	再実行する遅延トランザクションの DEFERROR ビューでの ID 番号。NULL の場合は、destination のキューにあるすべてのトランザクションが再実行されます。
destination	トランザクションが最初にキューに入れられたデータベースの DEFERROR ビューでの完全修飾データベース名。NULL 以外の値を設定ください。指定したデータベース名が完全に修飾されていないか、無効な場合、エラーは発生しません。

例外

表 11-14 EXECUTE_ERROR プロシージャの例外

例外	説明
ORA-24275	NULL と NULL 以外のパラメータを不正に組み合わせて使用しました。
badparam	パラメータが未指定または無効です（たとえば、destination が NULL の場合）。
missinguser	無効なユーザーです。

EXECUTE_ERROR_AS_USER プロシージャ

このプロシージャは、正常に完了しなかった遅延トランザクションを再実行します。各トランザクションは接続ユーザーのセキュリティ・コンテキスト内で実行されます。

構文

```
DBMS_DEFER_SYS.EXECUTE_ERROR_AS_USER (  
    deferred_tran_id IN   VARCHAR2,  
    destination      IN   VARCHAR2);
```

パラメータ

表 11-15 EXECUTE_ERROR_AS_USER プロシージャのパラメータ

パラメータ	説明
deferred_tran_id	再実行する遅延トランザクションの DEFERROR ビューでの ID 番号。 NULL の場合は、destination のキューにあるすべてのトランザク ションが再実行されます。
destination	トランザクションが最初にキューに入れられたデータベースの DEFERROR ビューでの完全修飾データベース名。NULL 以外の値を設 定ください。

例外

表 11-16 EXECUTE_ERROR_AS_USER プロシージャの例外

例外	説明
ORA-24275	NULL と NULL 以外のパラメータを不正に組み合わせて使用しました。
badparam	パラメータが未指定または無効です（たとえば、destination が NULL の場合）。
missinguser	無効なユーザーです。

PURGE ファンクション

このファンクションは、現行マスター・サイトまたはスナップショット・サイトの遅延トランザクション・キューから、送信済トランザクションをパージします。

構文

```

DBMS_DEFER_SYS.PURGE (
    purge_method      IN  BINARY_INTEGER := purge_method_quick,
    rollback_segment  IN  VARCHAR2       := NULL,
    startup_seconds   IN  BINARY_INTEGER := 0,
    execution_seconds IN  BINARY_INTEGER := seconds_infinity,
    delay_seconds     IN  BINARY_INTEGER := 0,
    transaction_count IN  BINARY_INTEGER := transactions_infinity,
    write_trace       IN  BOOLEAN        := NULL);
RETURN BINARY_INTEGER;

```

パラメータ

表 11-17 PURGE ファンクションのパラメータ

パラメータ	説明
purge_method	遅延トランザクション・キューのパージ方法を制御します。purge_method_quick を指定するとコストを削減でき、purge_method_precise を指定すると精度が高くなります。 purge_method_quick を使用すると、正常に送信された遅延トランザクションと遅延プロシージャ・コールを、パージ前に予測した時間より長い間、DEFTRAN データ・ディクショナリ・ビューと DEFCALL データ・ディクショナリ・ビューに残る場合があります。詳細は、11-12 ページの「 使用上の注意 」を参照してください。
rollback_segment	パージに使用するロールバック・セグメント名。デフォルトは NULL です。
startup_seconds	同じ遅延トランザクション・キューの直前のパージを待つ最大秒数。
execution_seconds	>0 の場合は、指定した秒数後にパージを即時停止します。
delay_seconds	遅延トランザクション・キューにパージするトランザクションが delay_seconds 秒間ない場合は、パージを停止します。
transaction_count	>0 の場合は、transaction_count が示す件数のトランザクションをパージした後、シャットダウンします。
write_trace	TRUE に設定すると、PURGE ファンクションによって戻された結果値がサーバーのトレース・ファイルに記録されます。

戻り値

表 11-18 Purge ファンクションの戻り値

戻り値	説明
0	OK。delay_seconds 秒経過後に終了しました。
1	起動中にロック・タイムアウトで終了しました。
2	execution_seconds 秒経過したため終了しました。
3	transaction_count 件を超えたため終了しました。
5	エラー発生後に終了しました。

例外

表 11-19 PURGE ファンクションの例外

例外	説明
argoutofrange	パラメータ値が有効範囲外です。
executiondisabled	ページの実行が使用禁止です。
defererror	内部エラーです。

使用上の注意

DBMS_DEFER_SYS.PURGE ファンクションの purge_method パラメータに purge_method_quick を使用すると、遅延トランザクションと遅延プロシージャ・コールは、正
常に送信された後、DEFCALL データ・ディクショナリ・ビューと DEFTRAN データ・ディク
ショナリ・ビューに残る場合があります。この動作は、複数のデータベース・リンクを持ち、
送信が 1 つのデータベース・リンクに対してのみ実行されるレプリケーション環境で発生し
ます。

遅延トランザクションと遅延プロシージャ・コールをパージするには、次のいずれかの処理を実行してください。

- `purge_method` パラメータに、`purge_method_quick` ではなく、`purge_method_precise` パラメータを使用します。`purge_method_precise` を使用すると、コストがかかりますが、遅延トランザクションと遅延プロシージャ・コールは、正常に送信された後、確実にパージされます。
- `purge_method` パラメータに `purge_method_quick` を使用すると、遅延トランザクションがすべてのデータベース・リンクに送信されます。遅延トランザクションと遅延プロシージャ・コールは、最後のデータベース・リンクへの送信が正常に完了するとパージされます。

PUSH ファンクション

このファンクションは、現行マスター・サイトまたはスナップショット・サイトの遅延リモート・プロシージャ・コール（RPC）のキューを、シリアルまたはパラレル伝播を使用して別のマスター・サイトに強制的に送信（伝播）します。

構文

```
DBMS_DEFER_SYS.PUSH (
    destination          IN  VARCHAR2,
    parallelism          IN  BINARY_INTEGER := 0,
    heap_size            IN  BINARY_INTEGER := 0,
    stop_on_error        IN  BOOLEAN       := FALSE,
    write_trace          IN  BOOLEAN       := FALSE,
    startup_seconds      IN  BINARY_INTEGER := 0,
    execution_seconds    IN  BINARY_INTEGER := seconds_infinity,
    delay_seconds        IN  BINARY_INTEGER := 0,
    transaction_count    IN  BINARY_INTEGER := transactions_infinity,
    delivery_order_limit IN  NUMBER         := delivery_order_infinity)
RETURN BINARY_INTEGER;
```

パラメータ

表 11-20 PUSH ファンクションのパラメータ

パラメータ	説明
destination	変更内容の転送先マスターの完全修飾データベース名。
parallelism	0 はシリアル伝播を、 $n > 1$ は n 個の平行・サーバー・プロセスを使用する平行伝播を、1 は 1 つの平行・サーバー・プロセスのみ使用する平行伝播をそれぞれ指定します。
heap_size	平行伝播スケジューリングで同時に検査されるトランザクションの最大数。最適なパフォーマンスのためのデフォルト設定は Oracle が自動的に計算します。オラクル社カスタマ・サポート・センターから指示がない限り、このパラメータは設定しないでください。
stop_on_error	デフォルトは FALSE で、競合などのエラーが発生しても処理は継続されます。TRUE を設定すると、宛先サイトでトランザクションにエラーが発生したことが最初に通知されたときに（可能な場合は正常に）シャットダウンします。
write_trace	TRUE に設定すると、ファンクションによって戻された結果値がサーバーのトレース・ファイルに記録されます。
startup_seconds	同じ宛先への直前の送信を待つ最大秒数。
execution_seconds	>0 の場合は、指定した秒数後に送信を停止します。transaction_count と execution_seconds が 0（デフォルト）の場合は、キュー内のトランザクションがなくなるまで実行されます。 execution_seconds パラメータは、操作を開始できる時間を制御するのみです。トランザクションがリモート・サイトで必要とする時間は含まれていません。したがって、execution_seconds パラメータは、リモート・サイトへのトランザクションの伝播を停止するための正確な制御に使用することを目的としていません。正確な制御が必要な場合は、transaction_count パラメータまたは delivery_order パラメータを使用してください。
delay_seconds	キューが空の場合でも、指定した秒数が経過するまで戻しません。PUSH がタイトなループからコールされる場合は、実行オーバーヘッドの削減に役立ちます。
transaction_count	>0 の場合は、停止までに送信されるトランザクションの最大数。transaction_count と execution_seconds が 0（デフォルト）の場合は、送信する必要があるトランザクションがキュー内になくなるまで実行されます。
delivery_order_limit	delivery_order >= delivery_order_limit の場合は、トランザクションを送信する前に実行を正確に停止します。

戻り値

PUSH
表 11-21 PUSH ファンクションの戻り値

戻り値	説明
0	OK。delay_seconds 秒経過後に終了しました。
1	起動中にロック・タイムアウトで終了しました。
2	execution_seconds 秒経過したため終了しました。
3	transaction_count 件を超えたため終了しました。
4	delivery_order_limit 数を超えたため終了しました。
5	エラー発生後に終了しました。

例外

表 11-22 PUSH ファンクションの例外

例外	説明
deferror	シリアル伝播を実行するときは、パラレル伝播が完全にシャットダウンされている必要があります。
incompleteparallelpush	
executiondisabled	宛先での遅延 RPC の実行が使用禁止です。
crt_err_err	DEFERROR のエントリ作成時にエラーが発生しました。
deferred_rpc_qiesce	オブジェクト・グループのレプリケーション・アクティビティが中断されています。
commfailure	遅延 RPC 中に通信エラーが発生しました。
missingpropator	プロパゲータが存在しません。

REGISTER_PROPAGATOR プロシージャ

このプロシージャは、指定したユーザーをローカル・データベースのプロパゲータとして登録します。また、そのユーザーに CREATE SESSION、CREATE PROCEDURE、CREATE DATABASE LINK および EXECUTE ANY PROCEDURE 権限を付与します（権限を付与されたユーザーはラッパーを作成できます）。

構文

```
DBMS_DEFER_SYS.REGISTER_PROPAGATOR (  
    username IN VARCHAR2);
```

パラメータ

表 11-23 REGISTER_PROPAGATOR プロシージャのパラメータ

パラメータ	説明
username	ユーザー名

例外

表 11-24 REGISTER_PROPAGATOR プロシージャの例外

例外	説明
missinguser	指定したユーザーが存在しません。
alreadypropagator	指定したユーザーはすでにプロパゲータです。
duplicatepropagator	別のプロパゲータがすでに存在します。

SCHEDULE_PURGE プロシージャ

このプロシージャは、現行マスター・サイトまたはスナップショット・サイトの遅延トランザクション・キューから、送信済トランザクションをパージするジョブをスケジュールします。スケジュールするパージ・ジョブは1つにしてください。

構文

```
DBMS_DEFER_SYS.SCHEDULE_PURGE (  
    interval          IN  VARCHAR2,  
    next_date         IN  DATE,  
    reset             IN  BOOLEAN          := NULL,  
    purge_method      IN  BINARY_INTEGER := NULL,  
    rollback_segment  IN  VARCHAR2        := NULL,  
    startup_seconds   IN  BINARY_INTEGER := NULL,  
    execution_seconds IN  BINARY_INTEGER := NULL,  
    delay_seconds     IN  BINARY_INTEGER := NULL,  
    transaction_count IN  BINARY_INTEGER := NULL,  
    write_trace       IN  BOOLEAN          := NULL);
```

パラメータ

表 11-25 SCHEDULE_PURGE プロシージャのパラメータ

パラメータ	説明
interval	ページの次回実行時間を計算するファンクションを提供します。この値は DEFSCHEDULE ビューの interval フィールドに格納され、このビューの next_date フィールドが計算されます。このパラメータのデフォルト値 NULL を使用すると、このフィールドの値は変更されません。フィールドに前の値が設定されていない場合は、NULL で作成されます。このフィールドに値を指定しない場合は、next_date の値を指定する必要があります。
next_date	サイトのキューから送信済トランザクションをページする時間を指定できます。この値は DEFSCHEDULE ビューの next_date フィールドに格納されます。このパラメータのデフォルト値 NULL を使用すると、このフィールドの値は変更されません。このフィールドに前の値が設定されていない場合は、NULL で作成されます。このフィールドに値を指定しない場合は、interval の値を指定する必要があります。
reset	TRUE に設定すると、LAST_TXN_COUNT、LAST_ERROR と LAST_MSG の値が NULL にリセットされます。
purge_method	遅延トランザクション・キューのページ方法を制御します。purge_method_quick を指定するとコストを削減でき、purge_method_precise を指定すると精度が高くなります。
rollback_segment	ページに使用するロールバック・セグメント名。デフォルトは NULL です。
startup_seconds	同じ遅延トランザクション・キューの直前のページを待つ最大秒数。
execution_seconds	>0 の場合は、指定した秒数後にページを即時停止します。
delay_seconds	遅延トランザクション・キューにページするトランザクションが delay_seconds 秒間ない場合は、ページを停止します。
transaction_count	>0 の場合は、transaction_count が示す件数のトランザクションをページした後、シャットダウンします。
write_trace	TRUE に設定すると、PURGE ファンクションによって戻された結果値がサーバーのトレース・ファイルに記録されます。

SCHEDULE_PUSH プロシージャ

このプロシージャは、遅延トランザクション・キューをリモート・マスター宛先に送信するジョブをスケジュールします。このプロシージャは COMMIT を実行します。

構文

```

DBMS_DEFER_SYS.SCHEDULE_PUSH (
    destination      IN  VARCHAR2,
    interval         IN  VARCHAR2,
    next_date        IN  DATE,
    reset            IN  BOOLEAN          := FALSE,
    parallelism      IN  BINARY_INTEGER := NULL,
    heap_size        IN  BINARY_INTEGER := NULL,
    stop_on_error    IN  BOOLEAN          := NULL,
    write_trace      IN  BOOLEAN          := NULL,
    startup_seconds  IN  BINARY_INTEGER := NULL,
    execution_seconds IN  BINARY_INTEGER := NULL,
    delay_seconds    IN  BINARY_INTEGER := NULL,
    transaction_count IN  BINARY_INTEGER := NULL);

```

パラメータ

表 11-26 SCHEDULE_PUSH プロシージャのパラメータ

パラメータ	説明
destination	変更内容の転送先マスターの完全修飾データベース名。
interval	送信の次回実行時間を計算するファンクションを提供します。この値は DEFSCHEDULE ビューの interval フィールドに格納され、このビューの next_date フィールドが計算されます。このパラメータのデフォルト値 NULL を使用すると、このフィールドの値は変更されません。フィールドに前の値が設定されていない場合は、NULL で作成されます。このフィールドに値を指定しない場合は、next_date の値を指定する必要があります。
next_date	遅延トランザクションをマスター・サイト宛先に送信する時間を指定できます。この値は DEFSCHEDULE ビューの next_date フィールドに格納されます。このパラメータのデフォルト値 NULL を使用すると、このフィールドの値は変更されません。フィールドに前の値が設定されていない場合は、NULL で作成されます。このフィールドに値を指定しない場合は、interval の値を指定する必要があります。
reset	TRUE に設定すると、LAST_TXN_COUNT、LST_ERROR および LAST_MSG の値が NULL にリセットされます。

表 11-26 SCHEDULE_PUSH プロシージャのパラメータ

パラメータ	説明
parallelism	0 はシリアル伝播を、 $n > 1$ は n 個の平行・サーバー・プロセスを使用する平行伝播を、1 は 1 つの平行・サーバー・プロセスのみ使用する平行伝播をそれぞれ指定します。
heap_size	平行伝播スケジューリングで同時に検査されるトランザクションの最大数。最適なパフォーマンスのためのデフォルト設定は Oracle が自動的に計算します。オラクル社カスタマ・サポート・センターから指示がない限り、このパラメータは設定しないでください。
stop_on_error	デフォルトは FALSE で、競合などのエラーが発生しても処理は継続されます。TRUE を設定すると、宛先サイトでトランザクションにエラーが発生したことが最初に通知されたときに（可能な場合は正常に）シャットダウンします。
write_trace	TRUE に設定すると、ファンクションによって戻された結果値がサーバーのトレース・ファイルに記録されます。
startup_seconds	同じ宛先への直前の送信を待つ最大秒数。
execution_seconds	> 0 の場合は、指定した秒数後に実行を停止します。transaction_count と execution_seconds が 0（デフォルト）の場合は、キュー内のトランザクションがなくなるまで実行されます。
delay_seconds	キューが空の場合でも、指定した秒数が経過するまで戻しません。PUSH がタイトなループからコールされる場合は、実行オーバーヘッドの削減に役立ちます。
transaction_count	> 0 の場合は、停止までに送信されるトランザクションの最大数。transaction_count と execution_seconds が 0（デフォルト）の場合は、送信する必要があるトランザクションがキュー内になくなるまで実行されます。

SET_DISABLED プロシージャ

このプロシージャは、現行のサイトから指定した宛先サイトへの遅延トランザクション・キューの伝播を使用禁止または使用可能にします。disabled パラメータが TRUE の場合は、指定した宛先への伝播が使用禁止になり、その後に PUSH を起動しても、遅延リモート・プロシージャ・コール (RPC) のキューは送信されません。結果的に、SET_DISABLED は、指定した宛先にキューをすでに送信しているセッションには効果がありますが、DBMS_DEFER でキューを追加しているセッションには効果がありません。

disabled パラメータが FALSE の場合は、指定した宛先への伝播は使用可能になります。この処理でキューは送信されませんが、その後に PUSH を起動すると指定した宛先にキューを送信できます。disabled パラメータが TRUE か FALSE に関係なく、他のセッションで効果を発揮するには、設定に COMMIT が必要です。

構文

```
DBMS_DEFER_SYS.SET_DISABLED (  
    destination    IN    VARCHAR2,  
    disabled       IN    BOOLEAN := TRUE);
```

パラメータ

表 11-27 SET_DISABLED プロシージャのパラメータ

パラメータ	説明
destination	伝播状態を変更するノードの完全修飾データベース名。
disabled	デフォルトでは、このパラメータは現行のサイトから指定した宛先への遅延トランザクション・キューの伝播を使用禁止にします。伝播を使用可能にするには、FALSE に設定してください。

例外

表 11-28 SET_DISABLED プロシージャの例外

例外	説明
NO_DATA_FOUND	DEFSCHEDULE ビューに、指定した destination のエントリが見つかりません。

UNREGISTER_PROPAGATOR プロシージャ

このプロシージャは、ローカル・データベースからプロパゲータとしてのユーザーの登録を解除します。このプロシージャは次の処理を実行します。

- 指定したプロパゲータを DEFPROPAGATOR から削除します。
- REGISTER_PROPAGATOR で付与した権限を、指定したユーザーから取り消します (個々に付与された同様の権限も含まれます)。
- 指定したプロパゲータのスキーマで生成されたラッパーを削除し、レプリケーション・カタログに削除のマークを設定します。

構文

```
DBMS_DEFER_SYS.UNREGISTER_PROPAGATOR (  
    username IN VARCHAR2  
    timeout   IN  INTEGER DEFAULT DBMS_LOCK.MAXWAIT);
```

パラメータ

表 11-29 UNREGISTER_PROPAGATOR プロシージャのパラメータ

パラメータ	説明
username	プロパゲータのユーザー名。
timeout	タイムアウト (秒数)。プロパゲータが使用中の場合は、タイムアウトまで待機します。デフォルトは DBMS_LOCK.MAXWAIT です。

例外

表 11-30 UNREGISTER_PROPAGATOR プロシージャの例外

パラメータ	説明
missingpropagator	指定したユーザーはプロパゲータではありません。
propagator_inuse	プロパゲータが使用中であるため、登録を解除できません。後で再試行してください。

UNSCCHEDULE_PURGE プロシージャ

このプロシージャは、スナップショット・サイトまたはマスター・サイトの遅延トランザクション・キューからの送信済トランザクションの自動パージを停止します。

構文

```
DBMS_DEFER_SYS.UNSCHEDULE_PURGE();
```

パラメータ

なし

UNSCCHEDULE_PUSH プロシージャ

このプロシージャは、スナップショット・サイトまたはマスター・サイトから別のマスター・サイトへの遅延トランザクション・キューの自動送信を停止します。

構文

```
DBMS_DEFER_SYS.UNSCHEDULE_PUSH (
    dblink IN VARCHAR2);
```

パラメータ

表 11-31 UNSCHEDULE_PUSH プロシージャのパラメータ

パラメータ	説明
dblink	遅延リモート・プロシージャ・コールの定期的な実行のスケジュールを解除するマスター・データベース・サイトへの完全修飾パス名。

表 11-32 UNSCHEDULE_PUSH プロシージャの例外

例外	説明
NO_DATA_FOUND	DEFSCHEDULE ビューに、指定した dblink のエントリが見つかりません。

DBMS_DESCRIBE

DBMS_DESCRIBE パッケージによって、PL/SQL オブジェクトに関する情報を取得できます。オブジェクト名を指定すると、DBMS_DESCRIBE は結果を含む索引表のセットを戻します。名前変換が完全に実行され、目的のオブジェクトに対するセキュリティ・チェックも行われます。

このパッケージは、Oracle コール・インタフェースの OCIDescribeAny コールと同じ機能を提供します。

関連項目：『Oracle8i コール・インタフェース・プログラマーズ・ガイド』

セキュリティ

このパッケージは PUBLIC で使用でき、記述されているスキーマ・オブジェクトに基づいた独自のセキュリティ・チェックが実行されます。

型

DBMS_DESCRIBE パッケージは、2 つの型の PL/SQL 表を宣言します。この表は、DESCRIBE_PROCEDURE が戻すデータを、その OUT パラメータに格納するために使用されます。2 つの型は次のとおりです。

```
TYPE VARCHAR2_TABLE IS TABLE OF VARCHAR2(30)
    INDEX BY BINARY_INTEGER;

TYPE NUMBER_TABLE IS TABLE OF NUMBER
    INDEX BY BINARY_INTEGER;
```

エラー

DBMS_DESCRIBE では、ORA-20000 ～ ORA-20004 までの範囲のアプリケーション・エラーが発生する可能性があります。

表 12-1 DBMS_DESCRIBE のエラー

エラー	説明
ORA-20000	ORU-10035: cannot describe a package ('X') only a procedure within a package.
ORA-20001	ORU-10032: procedure 'X' within package 'Y' does not exist.
ORA-20002	ORU-10033: object 'X' is remote, cannot describe; expanded name 'Y'.
ORA-20003	ORU-10036: object 'X' is invalid and cannot be described.
ORA-20004	Syntax error attempting to parse 'X'

サブプログラムの要約

DBMS_DESCRIBE に含まれるプロシージャは DESCRIBE_PROCEDURE のみです。

DESCRIBE_PROCEDURE プロシージャ

プロシージャ DESCRIBE_PROCEDURE は、ストアド・プロシージャ名、プロシージャの記述およびその各パラメータを受け入れます。

構文

```
DBMS_DESCRIBE.DESCRIBE_PROCEDURE(  
    object_name      IN  VARCHAR2,  
    reserved1        IN  VARCHAR2,  
    reserved2        IN  VARCHAR2,  
    overload         OUT NUMBER_TABLE,  
    position         OUT NUMBER_TABLE,  
    level            OUT NUMBER_TABLE,  
    argument_name    OUT VARCHAR2_TABLE,  
    datatype         OUT NUMBER_TABLE,  
    default_value    OUT NUMBER_TABLE,  
    in_out           OUT NUMBER_TABLE,  
    length           OUT NUMBER_TABLE,  
    precision        OUT NUMBER_TABLE,  
    scale            OUT NUMBER_TABLE,  
    radix            OUT NUMBER_TABLE,  
    spare            OUT NUMBER_TABLE);
```

パラメータ

表 12-2 DBMS_DESCRIBE.DESCRIBE_PROCEDURE のパラメータ

パラメータ	説明
object_name	<p>記述するプロシージャの名前。</p> <p>このパラメータの構文は、SQL の識別子に使用されている規則に従っています。名前はシノニムでもかまいません。このパラメータは必須であり、NULL は指定できません。名前の長さは合計で 197 バイトまでです。OBJECT_NAME が正しく指定されないと、次の例外のいずれかが発生する場合があります。</p> <p>ORA-20000: パッケージが指定されました。指定できるのは、ストアド・プロシージャ、ストアド・ファンクション、パッケージ・プロシージャまたはパッケージ・ファンクションのみです。</p> <p>ORA-20001: 指定したプロシージャまたはファンクションは、所定のパッケージ内に存在しません。</p> <p>ORA-20002: 指定したオブジェクトはリモート・オブジェクトです。このプロシージャは、現在リモート・オブジェクトを記述できません。</p> <p>ORA-20003: 指定したオブジェクトは無効であるため記述できません。</p> <p>ORA-20004: オブジェクトの指定に構文エラーがあります。</p>
reserved1 reserved2	<p>将来使用される予定です。-- NULL または空文字列を設定してください。</p>
overload	<p>プロシージャの署名に割り当てられた一意の番号。</p> <p>プロシージャがオーバーロードされている場合、このフィールドには、プロシージャのバージョンごとに異なる値が格納されます。</p>
position	<p>パラメータ・リスト内の引数の位置。</p> <p>位置 0 (ゼロ) には、ファンクションの戻り型の値が戻されます。</p>
level	<p>引数がレコードなどのコンポジット型の場合は、そのデータ型のレベルが戻されます。</p> <p>ODESSP コールの例の詳細は、『Oracle8i コール・インタフェース・プログラマーズ・ガイド』を参照してください。</p>
argument_name	<p>記述するプロシージャに関連付けられた引数の名前。</p>

表 12-2 DBMS_DESCRIBE.DESCRIBE_PROCEDURE のパラメータ

パラメータ	説明
datatype	<p>記述する引数の Oracle データ型。</p> <p>データ型とその数値型のコードは、次のとおりです。</p> <ul style="list-style-type: none"> 0 placeholder for procedures with no arguments 1 VARCHAR, VARCHAR, STRING 2 NUMBER, INTEGER, SMALLINT, REAL, FLOAT, DECIMAL 3 BINARY_INTEGER, PLS_INTEGER, POSITIVE, NATURAL 8 LONG 11 ROWID 12 DATE 23 RAW 24 LONG RAW 96 CHAR (ANSI FIXED CHAR), CHARACTER 106 MLSLABEL 250 PL/SQL RECORD 251 PL/SQL TABLE 252 PL/SQL BOOLEAN
default_value	<p>記述される引数にデフォルト値がある場合は 1、デフォルト値がない場合は 0（ゼロ）です。</p>
in_out	<p>パラメータのモードを記述します。値は次のとおりです。</p> <ul style="list-style-type: none"> 0 IN 1 OUT 2 IN OUT
length	<p>記述する引数のデータ長（バイト）。</p>
precision	<p>記述する引数のデータ型が 2（NUMBER）の場合、このパラメータはその数値の精度を表します。</p>
scale	<p>記述する引数のデータ型が 2（NUMBER など）の場合、このパラメータはその数値の位取りを表します。</p>
radix	<p>記述する引数のデータ型が 2（NUMBER など）の場合、このパラメータはその数値の基数を表します。</p>
spare	<p>将来使用するために予約されています。</p>

戻り値

DESCRIBE_PROCEDURE からの戻り値はすべて、その OUT パラメータに戻されます。このデータ型は、パラメータの変数値に適応するために、PL/SQL の表です。

例

DESCRIBE_PROCEDURE プロシージャは、外部サービス・インタフェースとしても使用できます。

たとえば、OBJECT_NAME に SCOTT.ACCOUNT_UPDATE を指定するクライアントを想定します。ACCOUNT_UPDATE は次の仕様を持つオーバーロードされたファンクションです。

```
table account (account_no number, person_id number,
               balance number(7,2))
table person  (person_id number(4), person_nm varchar2(10))

function ACCOUNT_UPDATE (account_no  number,
                        person        person%rowtype,
                        amounts       dbms_describe.number_table,
                        trans_date    date)
return accounts.balance%type;

function ACCOUNT_UPDATE (account_no  number,
                        person        person%rowtype,
                        amounts       dbms_describe.number_table,
                        trans_no      number)
return accounts.balance%type;
```

このプロシージャの記述は、次のように出力されます。

overload	position	argument	level	datatype	length	prec	scale	rad
-----	-----	-----	-----	-----	-----	-----	-----	---
1	0		0	2	22	7	2	10
1	1	ACCOUNT	0	2	0	0	0	0
1	2	PERSON	0	250	0	0	0	0
1	1	PERSON_ID	1	2	22	4	0	10
1	2	PERSON_NM	1	1	10	0	0	0
1	3	AMOUNTS	0	251	0	0	0	0
1	1		1	2	22	0	0	0
1	4	TRANS_DATE	0	12	0	0	0	0
2	0		0	2	22	7	2	10
2	1	ACCOUNT_NO	0	2	22	0	0	0
2	2	PERSON	0	2	22	4	0	10
2	3	AMOUNTS	0	251	22	4	0	10
2	1		1	2	0	0	0	0
2	4	TRANS_NO	0	2	0	0	0	0

次の PL/SQL プロシージャには、そのパラメータとしてすべての PL/SQL データ型があります。

```
CREATE OR REPLACE PROCEDURE p1 (  
    pvc2    IN    VARCHAR2,  
    pvc     OUT   VARCHAR,  
    pstr    IN OUT STRING,  
    plong   IN    LONG,  
    prowid  IN    ROWID,  
    pchara  IN    CHARACTER,  
    pchar   IN    CHAR,  
    praw    IN    RAW,  
    plraw   IN    LONG RAW,  
    pbinint IN    BINARY_INTEGER,  
    pplsint IN    PLS_INTEGER,  
    pbool   IN    BOOLEAN,  
    pnat    IN    NATURAL,  
    ppos    IN    POSITIVE,  
    pposn   IN    POSITIVEN,  
    pnatn   IN    NATURALN,  
    pnum    IN    NUMBER,  
    pintgr  IN    INTEGER,  
    pint    IN    INT,  
    psmall  IN    SMALLINT,  
    pdec    IN    DECIMAL,  
    preal   IN    REAL,  
    pfloat  IN    FLOAT,  
    pnumer  IN    NUMERIC,  
    pdp     IN    DOUBLE PRECISION,  
    pdate   IN    DATE,  
    pmls    IN    MLSLABEL) AS  
  
BEGIN  
    NULL;  
END;
```

このプロシージャを次のパッケージを使用して記述とします。

```
CREATE OR REPLACE PACKAGE describe_it AS  
  
    PROCEDURE desc_proc (name VARCHAR2);  
  
END describe_it;  
  
CREATE OR REPLACE PACKAGE BODY describe_it AS  
  
    PROCEDURE prt_value(val VARCHAR2, isize INTEGER) IS  
        n INTEGER;
```

```
BEGIN
    n := isize - LENGTHB(val);
    IF n < 0 THEN
        n := 0;
    END IF;
    DBMS_OUTPUT.PUT(val);
    FOR i in 1..n LOOP
        DBMS_OUTPUT.PUT(' ');
    END LOOP;
END prt_value;

PROCEDURE desc_proc (name VARCHAR2) IS

    overload      DBMS_DESCRIBE.NUMBER_TABLE;
    position      DBMS_DESCRIBE.NUMBER_TABLE;
    c_level       DBMS_DESCRIBE.NUMBER_TABLE;
    arg_name      DBMS_DESCRIBE.VARCHAR2_TABLE;
    dtype        DBMS_DESCRIBE.NUMBER_TABLE;
    def_val       DBMS_DESCRIBE.NUMBER_TABLE;
    p_mode        DBMS_DESCRIBE.NUMBER_TABLE;
    length        DBMS_DESCRIBE.NUMBER_TABLE;
    precision     DBMS_DESCRIBE.NUMBER_TABLE;
    scale         DBMS_DESCRIBE.NUMBER_TABLE;
    radix         DBMS_DESCRIBE.NUMBER_TABLE;
    spare         DBMS_DESCRIBE.NUMBER_TABLE;
    idx          INTEGER := 0;

BEGIN
    DBMS_DESCRIBE.DESCRIBE_PROCEDURE(
        name,
        null,
        null,
        overload,
        position,
        c_level,
        arg_name,
        dtype,
        def_val,
        p_mode,
        length,
        precision,
        scale,
        radix,
        spare);
```

```

DBMS_OUTPUT.PUT_LINE('Position      Name          DTY  Mode');
LOOP
    idx := idx + 1;
    prt_value(TO_CHAR(position(idx)), 12);
    prt_value(arg_name(idx), 12);
    prt_value(TO_CHAR(dty(idx)), 5);
    prt_value(TO_CHAR(p_mode(idx)), 5);
    DBMS_OUTPUT.NEW_LINE;
END LOOP;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.NEW_LINE;
        DBMS_OUTPUT.NEW_LINE;

END desc_proc;
END describe_it;

```

結果は次のようになり、PL/SQL データ型のすべての数値コードがリストされます。

Position	Name	Datatype_Code	Mode
1	PVC2	1	0
2	PVC	1	1
3	PSTR	1	2
4	PLONG	8	0
5	PROWID	11	0
6	PCHARA	96	0
7	PCHAR	96	0
8	PRAW	23	0
9	PLRAW	24	0
10	PBININT	3	0
11	PPLSINT	3	0
12	PBOOL	252	0
13	PNAT	3	0
14	PPOS	3	0
15	PPOSN	3	0
16	PNATN	3	0
17	PNUM	2	0
18	PINTGR	2	0
19	PINT	2	0
20	PSMALL	2	0
21	PDEC	2	0
22	PREAL	2	0
23	PFLOAT	2	0
24	PNUMER	2	0
25	PDP	2	0
26	PDATE	12	0
27	PMLS	106	0

使用上の注意

現在、第三世代言語で型が `record` または `boolean` の引数を直接バインドすることはできません。ブール値の場合は、次の方法があります。

- ファンクション `F` がブールを戻すと想定します。 `G` は、1 つの IN ブール引数を持つプロシージャで、 `H` は 1 つの OUT ブール引数を持つプロシージャです。この場合、これらのファンクションを `DTYINT`（自然数）にバインドして実行できます。ここでは、`0=>FALSE` で、`1=>TRUE` です。

```
begin :dtyint_bind_var := to_number(f); end;

begin g(to_boolean(:dtyint_bind_var)); end;

declare b boolean; begin h(b); if b then :dtyint_bind_var := 1;
else :dtyint_bind_var := 0; end if; end;
```

- 型が `record` の引数を使用するプロシージャにアクセスするためには、前述の最後の例（ファンクション `H` を参照）のラッパーと同様のラッパーを記述する必要があります。

DBMS_DISTRIBUTED_TRUST_ADMIN

DBMS_DISTRIBUTED_TRUST_ADMIN プロシージャは、Trusted Servers リストをメンテナンスします。サーバーが信頼されているかどうかを定義するには、これらのプロシージャを使用します。データベースが信頼されていない場合は、カレント・ユーザーのデータベース・リンクはそのデータベースから拒否されます。

注意： Oracle は、企業の LDAP ディレクトリ・サービスに格納されているドメイン・メンバーシップ・リストとともにローカルの Trusted Servers リストを使用して、別のデータベースが信頼されているかどうかを判断します。LDAP ディレクトリ・サービスのエントリは、OEM の Enterprise Security Manager Tool で管理されます。Oracle は、次の場合に別のデータベースが信頼されているとみなします。

- 1) ローカル・データベースとして、ディレクトリ・サービスの同じ企業ドメイン内にある場合
- 2) その企業ドメインに、ディレクトリ・サービスで信頼されていることを示すマークが付けられている場合
- 3) ローカルの Trusted Servers リストに信頼されていないデータベースとしてリストされていない場合

カレント・ユーザーのデータベース・リンクが別のデータベースから受け入れられるのは、関係する両方のデータベースが互いに信頼している場合のみです。

ディレクトリ・サービスにリストされているかどうかに関係なく、データベース・サーバーを Trusted Servers リストにローカルでリストすることができます。ただし、同じドメイン内にローカル・データベースとして存在しないデータベースをリストする場合、またはそのドメインが信頼されていない場合、そのエントリは無効になります。

この機能性は、Oracle Advanced Security オプションの Enterprise User Security 機能の一部です。

要件

DBMS_DISTRIBUTED_TRUST_ADMIN を実行するには、EXECUTE_CATALOG_ROLE ロールが DBA に付与されている必要があります。ビュー TRUSTED_SERVERS を検索するには、SELECT_CATALOG_ROLE ロールが DBA に付与されている必要があります。

すべてのサーバーについて、信頼されているかどうかを認識することが重要です。データベースがすべてのデータベースをすでに信頼している場合、またはそのデータベースがすでに信頼されている場合は、ALLOW_SERVER プロシージャで特定のサーバーを信頼しても効果はありません。同様に、そのデータベースがすべてのデータベースをすでに信頼していない場合、またはそのデータベースがすでに信頼されていない場合は、DENY_SERVER プロシージャで特定のサーバーを拒否しても効果はありません。

プロシージャ DENY_ALL と ALLOW_ALL は、それぞれ ALLOW_SERVER プロシージャまたは DENY_SERVER プロシージャで明示的に許可または拒否されたすべてのエントリ（サーバー名）を削除します。

サブプログラムの要約

表 13-1 DBMS_DISTRIBUTED_TRUST_ADMIN パッケージのサブプログラム

サブプログラム	説明
13-2 ページの ALLOW_ALL プロシージャ	リストを空にし、すべてのサーバーを信頼することを示す行を挿入します。
13-3 ページの ALLOW_SERVER プロシージャ	特定のサーバーへのアクセスを可能にします。リストに deny all が指定されている場合でも有効です。
13-4 ページの DENY_ALL プロシージャ	リストを空にし、すべてのサーバーを信頼しないことを示す行を挿入します。
13-4 ページの DENY_SERVER プロシージャ	特定のサーバーへのアクセスを拒否します。リストに allow all が指定されている場合でも有効です。

ALLOW_ALL プロシージャ

このプロシージャは、Trusted Servers リストを空にし、企業のディレクトリ・サービスで信頼されているドメインのメンバーであるすべてのサーバー、および同じドメイン内のすべてのサーバーがアクセスを許可されることを指定します。

ビュー TRUSTED_SERVERS には、"TRUSTED ALL" と表示されます。これは、企業のディレクトリ・サービスで現在信頼されているすべてのサーバーを、データベースが信頼していることを意味します。

構文

```
DBMS_DISTRIBUTED_TRUST_ADMIN.ALLOW_ALL;
```

パラメータ

なし

例外

なし

使用上の注意

ALLOW_ALL は、企業のディレクトリ・サービスで信頼されているとしてリストされているサーバー、および同じ企業ドメイン内のサーバーにのみ適用されます。

ALLOW_SERVER プロシージャ

このプロシージャは、指定したサーバーが信頼されていることを保証します ("deny all" が指定されている場合でも有効です)。

構文

```
DBMS_DISTRIBUTED_TRUST_ADMIN.ALLOW_SERVER (  
    server IN VARCHAR2);
```

パラメータ

表 13-2 ALLOW_SERVER プロシージャのパラメータ

パラメータ	説明
server	信頼するサーバーの一意の完全修飾名

例外

なし

使用上の注意

Trusted Servers リストにエントリ "deny all" が含まれている場合、このプロシージャは、特定のデータベース（例：DBx）を信頼することを示す指定を追加します。

Trusted Servers リストにエントリ "allow all" が含まれていて、そのリストに "deny DBx" エントリがない場合は、このプロシージャを実行しても何も変更されません。

Trusted Servers リストにエントリ "allow all" が含まれていて、そのリストに "deny DBx" エントリがある場合は、そのエントリが削除されます。

DENY_ALL プロシージャ

このプロシージャは、Trusted Servers リストを空にし、すべてのサーバーがアクセスを拒否されることを指定します。

ビュー TRUSTED_SERVERS には、"UNTRUSTED ALL" が表示されます。これは、現在のどのサーバーも信頼されていないことを示します。

構文

```
DBMS_DISTRIBUTED_TRUST_ADMIN.DENY_ALL;
```

パラメータ

なし

例外

なし

DENY_SERVER プロシージャ

このプロシージャは、指定したサーバーを信頼しないことを保証します ("allow all" が指定されている場合でも有効です)。

構文

```
DBMS_DISTRIBUTED_TRUST_ADMIN.DENY_SERVER (  
    server IN VARCHAR2);
```

パラメータ

表 13-3 DENY_SERVER プロシージャのパラメータ

パラメータ	説明
server	信頼しないサーバーの一意の完全修飾名

例外

なし

使用上の注意

Trusted Servers リストにエントリ "allow all" が含まれている場合、このプロシージャは、指定したデータベース（例：DBx）を信頼しないことを示すエントリを追加します。

Trusted Servers リストにエントリ "deny all" が含まれていて、そのリストに "allow DBx" エントリがない場合は、このプロシージャを実行しても何も変更されません。

Trusted Servers リストにエントリ "deny all" が含まれていて、"allow DBx" エントリがある場合は、そのエントリが削除されます。

例

パッケージ DBMS_DISTRIBUTED_TRUST_ADMIN を使用して信頼リストを変更したことがない場合、デフォルトでは、同じ企業ドメイン内のすべてのデータベースを信頼します（そのドメインがディレクトリ・サービスで信頼されているとしてリストされている場合に限りです）。

```
SELECT * FROM TRUSTED_SERVERS;
TRUST      NAME
-----
Trusted    All
```

1 row selected.

現在すべてのサーバーが信頼されているため、DENY_SERVER プロシージャを実行すると、特定のサーバーを信頼しないことを指定できます。

```
EXECUTE DBMS_DISTRIBUTED_TRUST_ADMIN.DENY_SERVER
        ('SALES.US.AMERICAS.ACME_AUTO.COM');
```

Statement processed.

```
SELECT * FROM TRUSTED_SERVERS;

TRUST      NAME
-----
Untrusted  SALES.US.AMERICAS.ACME_AUTO.COM
```

1 row selected

DENY_ALL プロシージャを実行すると、すべてのデータベース・サーバーを信頼しないことを選択できます。

```
EXECUTE DBMS_DISTRIBUTED_TRUST_ADMIN.DENY_ALL;
```

Statement processed.

```
SELECT * FROM TRUSTED_SERVERS;
```

```
TRUST      NAME
```

```
-----  
Untrusted All
```

```
1 row selected.
```

ALLOW_SERVER プロシージャを使用すると、特定のデータベース・サーバーを信頼することを指定できます。

```
EXECUTE
```

```
DBMS_DISTRIBUTED_TRUST_ADMIN.ALLOW_SERVER  
                                ('SALES.US.AMERICAS.ACME_AUTO.COM');
```

```
Statement processed.
```

```
SELECT * FROM TRUSTED_SERVERS;
```

```
TRUST      NAME
```

```
-----  
Trusted    SALES.US.AMERICAS.ACME_AUTO.COM
```

```
1 row selected.
```

DBMS_HS には、異機種間サービス (HS) の初期化パラメータ、機能、インスタンス名およびクラス名を、設定および設定解除するサブプログラムが含まれています。

このパッケージは、SYS (connect internal) によってインストールされます。HS 表は CATHS.SQL を実行すると作成され、システムが所有します。

関連項目：『Oracle8i 分散システム』

例外

```
miss_base_caps exception;
pragma exception_init(miss_base_caps, -24274);
miss_base_caps_num number := -24274; dupl_base_caps exception;
pragma exception_init(dupl_base_caps, -24270);
dupl_base_caps_num number := -24270;
dupl_base_caps_msg varchar2(76) := 'HS$_BASE_CAPS';

miss_base_dd exception;
pragma exception_init(miss_base_dd, -24274);
miss_base_dd_num number := -24274;
miss_base_dd_msg varchar2(76) := 'HS$_BASE_DD';

dupl_base_dd exception;
pragma exception_init(dupl_base_dd, -24270);
dupl_base_dd_num number := -24270;
dupl_base_dd_msg varchar2(76) := 'HS$_BASE_DD';

miss_external_object exception;
pragma exception_init(miss_external_object, -24274);
miss_external_object_num number := -24274;
miss_external_object_msg varchar2(76) := 'HS$_EXTERNAL_OBJECTS';

dupl_external_object exception;
pragma exception_init(dupl_external_object, -24270);
dupl_external_object_num number := -24270;
dupl_external_object_msg varchar2(76) := 'HS$_EXTERNAL_OBJECTS';

miss_granted_user exception;
pragma exception_init(miss_granted_user, -24274);
miss_granted_user_num number := -24274;
miss_granted_user_msg varchar2(76) := 'HS$_GRANTED_USERS';

dupl_granted_user exception;
pragma exception_init(dupl_granted_user, -24270);
dupl_granted_user_num number := -24270;
dupl_granted_user_msg varchar2(76) := 'HS$_GRANTED_USERS';

miss_access_grantee exception;
pragma exception_init(miss_access_grantee, -24274);
miss_access_grantee_num number := -24274;
miss_access_grantee_msg varchar2(76) := 'HS$_ACCESS GRANTEES';
```

```

dupl_access_grantee exception;
pragma exception_init(dupl_access_grantee, -24270);
dupl_access_grantee_num number := -24270;
dupl_access_grantee_msg varchar2(76) := 'HS$_ACCESS_GRANTEES';

miss_create_grantee exception;
pragma exception_init(miss_create_grantee, -24274);
miss_create_grantee_num number := -24274;
miss_create_grantee_msg varchar2(76) := 'HS$_CREATE_GRANTEES';

dupl_create_grantee exception;
pragma exception_init(dupl_create_grantee, -24270);
dupl_create_grantee_num number := -24270;
dupl_create_grantee_msg varchar2(76) := 'HS$_CREATE_GRANTEES';

miss_privilege exception;
pragma exception_init(miss_privilege, -24274);
miss_privilege_num number := -24274;
miss_privilege_msg varchar2(76) := 'HS$_PRIVILEGES';

dupl_privilege exception;
pragma exception_init(dupl_privilege, -24270);
dupl_privilege_num number := -24270;
dupl_privilege_msg varchar2(76) := 'HS$_PRIVILEGES';

miss_class_caps exception;
pragma exception_init(miss_class_caps, -24274);
miss_class_caps_num number := -24274;
miss_class_caps_msg varchar2(76) := 'HS$_CLASS_CAPS';

dupl_class_caps exception;
pragma exception_init(dupl_class_caps, -24270);
dupl_class_caps_num number := -24270;
dupl_class_caps_msg varchar2(76) := 'HS$_CLASS_CAPS';

miss_class_dd exception;
pragma exception_init(miss_class_dd, -24274);
miss_class_dd_num number := -24274;
miss_class_dd_msg varchar2(76) := 'HS$_CLASS_DD';

dupl_class_dd exception;
pragma exception_init(dupl_class_dd, -24270);
dupl_class_dd_num number := -24270;
dupl_class_dd_msg varchar2(76) := 'HS$_CLASS_DD';

```

```

bad_TRANSLATION_TYPE exception;
pragma exception_init(bad_TRANSLATION_TYPE, -24271);
bad_TRANSLATION_TYPE_num number := -24271;
bad_TRANSLATION_TYPE_msg varchar2(76) := 'NULL';

bad_TRANSLATION_TEXT exception;
pragma exception_init(bad_TRANSLATION_TEXT, -24273);
bad_TRANSLATION_TEXT_num number := -24273;
bad_TRANSLATION_TEXT_msg varchar2(76) := 'NULL';

miss_class_init exception;
pragma exception_init(miss_class_init, -24274);
miss_class_init_num number := -24274;
miss_class_init_msg varchar2(76) := 'HS$_CLASS_INIT';

dupl_class_init exception;
pragma exception_init(dupl_class_init, -24270);
dupl_class_init_num number := -24270;
dupl_class_init_msg varchar2(76) := 'HS$_CLASS_INIT';

bad_INIT_VALUE_TYPE exception;
pragma exception_init(bad_INIT_VALUE_TYPE, -24272);
bad_INIT_VALUE_TYPE_num number := -24272;
bad_INIT_VALUE_TYPE_msg varchar2(76) := 'NULL';

miss_fds_class exception;
pragma exception_init(miss_fds_class, -24274);
miss_fds_class_num number := -24274;
miss_fds_class_msg varchar2(76) := 'HS$_FDS_CLASS';

dupl_fds_class exception;
pragma exception_init(dupl_fds_class, -24270);
dupl_fds_class_num number := -24270;
dupl_fds_class_msg varchar2(76) := 'HS$_FDS_CLASS';

miss_fds_inst exception;
pragma exception_init(miss_fds_inst, -24274);
miss_fds_inst_num number := -24274;
miss_fds_inst_msg varchar2(76) := 'HS$_FDS_INST';

dupl_fds_inst exception;
pragma exception_init(dupl_fds_inst, -24270);
dupl_fds_inst_num number := -24270;
dupl_fds_inst_msg varchar2(76) := 'HS$_FDS_INST';

```

```

miss_inst_caps exception;
pragma exception_init(miss_inst_caps, -24274);
miss_inst_caps_num number := -24274;
miss_inst_caps_msg varchar2(76) := 'HS$_INST_CAPS';

dupl_inst_caps exception;
pragma exception_init(dupl_inst_caps, -24270);
dupl_inst_caps_num number := -24270;
dupl_inst_caps_msg varchar2(76) := 'HS$_INST_CAPS';

miss_inst_dd exception;
pragma exception_init(miss_inst_dd, -24274);
miss_inst_dd_num number := -24274;
miss_inst_dd_msg varchar2(76) := 'HS$_INST_DD';

dupl_inst_dd exception;
pragma exception_init(dupl_inst_dd, -24270);
dupl_inst_dd_num number := -24270;
dupl_inst_dd_msg varchar2(76) := 'HS$_INST_DD';

miss_inst_init exception;
pragma exception_init(miss_inst_init, -24274);
miss_inst_init_num number := -24274;
miss_inst_init_msg varchar2(76) := 'HS$_INST_INIT';

dupl_inst_init exception;
pragma exception_init(dupl_inst_init, -24270);
dupl_inst_init_num number := -24270;
dupl_inst_init_msg varchar2(76) := 'HS$_INST_INIT';

no_privilege exception;
pragma exception_init(no_privilege, -24277);
no_privilege_num number := -24277;
no_privilege_msg varchar2(76) := null;

privilege_mismatch exception;
pragma exception_init(privilege_mismatch, -24278);
privilege_mismatch_num number := -24278;
privilege_mismatch_msg varchar2(76) := null;

lib_priv_mismatch exception;
pragma exception_init(lib_priv_mismatch, -24279);
lib_priv_mismatch_num number := -24279;
lib_priv_mismatch_msg varchar2(76) := null;

```

サブプログラムの要約

表 14-1 DBMS_HS パッケージのサブプログラム

サブプログラム	説明
14-8 ページの ALTER_BASE_CAPS プロシージャ	HS\$_BASE_CAPS 表の行を変更します。
14-9 ページの ALTER_BASE_DD プロシージャ	HS\$_BASE_DD 表の行を変更します。
14-9 ページの ALTER_CLASS_CAPS プロシージャ	HS\$_CLASS_CAPS 表の内容を変更します。
14-9 ページの ALTER_CLASS_DD プロシージャ	HS\$_CLASS_DD 表の内容を変更します。
14-9 ページの ALTER_CLASS_INIT プロシージャ	HS\$_CLASS_INIT 表の内容を変更します。
14-10 ページの ALTER_FDS_CLASS プロシージャ	HS\$_FDS_CLASS 表の内容を変更します。
14-10 ページの ALTER_FDS_INST プロシージャ	HS\$_FDS_INST 表の内容を変更します。
14-10 ページの ALTER_INST_CAPS プロシージャ	\$HS_INST_CAPS 表の内容を変更します。
14-11 ページの ALTER_INST_DD プロシージャ	HS\$_INST_DD 表の内容を変更します。
14-11 ページの ALTER_INST_INIT プロシージャ	HS\$_INST_INIT 表の内容を変更します。
14-12 ページの COPY_CLASS プロシージャ	クラスに関するすべての内容を別のクラスにコピーします。
14-12 ページの COPY_INST プロシージャ	HS\$_FDS_INST に関するすべての内容を、同じ FDS_ CLASS 内の新規インスタンスにコピーします。
14-12 ページの CREATE_BASE_CAPS プロシージャ	HS\$_BASE_CAPS 表に行を作成します。
14-12 ページの CREATE_BASE_DD プロシージャ	HS\$_BASE_DD 表に行を作成します。
14-12 ページの CREATE_BASE_DD プロシージャ	HS\$_BASE_DD 表に行を作成します。
14-12 ページの CREATE_CLASS_CAPS プロシージャ	HS\$_CLASS_CAPS 表に行を作成します。

表 14-1 DBMS_HS パッケージのサブプログラム

サブプログラム	説明
14-13 ページの CREATE_CLASS_DD プロシージャ	HS\$_CLASS_DD 表に行を作成します。
14-13 ページの CREATE_CLASS_INIT プロシージャ	HS\$_CLASS_INIT 表に行を作成します。
14-14 ページの CREATE_FDS_CLASS プロシージャ	HS\$_FDS_CLASS 表に行を作成します。
14-14 ページの CREATE_FDS_INST プロシージャ	HS\$_FDS_INST 表に行を作成します。
14-14 ページの CREATE_INST_CAPS プロシージャ	HS\$_INST_CAPS 表に行を作成します。
14-14 ページの CREATE_INST_DD プロシージャ	HS\$_INST_DD 表に行を作成します。
14-15 ページの CREATE_INST_INIT プロシージャ	HS\$_INST_INIT 表に行を作成します。
14-15 ページの DROP_BASE_CAPS プロシージャ	CAP_NUMBER パラメータの指定どおりに、HS\$_BASE_CAPS 表から行を削除します。
14-16 ページの DROP_BASE_DD プロシージャ	table_name の指定どおりに、HS\$_BASE_DD 表から行を削除します。
14-16 ページの DROP_CLASS_CAPS プロシージャ	FDS_CLASS_NAME と CAP_NUMBER の指定どおりに、HS\$_CLASS_CAPS 表から行を削除します。
14-16 ページの DROP_CLASS_DD プロシージャ	FDS_CLASS_NAME と DD_TABLE_NAME の指定どおりに、HS\$_CLASS_DD の行を削除します。
14-16 ページの DROP_CLASS_INIT プロシージャ	FDS_CLASS_NAME と INIT_VALUE_NAME の指定どおりに、HS\$_CLASS_INIT の行を削除します。
14-17 ページの DROP_FDS_CLASS プロシージャ	FDS_CLASS_NAME の指定どおりに、HS\$_FDS_CLASS の行を削除します。
14-17 ページの DROP_FDS_INST プロシージャ	FDS_INST_NAME と FDS_CLASS_NAME の指定どおりに、HS\$_FDS_INST 表の行を削除します。
14-17 ページの DROP_INST_CAPS プロシージャ	FDS_INST_NAME、FDS_CLASS_NAME および CAP_NUMBER の指定どおりに、HS\$_INST_CAPS の行を削除します。
14-17 ページの DROP_INST_DD プロシージャ	FDS_INST_NAME、FDS_CLASS_NAME および DD_TABLE_NAME の指定どおりに、HS\$_INST_DD から行を削除します。

表 14-1 DBMS_HS パッケージのサブプログラム

サブプログラム	説明
14-18 ページの DROP_INST_INIT プロシージャ	FDS_INST_NAME、FDS_CLASS_NAME および INIT_VALUE_NAME の指定どおりに、HS\$_INST_INIT 表から行を削除します。
14-18 ページの REPLACE_BASE_CAPS プロシージャ	HS\$_BASE_CAPS 表の行を作成または置換します。
14-18 ページの REPLACE_BASE_DD プロシージャ	HS\$_BASE_DD 表の行を作成または置換します。
14-19 ページの REPLACE_CLASS_CAPS プロシージャ	HS\$_CLASS_CAPS 表上で作成または置換を実行します。
14-19 ページの REPLACE_CLASS_DD プロシージャ	HS\$_CLASS_DD 表上で作成または置換を実行します。
14-19 ページの REPLACE_CLASS_INIT プロシージャ	HS\$_CLASS_INIT 表の行を作成または更新します。
14-20 ページの REPLACE_FDS_CLASS プロシージャ	HS\$_FDS_CLASS 表上で作成または置換操作を実行します。
14-20 ページの REPLACE_FDS_INST プロシージャ	HS\$_FDS_INST 表の行を作成または置換します。
14-20 ページの REPLACE_INST_CAPS プロシージャ	HS\$_INST_CAPS 表上で作成または置換を実行します。
14-21 ページの REPLACE_INST_DD プロシージャ	HS\$_INST_DD 表上で作成または置換操作を実行します。
14-21 ページの REPLACE_INST_INIT プロシージャ	HS\$_INST_INIT 表上で作成または置換を実行します。

ALTER_BASE_CAPS プロシージャ

このプロシージャは、HS\$_BASE_CAPS 表の行を変更します。

構文

```
DBMS_HS.ALTER_BASE_CAPS (  
    CAP_NUMBER          IN NUMBER,  
    new_CAP_NUMBER      IN NUMBER    := -1e-130,  
    new_CAP_DESCRIPTION IN VARCHAR2 := '-');
```

ALTER_BASE_DD プロシージャ

このプロシージャは、HS\$_BASE_DD 表の行を変更します。

構文

```
DBMS_HS.ALTER_BASE_DD (  
    DD_TABLE_NAME      IN VARCHAR2,  
    new_DD_TABLE_NAME  IN VARCHAR2 := '-',  
    new_DD_TABLE_DESC  IN VARCHAR2 := '-');
```

ALTER_CLASS_CAPS プロシージャ

このプロシージャは、HS\$_CLASS_CAPS 表の内容を変更します。

構文

```
DBMS_HS.ALTER_CLASS_CAPS (  
    FDS_CLASS_NAME      IN VARCHAR2,  
    CAP_NUMBER          IN NUMBER,  
    new_FDS_CLASS_NAME  IN VARCHAR2 := '-',  
    new_CAP_NUMBER      IN NUMBER   := -1e-130,  
    new_CONTEXT         IN NUMBER   := -1e-130,  
    new_TRANSLATION     IN VARCHAR2 := '-',  
    new_ADDITIONAL_INFO IN NUMBER   := -1e-130);
```

ALTER_CLASS_DD プロシージャ

このプロシージャは、HS\$_CLASS_DD 表の内容を変更します。

構文

```
DBMS_HS.ALTER_CLASS_DD (  
    FDS_CLASS_NAME      IN VARCHAR2,  
    DD_TABLE_NAME       IN VARCHAR2,  
    new_FDS_CLASS_NAME  IN VARCHAR2 := '-',  
    new_DD_TABLE_NAME   IN VARCHAR2 := '-',  
    new_TRANSLATION_TYPE IN CHAR     := '-',  
    new_TRANSLATION_TEXT IN VARCHAR2 := '-');
```

ALTER_CLASS_INIT プロシージャ

このプロシージャは、HS\$_CLASS_INIT 表の内容を変更します。

構文

```
DEMS_HS.ALTER_CLASS_INIT (
    FDS_CLASS_NAME      IN VARCHAR2,
    INIT_VALUE_NAME     IN VARCHAR2,
    new_FDS_CLASS_NAME  IN VARCHAR2 := '-',
    new_INIT_VALUE_NAME IN VARCHAR2 := '-',
    new_INIT_VALUE      IN VARCHAR2 := '-',
    new_INIT_VALUE_TYPE IN VARCHAR2 := '-');
```

ALTER_FDS_CLASS プロシージャ

このプロシージャは、HS\$_FDS_CLASS 表の内容を変更します。

構文

```
DEMS_HS.ALTER_FDS_CLASS (
    FDS_CLASS_NAME      IN VARCHAR2,
    new_FDS_CLASS_NAME  IN VARCHAR2 := '-',
    new_FDS_CLASS_COMMENTS IN VARCHAR2 := '-');
```

ALTER_FDS_INST プロシージャ

このプロシージャは、HS\$_FDS_INST 表の内容を変更します。

構文

```
DEMS_HS.ALTER_FDS_INST (
    FDS_INST_NAME      IN VARCHAR2,
    FDS_CLASS_NAME     IN VARCHAR2,
    new_FDS_INST_NAME  IN VARCHAR2 := '-',
    new_FDS_CLASS_NAME IN VARCHAR2 := '-',
    new_FDS_INST_COMMENTS IN VARCHAR2 := '-');
```

ALTER_INST_CAPS プロシージャ

このプロシージャは、\$HS_INST_CAPS 表の内容を変更します。

構文

```
DBMS_HS.ALTER_INST_CAPS (  
    FDS_INST_NAME      IN VARCHAR2,  
    FDS_CLASS_NAME     IN VARCHAR2,  
    CAP_NUMBER         IN NUMBER,  
    new_FDS_INST_NAME  IN VARCHAR2 := '-',  
    new_FDS_CLASS_NAME IN VARCHAR2 := '-',  
    new_CAP_NUMBER     IN NUMBER   := -1e-130,  
    new_CONTEXT        IN NUMBER   := -1e-130,  
    new_TRANSLATION    IN VARCHAR2 := '-',  
    new_ADDITIONAL_INFO IN NUMBER   := -1e-130);
```

ALTER_INST_DD プロシージャ

このプロシージャは、HS\$_INST_DD 表の内容を変更します。

構文

```
DBMS_HS.ALTER_INST_DD (  
    FDS_INST_NAME      IN VARCHAR2,  
    FDS_CLASS_NAME     IN VARCHAR2,  
    DD_TABLE_NAME      IN VARCHAR2,  
    new_FDS_INST_NAME  IN VARCHAR2 := '-',  
    new_FDS_CLASS_NAME IN VARCHAR2 := '-',  
    new_DD_TABLE_NAME  IN VARCHAR2 := '-',  
    new_TRANSLATION_TYPE IN CHAR    := '-',  
    new_TRANSLATION_TEXT IN VARCHAR2 := '-');
```

ALTER_INST_INIT プロシージャ

このプロシージャは、HS\$_INST_INIT 表の内容を変更します。

構文

```
DBMS_HS.ALTER_INST_INIT (  
    FDS_INST_NAME      IN VARCHAR2,  
    FDS_CLASS_NAME     IN VARCHAR2,  
    INIT_VALUE_NAME    IN VARCHAR2,  
    new_FDS_INST_NAME  IN VARCHAR2 := '-',  
    new_FDS_CLASS_NAME IN VARCHAR2 := '-',  
    new_INIT_VALUE_NAME IN VARCHAR2 := '-',  
    new_INIT_VALUE     IN VARCHAR2 := '-',  
    new_INIT_VALUE_TYPE IN VARCHAR2 := '-');
```

COPY_CLASS プロシージャ

このプロシージャは、クラスに関するすべての内容を別のクラスにコピーします。

構文

```
DBMS_HS.COPY_CLASS (  
    old_fds_class_name VARCHAR2,
```

COPY_INST プロシージャ

このプロシージャは、HS\$_FDS_INST に関するすべての内容を、同じ FDS_CLASS 内の新規インスタンスにコピーします。

構文

```
DBMS_HS.COPY_INST (  
    FDS_INST_NAME      IN VARCHAR2,  
    FDS_CLASS_NAME     IN VARCHAR2,  
    new_FDS_INST_NAME  IN VARCHAR2,  
    new_FDS_COMMENTS   IN VARCHAR2 default '-');
```

CREATE_BASE_CAPS プロシージャ

このプロシージャは、HS\$_BASE_CAPS 表に行を作成します。

構文

```
DBMS_HS.CREATE_BASE_CAPS (  
    CAP_NUMBER      IN NUMBER,  
    CAP_DESCRIPTION IN VARCHAR2 := NULL);
```

CREATE_BASE_DD プロシージャ

このプロシージャは、HS\$_BASE_DD 表に行を作成します。

構文

```
DBMS_HS.CREATE_BASE_DD (  
    DD_TABLE_NAME IN VARCHAR2,  
    DD_TABLE_DESC IN VARCHAR2 := NULL);
```

CREATE_CLASS_CAPS プロシージャ

このプロシージャは、HS\$_CLASS_CAPS 表に行を作成します。

FDS_CLASS_NAME が HS\$_FDS_CLASS 表に存在している必要があります。CAP_NUMBER が HS\$_BASE_CAPS 表に定義されている必要があります。

構文

```
DBMS_HS.CREATE_CLASS_CAPS (  
    FDS_CLASS_NAME  IN VARCHAR2,  
    CAP_NUMBER      IN NUMBER,  
    CONTEXT         IN NUMBER    := NULL,  
    TRANSLATION     IN VARCHAR2  := NULL,  
    ADDITIONAL_INFO IN NUMBER    := NULL);
```

CREATE_CLASS_DD プロシージャ

このプロシージャは、HS\$_CLASS_DD 表に行を作成します。

FDS_CLASS_NAME が HS\$_FDS_CLASS 表に存在している必要があります。DD_TABLE_NAME が HS\$_BASE_DD 表に存在している必要があります。TRANSLATION_TYPE には、'T' (変換済) または 'M' (代替済) のいずれかを設定する必要があります。TRANSLATION_TYPE = 'T' の場合は、TRANSLATION_TEXT 文字列を指定する必要があります。

構文

```
DBMS_HS.CREATE_CLASS_DD (  
    FDS_CLASS_NAME  IN VARCHAR2,  
    DD_TABLE_NAME   IN VARCHAR2,  
    TRANSLATION_TYPE IN CHAR,  
    TRANSLATION_TEXT IN VARCHAR2 := NULL);
```

CREATE_CLASS_INIT プロシージャ

このプロシージャは、HS\$_CLASS_INIT 表に行を作成します。

FDS_CLASS_NAME が HS\$_FDS_CLASS 表に存在している必要があります。INIT_VALUE_TYPE には、'F' (環境変数) または 'M' (非環境変数) のいずれかを設定する必要があります。

構文

```
DBMS_HS.CREATE_CLASS_INIT (  
    FDS_CLASS_NAME  IN VARCHAR2,  
    INIT_VALUE_NAME IN VARCHAR2,  
    INIT_VALUE      IN VARCHAR2,  
    INIT_VALUE_TYPE IN VARCHAR2);
```

CREATE_FDS_CLASS プロシージャ

このプロシージャは、HS\$_FDS_CLASS 表に行を作成します。

構文

```
DBMS_HS.CREATE_FDS_CLASS (  
    FDS_CLASS_NAME      IN VARCHAR2,  
    FDS_CLASS_COMMENTS IN VARCHAR2 := NULL);
```

CREATE_FDS_INST プロシージャ

このプロシージャは、HS\$_FDS_INST 表に行を作成します。

FDS_CLASS_NAME が HS\$_FDS_CLASS 表に存在している必要があります。

構文

```
DBMS_HS.CREATE_FDS_INST (  
    FDS_INST_NAME      IN VARCHAR2,  
    FDS_CLASS_NAME     IN VARCHAR2,  
    FDS_INST_COMMENTS IN VARCHAR2 := NULL);
```

CREATE_INST_CAPS プロシージャ

このプロシージャは、HS\$_INST_CAPS 表に行を作成します。

FDS_INST_NAME が HS\$_FDS_INST 表に存在し、FDS_CLASS_NAME 行で指定された HS\$_FDS_CLASS の行に対して定義されている必要があります。CAP_NUMBER が HS\$_BASE_CAPS 表に定義されている必要があります。

構文

```
DBMS_HS.CREATE_INST_CAPS (  
    FDS_INST_NAME      IN VARCHAR2,  
    FDS_CLASS_NAME     IN VARCHAR2,  
    CAP_NUMBER         IN NUMBER,  
    CONTEXT            IN NUMBER  := NULL,  
    TRANSLATION        IN VARCHAR2 := NULL,  
    ADDITIONAL_INFO    IN NUMBER  := NULL);
```

CREATE_INST_DD プロシージャ

このプロシージャは、HS\$_INST_DD 表に行を作成します。

FDS_INST_NAME が HS\$_FDS_INST 表に定義され、HS\$_FDS_CLASS パラメータで指定された FDS_CLASS に属している必要があります。DD_TABLE_NAME が HS\$_BASE_DD 表に定

義されている必要があります。TRANSLATION_TYPE には、'T' (変換済) または 'M' (代替済) のいずれかを設定する必要があります。TRANSLATION_TYPE が 'T' の場合は、TRANSLATION_TEXT を指定する必要があります。

構文

```
DBMS_HS.CREATE_INST_DD (
    FDS_INST_NAME      IN VARCHAR2,
    FDS_CLASS_NAME     IN VARCHAR2,
    DD_TABLE_NAME      IN VARCHAR2,
    TRANSLATION_TYPE IN CHAR,
    TRANSLATION_TEXT IN VARCHAR2 := NULL);
```

CREATE_INST_INIT プロシージャ

このプロシージャは、HS\$_INST_INIT 表に行を作成します。

FDS_INST_NAME が HS\$_FDS_INST 表に存在し、FDS_CLASS_NAME パラメータの指定どおりに HS\$_FDS_CLASS 表に存在していることが必要です。INIT_VALUE_TYPE は、'F' または 'T' のいずれかに定義されている必要があります。

構文

```
DBMS_HS.CREATE_INST_INIT(
    FDS_INST_NAME      IN VARCHAR2,
    FDS_CLASS_NAME     IN VARCHAR2,
    INIT_VALUE_NAME    IN VARCHAR2,
    INIT_VALUE         IN VARCHAR2,
    INIT_VALUE_TYPE    IN VARCHAR2);
```

DROP_BASE_CAPS プロシージャ

このプロシージャは、CAP_NUMBER パラメータの指定どおりに、HS\$_BASE_CAPS 表から行を削除します。

構文

```
DBMS_HS.DROP_BASE_CAPS (
    CAP_NUMBER IN NUMBER);
```

DROP_BASE_DD プロシージャ

このプロシージャは、table_name の指定どおりに、HS\$_BASE_DD 表から行を削除します。

構文

```
DEMS_HS.DROP_BASE_DD (  
    DD_TABLE_NAME IN VARCHAR2);
```

DROP_CLASS_CAPS プロシージャ

このプロシージャは、FDS_CLASS_NAME と CAP_NUMBER の指定どおりに、HS\$_CLASS_CAPS 表から行を削除します。

構文

```
DEMS_HS.DROP_CLASS_CAPS (  
    FDS_CLASS_NAME IN VARCHAR2,  
    CAP_NUMBER      IN NUMBER);
```

DROP_CLASS_DD プロシージャ

このプロシージャは、FDS_CLASS_NAME と DD_TABLE_NAME の指定どおりに、HS\$_CLASS_DD の行を削除します。

構文

```
DEMS_HS.DROP_CLASS_DD(  
    FDS_CLASS_NAME IN VARCHAR2,  
    DD_TABLE_NAME  IN VARCHAR2);
```

DROP_CLASS_INIT プロシージャ

FDS_CLASS_NAME と INIT_VALUE_NAME の指定どおりに、HS\$_CLASS_INIT の行を削除します。

構文

```
DEMS_HS.DROP_CLASS_INIT (  
    FDS_CLASS_NAME IN VARCHAR2,  
    INIT_VALUE_NAME IN VARCHAR2);
```

DROP_FDS_CLASS プロシージャ

このプロシージャは、FDS_CLASS_NAME の指定どおりに、HS\$_FDS_CLASS の行を削除します。

構文

```
DBMS_HS.DROP_FDS_CLASS (  
    FDS_CLASS_NAME IN VARCHAR2);
```

DROP_FDS_INST プロシージャ

このプロシージャは、FDS_INST_NAME と FDS_CLASS_NAME の指定どおりに、HS\$_FDS_INST 表の行を削除します。

構文

```
DBMS_HS.DROP_FDS_INST (  
    FDS_INST_NAME  IN VARCHAR2,  
    FDS_CLASS_NAME IN VARCHAR2);
```

DROP_INST_CAPS プロシージャ

このプロシージャは、FDS_INST_NAME、FDS_CLASS_NAME および CAP_NUMBER の指定どおりに、HS\$_INST_CAPS の行を削除します。

構文

```
DBMS_HS.DROP_INST_CAPS (  
    FDS_INST_NAME  IN VARCHAR2,  
    FDS_CLASS_NAME IN VARCHAR2,  
    CAP_NUMBER     IN NUMBER);
```

DROP_INST_DD プロシージャ

このプロシージャは、FDS_INST_NAME、FDS_CLASS_NAME および DD_TABLE_NAME の指定どおりに、HS\$_INST_DD から行を削除します。

構文

```
DBMS_HS.DROP_INST_DD (  
    FDS_INST_NAME  IN VARCHAR2,  
    FDS_CLASS_NAME IN VARCHAR2,  
    DD_TABLE_NAME  IN VARCHAR2);
```

DROP_INST_INIT プロシージャ

FDS_INST_NAME、FDS_CLASS_NAME および INIT_VALUE_NAME の指定どおりに、HS\$INST_INIT 表から行を削除します。

構文

```
DEMS_HS.DROP_INST_INIT (
    FDS_INST_NAME    IN    VARCHAR2,
    FDS_CLASS_NAME   IN    VARCHAR2,
    INIT_VALUE_NAME  IN    VARCHAR2);
new_fds_class_name   VARCHAR2,
new_fds_class_comments VARCHAR2 default '-');
```

REPLACE_BASE_CAPS プロシージャ

このプロシージャは、HS\$_BASE_CAPS 表の行を作成または置換します。

最初に HS\$_BASE_CAPS の行を更新しようとし、行が存在しない場合は挿入します。new_CAP_NUMBER パラメータは、CAP_NUMBER で指定された行が存在しない場合は無視されます。

構文

```
DEMS_HS.REPLACE_BASE_CAPS (
    CAP_NUMBER        IN NUMBER,
    new_CAP_NUMBER     IN NUMBER    := NULL,
    new_CAP_DESCRIPTION IN VARCHAR2 := NULL);
```

REPLACE_BASE_DD プロシージャ

このプロシージャは、HS\$_BASE_DD 表の行を作成または置換します。

最初に、行を更新しようとします。行が存在しない場合は挿入され、new_DD_TABLE_NAME パラメータは無視されます。

構文

```
DEMS_HS.REPLACE_BASE_DD (
    DD_TABLE_NAME      IN VARCHAR2,
    new_DD_TABLE_NAME  IN VARCHAR2 := NULL,
    new_DD_TABLE_DESC  IN VARCHAR2 := NULL);
```


REPLACE_CLASS_CAPS プロシージャ

このプロシージャは、HS\$_CLASS_CAPS 表上で作成または置換を実行します。

FDS_CLASS_NAME と CAP_NUMBER に対する行が存在している場合は、その行が更新されます。行が存在しない場合は挿入され、new_FDS_CLASS_NAME および new_CAP_NUMBER パラメータは無視されます。

構文

```
DBMS_HS.REPLACE_CLASS_CAPS (  
    FDS_CLASS_NAME      IN VARCHAR2,  
    CAP_NUMBER          IN NUMBER,  
    new_FDS_CLASS_NAME  IN VARCHAR2 := NULL,  
    new_CAP_NUMBER      IN NUMBER   := NULL,  
    new_CONTEXT         IN NUMBER   := NULL,  
    new_TRANSLATION     IN VARCHAR2 := NULL,  
    new_ADDITIONAL_INFO IN NUMBER   := NULL);
```

REPLACE_CLASS_DD プロシージャ

このプロシージャは、HS\$_CLASS_DD 表上で作成または置換を実行します。

FDS_CLASS_NAME と DD_TABLE_NAME に対する行が存在している場合は、その行が更新されます。行が存在しない場合は挿入され、new_FDS_CLASS_NAME および new_DD_TABLE_NAME パラメータは無視されます。

構文

```
DBMS_HS.REPLACE_CLASS_DD (  
    FDS_CLASS_NAME      IN VARCHAR2,  
    DD_TABLE_NAME       IN VARCHAR2,  
    new_FDS_CLASS_NAME  IN VARCHAR2 := NULL,  
    new_DD_TABLE_NAME   IN VARCHAR2 := NULL,  
    new_TRANSLATION_TYPE IN CHAR     := NULL,  
    new_TRANSLATION_TEXT IN VARCHAR2 := NULL);
```

REPLACE_CLASS_INIT プロシージャ

このプロシージャは、HS\$_CLASS_INIT 表の行を作成または更新します。

指定した FDS_CLASS_NAME と INIT_VALUE_NAME に対する行が存在している場合は、その行が更新されます。行が存在しない場合は挿入され、new_FDS_CLASS_NAME および new_INIT_VALUE_NAME パラメータは無視されます。

構文

```
DBMS_HS.REPLACE_CLASS_INIT (
    FDS_CLASS_NAME      IN VARCHAR2,
    INIT_VALUE_NAME     IN VARCHAR2,
    new_FDS_CLASS_NAME  IN VARCHAR2 := NULL,
    new_INIT_VALUE_NAME IN VARCHAR2 := NULL,
    new_INIT_VALUE      IN VARCHAR2 := NULL,
    new_INIT_VALUE_TYPE IN VARCHAR2 := NULL);
```

REPLACE_FDS_CLASS プロシージャ

このプロシージャは、HS\$_FDS_CLASS 表上で作成または置換操作を実行します。

FDS_CLASS_NAME に対する行が存在している場合は、その行が更新されます。行が存在しない場合は作成され、new_FDS_CLASS_NAME パラメータは無視されます。

構文

```
DBMS_HS.REPLACE_FDS_CLASS (
    FDS_CLASS_NAME      IN VARCHAR2,
    new_FDS_CLASS_NAME  IN VARCHAR2 := NULL,
    new_FDS_CLASS_COMMENTS IN VARCHAR2 := NULL);
```

REPLACE_FDS_INST プロシージャ

このプロシージャは、HS\$_FDS_INST 表の行を作成または置換します。

FDS_INST_NAME と FDS_CLASS_NAME に対する行が存在している場合は、その行が更新されます。行が存在しない場合は作成され、new_FDS_INST_NAME および new_FDS_CLASS_NAME パラメータは無視されます。

構文

```
DBMS_HS.REPLACE_FDS_INST (
    FDS_INST_NAME      IN VARCHAR2,
    FDS_CLASS_NAME     IN VARCHAR2,
    new_FDS_INST_NAME  IN VARCHAR2 := NULL,
    new_FDS_CLASS_NAME IN VARCHAR2 := NULL,
    new_FDS_INST_COMMENTS IN VARCHAR2 := NULL);
```

REPLACE_INST_CAPS プロシージャ

このプロシージャは、HS\$_INST_CAPS 表上で作成または置換を実行します。

FDS_INST_NAME、FDS_CLASS_NAME および CAP_NUMBER に対する行が存在しない場合は作成されます。行が存在している場合は、その行が更新されます。挿入が実行される場合

は、new_FDS_INST_NAME、new_FDS_CLASS_NAME および new_CLASS_NUMBER パラメータは無視されます。

構文

```
DBMS_HS.REPLACE_INST_CAPS (  
    FDS_INST_NAME      IN VARCHAR2,  
    FDS_CLASS_NAME     IN VARCHAR2,  
    CAP_NUMBER         IN NUMBER,  
    new_FDS_INST_NAME  IN VARCHAR2 := NULL,  
    new_FDS_CLASS_NAME IN VARCHAR2 := NULL,  
    new_CAP_NUMBER     IN NUMBER   := NULL,  
    new_CONTEXT        IN NUMBER   := NULL,  
    new_TRANSLATION    IN VARCHAR2 := NULL,  
    new_ADDITIONAL_INFO IN NUMBER  := NULL);
```

REPLACE_INST_DD プロシージャ

このプロシージャは、HS\$_INST_DD 表上で作成または置換操作を実行します。

FDS_INST_NAME、FDS_CLASS_NAME および DD_TABLE_NAME に対する行が存在している場合は、その行が更新されます。行が存在しない場合は作成され、new_FDS_INST_NAME、new_FDS_CLASS_NAME および new_DD_TABLE_NAME の値は無視されます。

構文

```
DBMS_HS.REPLACE_INST_DD (  
    FDS_INST_NAME      IN VARCHAR2,  
    FDS_CLASS_NAME     IN VARCHAR2,  
    DD_TABLE_NAME      IN VARCHAR2,  
    new_FDS_INST_NAME  IN VARCHAR2 := NULL,  
    new_FDS_CLASS_NAME IN VARCHAR2 := NULL,  
    new_DD_TABLE_NAME  IN VARCHAR2 := NULL,  
    new_TRANSLATION_TYPE IN CHAR    := NULL,  
    new_TRANSLATION_TEXT IN VARCHAR2 := NULL);
```

REPLACE_INST_INIT プロシージャ

このプロシージャは、HS\$_INST_INIT 表上で作成または置換を実行します。

FDS_INST_NAME、FDS_CLASS_NAME および INIT_VALUE_NAME に対する行が存在している場合は、その行が更新されます。行が存在しない場合は作成されます。作成される場合は、new_FDS_INST_NAME、new_FDS_CLASS_NAME および new_INIT_VALUE_NAME は無視されます。

構文

```
DBMS_HS.REPLACE_INST_INIT (
    FDS_INST_NAME          IN VARCHAR2,
    FDS_CLASS_NAME         IN VARCHAR2,
    INIT_VALUE_NAME        IN VARCHAR2,
    new_FDS_INST_NAME      IN VARCHAR2 := NULL,
    new_FDS_CLASS_NAME     IN VARCHAR2 := NULL,
    new_INIT_VALUE_NAME    IN VARCHAR2 := NULL,
    new_INIT_VALUE         IN VARCHAR2 := NULL,
    new_INIT_VALUE_TYPE    IN VARCHAR2 := NULL);
```

DBMS_HS_PASSTHROUGH

パススルー SQL 機能を使用すると、アプリケーション開発者は、Oracle Server による解釈を経由せずに非 Oracle システムに文を直接送信できます。この機能は、非 Oracle システムで利用できる文と同等の文が Oracle がない場合に役立ちます。

PL/SQL パッケージ DBMS_HS_PASSTHROUGH を使用すると、これらの文を非 Oracle システムで直接実行できます。このパッケージで実行される文は、標準の " 透過的な " SQL 文と同じトランザクションで実行されます。

関連項目： 異機種間サービスおよび変数のバインド方法は、『Oracle8i 分散システム』を参照してください。

セキュリティ

DBMS_HS_PASSTHROUGH パッケージは、概念的には非 Oracle システムに常駐します。パッケージ内のプロシージャおよびファンクションは、非 Oracle システムへの適切なデータベース・リンクを使用してコールする必要があります。

サブプログラムの要約

表 15-1 DBMS_HS_PASSTHROUGH パッケージのサブプログラム

サブプログラム	説明
15-3 ページの BIND_VARIABLE プロシージャ	位置を基準にして IN 変数を PL/SQL プログラム変数にバインドします。
15-4 ページの BIND_VARIABLE_RAW プロシージャ	RAW 型 IN 変数をバインドします。
15-5 ページの BIND_OUT_VARIABLE プロシージャ	OUT 変数を PL/SQL プログラム変数にバインドします。
15-7 ページの BIND_OUT_VARIABLE_RAW プロシージャ	データ型 RAW の OUT 変数を PL/SQL プログラム変数にバインドします。
15-8 ページの BIND_INOUT_VARIABLE プロシージャ	IN OUT バインド変数をバインドします。
15-9 ページの BIND_INOUT_VARIABLE_RAW プロシージャ	データ型 RAW の IN OUT バインド変数をバインドします。
15-10 ページの CLOSE_CURSOR プロシージャ	SQL 文が非 Oracle システムで実行された後、カーソルをクローズし、関連メモリーを解放します。
15-11 ページの EXECUTE_IMMEDIATE プロシージャ	バインド変数を使用せずに、(SELECT 以外の) SQL 文を即時実行します。
15-12 ページの EXECUTE_NON_QUERY ファンクション	(SELECT 以外の) SQL 文を実行します。
15-13 ページの FETCH_ROW ファンクション	問合せから行をフェッチします。
15-14 ページの GET_VALUE プロシージャ	SELECT 文から列値を取り出す、または OUT バインド・パラメータを取り出します。
15-16 ページの GET_VALUE_RAW プロシージャ	データ型 RAW が対象である以外は GET_VALUE と同じです。
15-17 ページの OPEN_CURSOR ファンクション	非 Oracle システムでパススルー SQL 文を実行するためのカーソルをオープンします。

表 15-1 DBMS_HS_PASSTHROUGH パッケージのサブプログラム

サブプログラム	説明
15-17 ページの PARSE プロシージャ	非 Oracle システムで SQL 文を解析します。

BIND_VARIABLE プロシージャ

このプロシージャは、位置を基準にして IN 変数を PL/SQL プログラム変数にバインドします。

構文

```
DBMS_HS_PASSTHROUGH.BIND_VARIABLE (  
  c      IN BINARY_INTEGER NOT NULL,  
  pos    IN BINARY_INTEGER NOT NULL,  
  val    IN <dt>,  
  name   IN VARCHAR2);
```

<dt> は DATE、NUMBER または VARCHAR2 のいずれかです。

関連項目： RAW 変数のバインドには、15-4 ページの「[BIND_VARIABLE_RAW プロシージャ](#)」を使用します。

パラメータ

表 15-2 BIND_VARIABLE プロシージャのパラメータ

パラメータ	説明
c	パススルー SQL 文に関連付けられたカーソル。カーソルは、ルーチン OPEN_CURSOR を使用してオープンし、PARSE を使用して解析する必要があります。
pos	SQL 文におけるバインド変数の位置。1 から始まります。
val	バインド変数名に渡す必要がある値。
name	(オプション) バインド変数名。 たとえば、SELECT * FROM emp WHERE ename=:ename では、バインド変数 :ename の位置は 1、名前は :ename です。このパラメータは、非 Oracle システムで位置によるバインドではなく名前によるバインドがサポートされている場合に使用できます。この場合も位置を渡す必要があります。

例外

表 15-3 BIND_VARIABLE プロシージャの例外

例外	説明
ORA-28550	渡されたカーソルが無効です。
ORA-28552	プロシージャが正しい順序で実行されません（最初にカーソルをオープンし、SQL 文を解析する必要があります）。
ORA-28553	バインド変数の位置が有効範囲外です。
ORA-28555	NOT NULL のパラメータに NULL 値が渡されました。

プラグマ

Purity level defined: WNDS, RNDS

BIND_VARIABLE_RAW プロシージャ

このプロシージャは、RAW 型 IN 変数をバインドします。

構文

```
DBMS_HS_PASSTHROUGH.BIND_VARIABLE_RAW (  
  c      IN BINARY_INTEGER NOT NULL,  
  pos    IN BINARY_INTEGER NOT NULL,  
  val    IN RAW,  
  name   IN VARCHAR2);
```

パラメータ

表 15-4 BIND_VARIABLE_RAW プロシージャのパラメータ

パラメータ	説明
c	パススルー SQL 文に関連付けられたカーソル。カーソルは、ルーチン OPEN_CURSOR を使用してオープンし、PARSE を使用して解析する必要があります。
pos	SQL 文におけるバインド変数の位置。1 から始まります。
val	バインド変数に渡す必要がある値。

表 15-4 BIND_VARIABLE_RAW プロシージャのパラメータ

パラメータ	説明
name	(オプション) バインド変数名。 たとえば、SELECT * FROM emp WHERE ename=:ename では、バインド変数 :ename の位置は 1、名前は :ename です。このパラメータは、非 Oracle システムで位置によるバインドではなく名前によるバインドがサポートされている場合に使用できます。この場合も位置を渡す必要があります。

例外

表 15-5 BIND_VARIABLE_RAW プロシージャの例外

例外	説明
ORA-28550	渡されたカーソルが無効です。
ORA-28552	プロシージャが正しい順序で実行されません（最初にカーソルをオープンし、SQL 文を解析する必要があります）。
ORA-28553	バインド変数の位置が有効範囲外です。
ORA-28555	NOT NULL のパラメータに NULL 値が渡されました。

プラグマ

Purity level defined : WNDS, RNDS

BIND_OUT_VARIABLE プロシージャ

このプロシージャは、OUT 変数を PL/SQL プログラム変数にバインドします。

構文

```
DBMS_HS_PASSTHROUGH.BIND_OUT_VARIABLE (  
  c          IN  BINARY_INTEGER NOT NULL,  
  pos        IN  BINARY_INTEGER NOT NULL,  
  val        OUT <dt>,<br>  
  name       IN  VARCHAR2);
```

<dt> は DATE、NUMBER または VARCHAR2 のいずれかです。

関連項目： データ型 RAW の OUT 変数のバインド方法は、15-7 ページの「[BIND_OUT_VARIABLE_RAW プロシージャ](#)」を参照してください。

パラメータ

表 15-6 BIND_OUT_VARIABLE プロシージャのパラメータ

パラメータ	説明
c	パススルー SQL 文に関連付けられたカーソル。カーソルは、ルーチン OPEN_CURSOR を使用してオープンし、PARSE を使用して解析する必要があります。
pos	SQL 文におけるバインド変数の位置。1 から始まります。
val	OUT バインド変数とその値を格納する変数。パッケージは、変数のサイズのみ記憶しています。SQL 文が実行された後、GET_VALUE を使用して OUT パラメータの値を取り出すことができます。取り出される値のサイズが、BIND_OUT_VARIABLE を使用して渡したパラメータのサイズを超えないようにしてください。
name	(オプション) バインド変数名。 たとえば、SELECT * FROM emp WHERE ename=:ename では、バインド変数 :ename の位置は 1、名前は :ename です。このパラメータは、非 Oracle システムで位置によるバインドではなく名前によるバインドがサポートされている場合に使用できます。この場合も位置を渡す必要があります。

例外

表 15-7 BIND_OUT_VARIABLE プロシージャの例外

例外	説明
ORA-28550	渡されたカーソルが無効です。
ORA-28552	プロシージャが正しい順序で実行されません（最初にカーソルをオープンし、SQL 文を解析する必要があります）。
ORA-28553	バインド変数の位置が有効範囲外です。
ORA-28555	NOT NULL のパラメータに NULL 値が渡されました。

プラグマ

Purity level defined : WNDS, RNDS

BIND_OUT_VARIABLE_RAW プロシージャ

このプロシージャは、データ型 RAW の OUT 変数を PL/SQL プログラム変数にバインドします。

構文

```
DBMS_HS_PASSTHROUGH.BIND_OUT_VARIABLE (  
  c          IN  BINARY_INTEGER NOT NULL,  
  pos        IN  BINARY_INTEGER NOT NULL,  
  val        OUT RAW,  
  name       IN  VARCHAR2);
```

パラメータ

表 15-8 BIND_OUT_VARIABLE_RAW プロシージャのパラメータ

パラメータ	説明
c	パススルー SQL 文に関連付けられたカーソル。カーソルは、ルーチン OPEN_CURSOR を使用してオープンし、PARSE を使用して解析する必要があります。
pos	SQL 文におけるバインド変数の位置。1 から始まります。
val	OUT バインド変数とその値を格納する変数。パッケージは、変数のサイズのみ記憶しています。SQL 文が実行された後、GET_VALUE を使用して OUT パラメータの値を取り出すことができます。取り出される値のサイズは、BIND_OUT_VARIABLE_RAW を使用して渡したパラメータのサイズを超えないようにしてください。
name	(オプション) バインド変数名。 たとえば、SELECT * FROM emp WHERE ename=:ename では、バインド変数 :ename の位置は 1、名前は :ename です。このパラメータは、非 Oracle システムで位置によるバインドではなく名前によるバインドがサポートされている場合に使用できます。この場合も位置を渡す必要があります。

例外

表 15-9 BIND_OUT_VARIABLE_RAW プロシージャの例外

例外	説明
ORA-28550	渡されたカーソルが無効です。
ORA-28552	プロシージャが正しい順序で実行されません（最初にカーソルをオープンし、SQL 文を解析する必要があります）。
ORA-28553	バインド変数の位置が有効範囲外です。
ORA-28555	NOT NULL のパラメータに NULL 値が渡されました。

プラグマ

Purity level defined : WNDS, RNDS

BIND_INOUT_VARIABLE プロシージャ

このプロシージャは、IN OUT バインド変数をバインドします。

構文

```
DBMS_HS_PASSTHROUGH.BIND_INOUT_VARIABLE (  
  c      IN      BINARY_INTEGER NOT NULL,  
  pos    IN      BINARY_INTEGER NOT NULL,  
  val    IN OUT  <dt>,  
  name   IN      VARCHAR2);
```

<dt> は DATE、NUMBER または VARCHAR2 のいずれかです。

関連項目： データ型 RAW の IN OUT 変数のバインド方法は、15-9 ページの「[BIND_INOUT_VARIABLE_RAW プロシージャ](#)」を参照してください。

パラメータ

表 15-10 BIND_INOUT_VARIABLE プロシージャのパラメータ

パラメータ	説明
c	パススルー SQL 文に関連付けられたカーソル。カーソルは、ルーチン OPEN_CURSOR を使用してオープンし、PARSE を使用して解析する必要があります。
pos	SQL 文におけるバインド変数の位置。1 から始まります。

表 15-10 BIND_INOUT_VARIABLE プロシージャのパラメータ

パラメータ	説明
val	この値は次の 2 つの目的で使用されます。 - SQL 文が実行される前に IN の値を設定します。 - OUT 値のサイズを判別します。
name	(オプション) バインド変数名。 たとえば、SELECT * FROM emp WHERE ename=:ename では、バインド変数 :ename の位置は 1、名前は :ename です。このパラメータは、非 Oracle システムで位置によるバインドではなく名前によるバインドがサポートされている場合に使用できます。この場合も位置を渡す必要があります。

例外

表 15-11 BIND_INOUT_VARIABLE プロシージャの例外

例外	説明
ORA-28550	渡されたカーソルが無効です。
ORA-28552	プロシージャが正しい順序で実行されません（最初にカーソルをオープンし、SQL 文を解析する必要があります）。
ORA-28553	バインド変数の位置が有効範囲外です。
ORA-28555	NOT NULL のパラメータに NULL 値が渡されました。

プログラマ

Purity level defined : WNDS, RNDS

BIND_INOUT_VARIABLE_RAW プロシージャ

このプロシージャは、データ型 RAW の IN OUT 変数をバインドします。

構文

```
DBMS_HS_PASSTHROUGH.BIND_INOUT_VARIABLE (  
  c          IN      BINARY_INTEGER NOT NULL,  
  pos        IN      BINARY_INTEGER NOT NULL,  
  val        IN OUT RAW,  
  name       IN      VARCHAR2);
```

パラメータ

表 15-12 BIND_INOUT_VARIABLE_RAW プロシージャのパラメータ

パラメータ	説明
c	パススルー SQL 文に関連付けられたカーソル。カーソルは、ルーチン OPEN_CURSOR を使用してオープンし、PARSE を使用して解析する必要があります。
pos	SQL 文におけるバインド変数の位置。1 から始まります。
val	この値は次の 2 つの目的で使用されます。 - SQL 文が実行される前に IN の値を設定します。 - OUT 値のサイズを判別します。
name	(オプション) バインド変数名。 たとえば、SELECT * FROM emp WHERE ename=:ename では、バインド変数 :ename の位置は 1、名前は :ename です。このパラメータは、非 Oracle システムで位置によるバインドではなく名前によるバインドがサポートされている場合に使用できます。この場合も位置を渡す必要があります。

例外

表 15-13 BIND_INOUT_VARIABLE_RAW プロシージャの例外

例外	説明
ORA-28550	渡されたカーソルが無効です。
ORA-28552	プロシージャが正しい順序で実行されません（最初にカーソルをオープンし、SQL 文を解析する必要があります）。
ORA-28553	バインド変数の位置が有効範囲外です。
ORA-28555	NOT NULL のパラメータに NULL 値が渡されました。

プラグマ

Purity level defined : WNDS, RNDS

CLOSE_CURSOR プロシージャ

このファンクションは、SQL 文が非 Oracle システムで実行された後に、カーソルをクローズし、関連メモリーを解放します。カーソルがオープンされていない場合、操作は何も行われません。

構文

```
DBMS_HS_PASSTHROUGH.CLOSE_CURSOR (  
    c IN BINARY_INTEGER NOT NULL);
```

パラメータ

表 15-14 CLOSE_CURSOR プロシージャのパラメータ

パラメータ	説明
c	解放するカーソル

例外

表 15-15 CLOSE_CURSOR プロシージャの例外

例外	説明
ORA-28555	NOT NULL のパラメータに NULL 値が渡されました。

プラグマ

```
Purity level defined : WNDS, RNDS
```

EXECUTE_IMMEDIATE プロシージャ

このファンクションは、SQL 文を即時実行します。有効な SQL コマンド（SELECT を除く）をすぐに実行できます。文にバインド変数を含めることはできません。文は引数で VARCHAR2 として渡されます。SQL 文は、内部的には、OPEN_CURSOR、PARSE、EXECUTE_NON_QUERY、CLOSE_CURSOR のパススルー SQL プロトコル・シーケンスを使用して実行されます。

構文

```
DBMS_HS_PASSTHROUGH.EXECUTE_IMMEDIATE (  
    S IN VARCHAR2 NOT NULL)  
RETURN BINARY_INTEGER;
```

パラメータ

表 15-16 EXECUTE_IMMEDIATE プロシージャのパラメータ

パラメータ	説明
s	即時実行対象の文を含む VARCHAR2 変数

戻り値

SQL 文の実行によって影響を受ける行数が戻されます。

例外

表 15-17 EXECUTE_IMMEDIATE プロシージャの例外

例外	説明
ORA-28551	SQL 文が正しくありません。
ORA-28544	オープンしているカーソルが最大数に達しました。
ORA-28555	NOT NULL のパラメータに NULL 値が渡されました。

プラグマ

なし

EXECUTE_NON_QUERY ファンクション

このファンクションは SQL 文を実行します。SQL 文に SELECT 文は使用できません。SQL 文を実行する前に、カーソルをオープンし、SQL 文を解析する必要があります。

構文

```
DBMS_HS_PASSTHROUGH.EXECUTE_NON_QUERY (  
  c IN BINARY_INTEGER NOT NULL)  
  RETURN BINARY_INTEGER;
```


パラメータ

表 15-18 EXECUTE_NON_QUERY ファンクションのパラメータ

パラメータ	説明
c	パススルー SQL 文に関連付けられたカーソル。カーソルは、ルーチン OPEN_CURSOR を使用してオープンし、PARSE を使用して解析する必要があります。

戻り値

非 Oracle システムで SQL 文によって影響を受ける行数が戻されます。

例外

表 15-19 EXECUTE_NON_QUERY プロシージャの例外

例外	説明
ORA-28550	渡されたカーソルが無効です。
ORA-28552	BIND_VARIABLE プロシージャが正しい順序で実行されません (最初にカーソルをオープンし、SQL 文を解析する必要があります)。
ORA-28555	NOT NULL のパラメータに NULL 値が渡されました。

プラグマ

なし

FETCH_ROW ファンクション

このファンクションは、結果セットから行をフェッチします。結果セットは、SQL SELECT 文で定義されます。それ以上フェッチする行がなくなると、例外 NO_DATA_FOUND が発生します。行をフェッチする前に、カーソルをオープンし、SQL 文を解析する必要があります。

構文

```
DBMS_HS_PASSTHROUGH.FETCH_ROW (  
    c          IN BINARY_INTEGER NOT NULL,  
    first      IN BOOLEAN)  
RETURN BINARY_INTEGER;
```

パラメータ

表 15-20 FETCH_ROW ファンクションのパラメータ

パラメータ	説明
c	パススルー SQL 文に関連付けられたカーソル。カーソルは、ルーチン OPEN_CURSOR を使用してオープンし、PARSE を使用して解析する必要があります。
first	(オプション) SELECT 文を再実行します。設定可能な値は次のとおりです。 - TRUE: SELECT 文を再実行します。 - FALSE: 次の行をフェッチするか、または初めて実行された場合は、SELECT 文を実行して行をフェッチします (デフォルト)。

戻り値

フェッチされた行数が戻されます。最終行がすでにフェッチされた場合は、0 (ゼロ) が戻されます。

例外

表 15-21 FETCH_ROW プロシージャの例外

例外	説明
ORA-28550	渡されたカーソルが無効です。
ORA-28552	プロシージャが正しい順序で実行されません (最初にカーソルをオープンし、SQL 文を解析する必要があります)。
ORA-28555	NOT NULL のパラメータに NULL 値が渡されました。

プラグマ

Purity level defined : WINDS

GET_VALUE プロシージャ

このプロシージャには 2 つの目的があります。

- 行がフェッチされた後に、SELECT 文の選択リスト項目を取り出します。
- SQL 文が実行された後に、OUT バインド値を取り出します。

構文

```
DBMS_HS_PASSTHROUGH.GET_VALUE (  
    c      IN  BINARY_INTEGER NOT NULL,  
    pos    IN  BINARY_INTEGER NOT NULL,  
    val    OUT <dt>);
```

<dt> は DATE、NUMBER または VARCHAR2 のいずれかです。

関連項目： データ型 RAW の値の取出し方法は、15-16 ページの「[GET_VALUE_RAW プロシージャ](#)」を参照してください。

パラメータ

表 15-22 GET_VALUE プロシージャのパラメータ

パラメータ	説明
c	パススルー SQL 文に関連付けられたカーソル。カーソルは、ルーチン OPEN_CURSOR を使用してオープンし、PARSE を使用して解析する必要があります。
pos	SQL 文におけるバインド変数または選択リスト項目の位置。1 から始まります。
val	OUT バインド変数または選択リスト項目がその値を格納する変数。

例外

表 15-23 GET_VALUE プロシージャの例外

例外	説明
ORA-1403	最終行がフェッチされた後に、GET_VALUE を実行すると NO_DATA_FOUND 例外が戻されます（つまり、FETCH_ROW によって 0（ゼロ）が戻されます）。
ORA-28550	渡されたカーソルが無効です。
ORA-28552	プロシージャが正しい順序で実行されません（最初にカーソルをオープンし、SQL 文を解析する必要があります）。
ORA-28553	バインド変数の位置が有効範囲外です。
ORA-28555	NOT NULL のパラメータに NULL 値が渡されました。

プラグマ

Purity level defined : WND5

GET_VALUE_RAW プロシージャ

このプロシージャは、データ型 RAW が対象である以外は GET_VALUE と同じです。

構文

```
DBMS_HS_PASSTHROUGH.GET_VALUE_RAW (  
  c      IN  BINARY_INTEGER NOT NULL,  
  pos    IN  BINARY_INTEGER NOT NULL,  
  val    OUT RAW);
```

パラメータ

表 15-24 GET_VALUE_RAW プロシージャのパラメータ

パラメータ	説明
c	パススルー SQL 文に関連付けられたカーソル。カーソルは、ルーチン OPEN_CURSOR を使用してオープンし、PARSE を使用して解析する必要があります。
pos	SQL 文におけるバインド変数または選択リスト項目の位置。1 から始まります。
val	OUTバインド変数または選択リスト項目がその値を格納する変数。

例外

表 15-25 GET_VALUE_RAW プロシージャの例外

例外	説明
ORA-1403	最終行がフェッチされた後に、GET_VALUE を実行すると NO_DATA_FOUND 例外が戻されます（つまり、FETCH_ROW によって 0（ゼロ）が戻されます）。
ORA-28550	渡されたカーソルが無効です。
ORA-28552	プロシージャが正しい順序で実行されません（最初にカーソルをオープンし、SQL 文を解析する必要があります）。
ORA-28553	バインド変数の位置が有効範囲外です。
ORA-28555	NOT NULL のパラメータに NULL 値が渡されました。

プラグマ

Purity level defined : WNDS

OPEN_CURSOR ファンクション

このファンクションは、非 Oracle システムでパススルー SQL 文を実行するためのカーソルをオープンします。このファンクションは、すべてのタイプの SQL 文に対してコールする必要があります。

後続のコールで使用する必要があるカーソルが戻されます。このコールによってメモリーが割り当てられます。関連付けられたメモリーの割当てを解除するには、プロシージャ CLOSE_CURSOR をコールします。

構文

```
DBMS_HS_PASSTHROUGH.OPEN_CURSOR
RETURN BINARY_INTEGER;
```

戻り値

後続のプロシージャおよびファンクション・コールで使用されるカーソルが戻されます。

例外

表 15-26 OPEN_CURSOR ファンクションの例外

例外	説明
ORA-28554	カーソルの最大オープン数を超えました。異機種間サービスの OPEN_CURSORS 初期化パラメータの値を増やしてください。

プラグマ

Purity level defined : WNDS, RNDS

PARSE プロシージャ

このプロシージャは、非 Oracle システムで SQL 文を解析します。

構文

```
DBMS_HS_PASSTHROUGH.GET_VALUE_RAW (
c      IN BINARY_INTEGER NOT NULL,
stmt   IN VARCHAR2      NOT NULL);
```

パラメータ

表 15-27 PARSE プロシージャのパラメータ

パラメータ	説明
c	パススルー SQL 文に関連付けられたカーソル。ファンクション OPEN_CURSOR を使用してオープンする必要があります。
stmt	解析する文。

例外

表 15-28 GET_VALUE プロシージャの例外

例外	説明
ORA-28550	渡されたカーソルが無効です。
ORA-28551	SQL 文が正しくありません。
ORA-28555	NOT NULL のパラメータに NULL 値が渡されました。

プラグマ

Purity level defined : WNDS, RNDS

16

DBMS_IOT

DBMS_IOT パッケージは、索引構成表に対する連鎖行への参照を ANALYZE コマンドを使用して格納できる表を作成します。また、enable_constraint 操作時に、索引構成表の中で制約に違反する行を格納できる例外表も作成できます。

サブプログラムの要約

表 16-1 DBMS_IOT パッケージのサブプログラム

サブプログラム	説明
16-2 ページの BUILD_CHAIN_ROWS_TABLE プロシージャ	索引構成表に対する連鎖行への参照を ANALYZE コマンドを使用して格納できる表を作成します。
16-3 ページの BUILD_EXCEPTIONS_TABLE プロシージャ	enable_constraint 操作時に、索引構成表の中で制約に違反する行を格納できる例外表を作成します。

BUILD_CHAIN_ROWS_TABLE プロシージャ

BUILD_CHAIN_ROWS_TABLE プロシージャは、索引構成表に対する連鎖行への参照を ANALYZE コマンドを使用して格納できる表を作成します。

構文

```
DBMS_IOT.BUILD_CHAIN_ROWS_TABLE (  
    owner          IN VARCHAR2,  
    iot_name       IN VARCHAR2,  
    chainrow_table_name IN VARCHAR2 default 'IOT_CHAINED_ROWS');
```

パラメータ

表 16-2 BUILD_CHAIN_ROWS_TABLE プロシージャのパラメータ

パラメータ	説明
owner	索引構成表の所有者
iot_name	索引構成表名
chainrow_table_name	連鎖行表の目的名

例

```
CREATE TABLE l(a char(16),b char(16), c char(16), d char(240),  
PRIMARY KEY(a,b,c)) ORGANIZATION INDEX pctthreshold 10 overflow;  
EXECUTE DBMS_IOT.BUILD_CHAIN_ROWS_TABLE('SYS','L','LC');
```


次の列を含んだ連鎖行表が作成されます。

Column Name	Null?	Type
OWNER_NAME		VARCHAR2 (30)
TABLE_NAME		VARCHAR2 (30)
CLUSTER_NAME		VARCHAR2 (30)
PARTITION_NAME		VARCHAR2 (30)
SUBPARTITION_NAME		VARCHAR2 (30)
HEAD_ROWID		ROWID
TIMESTAMP		DATE
A		CHAR (16)
B		CHAR (16)
C		CHAR (16)

BUILD_EXCEPTIONS_TABLE プロシージャ

BUILD_EXCEPTIONS_TABLE プロシージャは、enable_constraint 操作時に、索引構成表の中で制約に違反する行を格納できる例外表を作成します。

各索引構成表について、その主キーに対応する別々の連鎖行表と例外表を作成する必要があります。

注意： このフォームの連鎖行表と例外表は、Oracle8、リリース 8.0 互換で稼動しているサーバーにのみ必要です。

構文

```
DBMS_IOT.BUILD_EXCEPTIONS_TABLE (  
    owner          IN VARCHAR2,  
    iot_name        IN VARCHAR2,  
    exceptions_table_name IN VARCHAR2 default 'IOT_EXCEPTIONS');
```

パラメータ

表 16-3 BUILD_EXCEPTIONS_TABLE プロシージャのパラメータ

パラメータ	説明
owner	索引構成表の所有者
iot_name	索引構成表名
exceptions_table_name	例外表の目的名

例

```
EXECUTE DBMS_IOT.BUILD_EXCEPTIONS_TABLE('SYS','L','LE');
```

前述の索引構成表に対する例外表の列は、次のようになります。

Column Name	Null?	Type
-----	-----	-----
ROW_ID		VARCHAR2(30)
OWNER		VARCHAR2(30)
TABLE_NAME		VARCHAR2(30)
CONSTRAINT		VARCHAR2(30)
A		CHAR(16)
B		CHAR(16)
C		CHAR(16)

DBMS_JOB サブプログラムは、ジョブ・キュー内のジョブをスケジュールおよび管理します。

関連項目： DBMS_JOB パッケージとジョブ・キューの詳細は、『Oracle8i 管理者ガイド』を参照してください。

要件

ジョブに関連付けられているデータベース権限はありません。DBMS_JOB では、ユーザーのジョブ以外のジョブにアクセスすることはできません。

サブプログラムの要約

表 17-1 DBMS_JOB パッケージのサブプログラム

サブプログラム	説明
17-3 ページの SUBMIT プロシージャ	新規ジョブをジョブ・キューに送信します。
17-4 ページの REMOVE プロシージャ	指定したジョブをジョブ・キューから削除します。
17-5 ページの CHANGE プロシージャ	ジョブに関連付けられているユーザー定義パラメータを変更します。
17-6 ページの WHAT プロシージャ	指定したジョブに関するジョブの記述を変更します。
17-7 ページの NEXT_DATE プロシージャ	指定したジョブの次の実行時間を変更します。
17-7 ページの INSTANCE プロシージャ	インスタンスでジョブが実行されるように割り当てます。
17-8 ページの INTERVAL プロシージャ	指定したジョブの実行間隔を変更します。
17-9 ページの BROKEN プロシージャ	ジョブの実行を禁止します。
17-9 ページの RUN プロシージャ	指定したジョブを強制的に実行します。
17-10 ページの USER_EXPORT プロシージャ	指定したジョブをエクスポート用に再作成します。
17-11 ページの USER_EXPORT プロシージャ	指定したジョブをインスタンス親和性付きでエクスポート用に再作成します。

SUBMIT プロシージャ

このプロシージャは新規ジョブを送信します。順序 sys.jobseq からジョブを選択します。

構文

```
DBMS_JOB.SUBMIT (  
    job          OUT BINARY_INTEGER,  
    what         IN  VARCHAR2,  
    next_date    IN  DATE DEFAULT sysdate,  
    interval     IN  VARCHAR2 DEFAULT 'null',  
    no_parse     IN  BOOLEAN DEFAULT FALSE,  
    instance     IN  BINARY_INTEGER DEFAULT any_instance,  
    force        IN  BOOLEAN DEFAULT FALSE);
```

パラメータ

表 17-2 SUBMIT プロシージャのパラメータ

パラメータ	説明
job	実行するジョブの番号。
what	実行する PL/SQL プロシージャ。
next_date	ジョブを次回実行する日付。
interval	ジョブを次回実行する時間を計算する日付ファンクション。デフォルトは NULL です。このファンクションは、将来の日時または NULL に設定される必要があります。
no_parse	フラグ。デフォルトは FALSE です。FALSE に設定すると、ジョブに関連付けられているプロシージャが解析されます。TRUE に設定すると、ジョブに関連付けられているプロシージャがそのジョブの初回実行時に解析されます。 たとえば、ジョブに関連付けられている表を作成する前にそのジョブを送信する場合は、この値を TRUE に設定します。
instance	ジョブが送信されたときに、そのジョブを実行できるインスタンスを指定します。
force	TRUE の場合は、ジョブ・インスタンスとして正の整数がすべて受け入れられます。FALSE (デフォルト) の場合は、指定したインスタンスで実行する必要がある、そうでない場合は、例外が発生します。

使用上の注意

パラメータ `instance` と `force` がジョブ・キュー親和性のために追加されています。ジョブ・キュー親和性によって、ユーザーは、送信されたジョブを特定のインスタンスで実行するか、またはどのインスタンスでも実行可能とするかを指示できます。

例

新規ジョブをジョブ・キューに送信する例です。このジョブは、プロシージャ `DBMS_DDL.ANALYZE_OBJECT` をコールし、表 `DQUON.ACCOUNTS` に関するオプティマイザの統計情報を生成します。統計情報は、`ACCOUNTS` 表にある行の半分をサンプルとして使用します。このジョブは 24 時間ごとに実行されます。

```
VARIABLE jobno number;
BEGIN
    DBMS_JOB.SUBMIT(:jobno,
        'dms_ddl.analyze_object(''TABLE'',
        ''DQUON'', ''ACCOUNTS'',
        ''ESTIMATE'', NULL, 50);'
        SYSDATE, 'SYSDATE + 1');
    commit;
END;
/
Statement processed.
print jobno
JOBNO
-----
14144
```

REMOVE プロシージャ

このプロシージャは、ジョブ・キューから既存のジョブを削除します。実行中のジョブを停止する機能は、現在はありません。

構文

```
DBMS_JOB.REMOVE (
    job          IN BINARY_INTEGER );
```

パラメータ

表 17-3 REMOVE プロシージャのパラメータ

パラメータ	説明
job	実行するジョブの番号

例

```
EXECUTE DBMS_JOB.REMOVE(14144);
```

CHANGE プロシージャ

このプロシージャは、ジョブ内のユーザー設定可能フィールドを変更します。

構文

```
DBMS_JOB.CHANGE (  
  job      IN  BINARY_INTEGER,  
  what     IN  VARCHAR2,  
  next_date IN  DATE,  
  interval IN  VARCHAR2,  
  instance IN  BINARY_INTEGER DEFAULT NULL,  
  force    IN  BOOLEAN DEFAULT FALSE);
```

パラメータ

表 17-4 CHANGE プロシージャのパラメータ

パラメータ	説明
job	実行するジョブの番号。
what	実行する PL/SQL プロシージャ。
next_date	次のリフレッシュ日付。
interval	日付ファンクション。ジョブ実行の直前に評価されます。
instance	ジョブが送信されたときに、そのジョブを実行できるインスタンスを指定します。デフォルトは NULL で、インスタンス親和性を変更されないことを示します。

表 17-4 CHANGE プロシージャのパラメータ

パラメータ	説明
force	FALSE の場合、（インスタンス番号を変更する対象の）指定インスタンスで実行する必要があります。そうでない場合は、例外が発生します。 TRUE の場合は、ジョブ・インスタンスとして正の整数がすべて受け入れられます。

使用上の注意

パラメータ instance と force がジョブ・キュー親和性のために追加されています。ジョブ・キュー親和性によって、ユーザーは、送信されたジョブを特定のインスタンスで実行するか、またはどのインスタンスでも実行可能とするかを指示できます。

パラメータ what、next_date または interval が NULL の場合、現在の値は変更されません。

例

```
EXECUTE DBMS_JOB.CHANGE(14144, null, null, 'sysdate+3');
```

WHAT プロシージャ

このプロシージャは、既存のジョブが実行する内容を変更し、その環境を置き換えます。

構文

```
DBMS_JOB.WHAT (  
    job          IN  BINARY_INTEGER,  
    what         IN  VARCHAR2);
```

パラメータ

表 17-5 WHAT プロシージャのパラメータ

パラメータ	説明
job	実行するジョブの番号
what	実行する PL/SQL プロシージャ

次に正しい what の値の例（ルーチンが存在していると仮定）を示します。

- 'myproc('10-JAN-82'', next_date, broken);'
- 'scott.emppackage.give_raise('JENKINS', 30000.00);'
- 'dbms_job.remove(job);'

NEXT_DATE プロシージャ

このプロシージャは、既存のジョブの次回実行時間を変更します。

構文

```
DBMS_JOB.NEXT_DATE (  
    job          IN BINARY_INTEGER,  
    next_date IN DATE);
```

パラメータ

表 17-6 NEXT_DATE プロシージャのパラメータ

パラメータ	説明
job	実行するジョブの番号。
next_date	次のリフレッシュ日付。ジョブはこの日付に自動的に実行されます。ただし、ジョブを実行するバックグラウンド・プロセスがあることが前提です。

INSTANCE プロシージャ

このプロシージャは、ジョブ・インスタンス親和性を変更します。

構文

```
DBMS_JOB.INSTANCE (  
    job          IN BINARY_INTEGER,  
    instance     IN BINARY_INTEGER,  
    force        IN BOOLEAN DEFAULT FALSE);
```

パラメータ

表 17-7 INSTANCE プロシージャのパラメータ

パラメータ	説明
job	実行するジョブの番号。
instance	ジョブを送信するときに、ユーザーはジョブを実行できるインスタンスを指定できます。
force	TRUE の場合は、ジョブ・インスタンスとして正の整数がすべて受け入れられます。FALSE（デフォルト）の場合は、指定したインスタンスで実行する必要がある、そうでない場合は、例外が発生します。

INTERVAL プロシージャ

このプロシージャは、ジョブの実行間隔を変更します。

構文

```
DBMS_JOB.INTERVAL (  
    job      IN  BINARY_INTEGER,  
    interval IN  VARCHAR2);
```

パラメータ

表 17-8 INTERVAL プロシージャのパラメータ

パラメータ	説明
job	実行するジョブの番号。
interval	日付ファンクション。ジョブの実行直前に評価されます。

使用上の注意

ジョブが正常に完了すると、next_date にこの新しい日付が設定されます。interval は、その日付を select interval into next_date from dual 文に接続することによって評価されます。

interval パラメータの評価結果は将来の日時に設定される必要があります。有効な間隔には次の例があります。

'sysdate + 7'	毎週 1 回実行
'next_day(sysdate, ''TUESDAY'')'	毎週火曜日に実行
'null'	1 回のみ実行

interval の評価結果が NULL で、ジョブが正常に完了した場合、そのジョブはキューから自動的に削除されます。

BROKEN プロシージャ

このプロシージャは中断フラグを設定します。中断状態のジョブはその後実行されません。

構文

```
DBMS_JOB.BROKEN (  
  job          IN  BINARY_INTEGER,  
  broken       IN  BOOLEAN,  
  next_date    IN  DATE DEFAULT SYSDATE);
```

パラメータ

表 17-9 Broken プロシージャのパラメータ

パラメータ	説明
job	実行するジョブの番号。
broken	ジョブ中断: IN の値は FALSE です。
next_data	次のリフレッシュ日付。

注意: 実行中のジョブに中断を指定すると、ジョブの完了後、ジョブのステータスは normal にリセットされます。したがって、このプロシージャは、実行中でないジョブに対してのみ実行してください。

RUN プロシージャ

このプロシージャは、ジョブ JOB をすぐに実行します。そのジョブが中断されている場合でも実行します。

ジョブが実行されると、next_date が再計算されます。ビュー user_jobs を参照してください。

構文

```
DEMS_JOB.RUN (
  job      IN  BINARY_INTEGER,
  force     IN  BOOLEAN DEFAULT FALSE);
```

パラメータ

表 17-10 Run プロシージャのパラメータ

パラメータ	説明
job	実行するジョブの番号。
force	TRUE の場合、フォアグラウンド・プロセスでのジョブの実行にインスタンス親和性は無関係です。FALSE の場合は、指定したインスタンスでのみフォアグラウンドでジョブを実行できます。

例

```
EXECUTE DEMS_JOB.RUN(14144);
```

注意： これは、現行セッションのパッケージを再初期化します。

例外

force が FALSE で、接続インスタンスが正しくない場合は、例外が発生します。

USER_EXPORT プロシージャ

このプロシージャは、指定したジョブを再作成するコールのテキストを生成します。

構文

```
DEMS_JOB.USER_EXPORT (
  job      IN  BINARY_INTEGER,
  mycall   IN OUT VARCHAR2);
```

パラメータ

表 17-11 USER_EXPORT プロシージャのパラメータ

パラメータ	説明
job	実行するジョブの番号
mycall	指定したジョブを再作成するコールのテキスト

USER_EXPORT プロシージャ

このプロシージャは、インスタンス親和性（8i 以上）を変更し、互換性を保ちます。

構文

```
DBMS_JOB.USER_EXPORT (  
    job      IN      BINARY_INTEGER,  
    mycall   IN OUT  VARCHAR2,  
    myinst   IN OUT  VARCHAR2);
```

パラメータ

表 17-12 USER_EXPORT プロシージャのパラメータ

パラメータ	説明
job	実行するジョブの番号
mycall	指定したジョブを再作成するコールのテキスト
myinst	インスタンス親和性を変更するコールのテキスト

Oracle Parallel Server での DBMS_JOB パッケージの使用方法

この例では、DBMS_JOB 内の定数は、ジョブとインスタンスの間で "マッピングなし" を示します。つまり、いずれのインスタンスでもジョブを実行できます。

DBMS_JOB.SUBMIT ジョブをジョブ・キューに送るには、次の構文を使用します。

```
DBMS_JOB.SUBMIT( JOB OUT BINARY_INTEGER,  
WHAT IN VARCHAR2, NEXT_DATE IN DATE DEFAULTSYSDATE,  
INTERVAL IN VARCHAR2 DEFAULT 'NULL',  
NO_PARSE IN BOOLEAN DEFAULT FALSE,  
INSTANCE IN BINARY_INTEGER DEFAULT ANY_INSTANCE,  
FORCE IN BOOLEAN DEFAULT FALSE)
```

パラメータ INSTANCE と FORCE を使用して、ジョブとインスタンスの親和性を制御します。INSTANCE のデフォルト値は 0（ゼロ）で、いずれのインスタンスでもジョブを実行できることを示します。特定のインスタンスでジョブを実行するには、INSTANCE の値を指定します。INSTANCE の値が負数または NULL の場合は、エラー ORA-23319 が表示されます。

FORCE パラメータは FALSE にデフォルト設定されます。TRUE の場合は、ジョブ・インスタンスとして正の整数がすべて受け入れられます。FORCE が FALSE の場合は、指定したインスタンスが実行されている必要があります。実行されていない場合は、エラー番号 ORA-23428 が表示されます。

DBMS_JOB.INSTANCE ジョブを実行するために特定のインスタンスを割り当てるには、次の構文を使用します。

```
DBMS_JOB.INSTANCE( JOB IN BINARY_INTEGER,  
INSTANCE IN BINARY_INTEGER,  
FORCE IN BOOLEAN DEFAULT FALSE)
```

この例の FORCE パラメータは FALSE にデフォルト設定されます。INSTANCE の値が 0（ゼロ）の場合は、ジョブ親和性が変更され、FORCE の値に関係なく、使用可能なすべてのインスタンスがジョブを実行できます。INSTANCE の値が正数で FORCE パラメータが FALSE の場合、ジョブ親和性は、指定したインスタンスが実行されている場合のみ変更されます。実行されていない場合は、エラー ORA-23428 が表示されます。

FORCE パラメータが TRUE の場合は、ジョブ・インスタンスとして正の整数がすべて受け入れられ、ジョブ親和性が変更されます。INSTANCE の値が負数または NULL の場合は、エラー ORA-23319 が表示されます。

DBMS_JOB.CHANGE ジョブに関連付けられたユーザー定義可能なパラメータを変更するには、次の構文を使用します。

```
DBMS_JOB.CHANGE( JOB IN BINARY_INTEGER,  
WHAT IN VARCHAR2 DEFAULT NULL,  
NEXT_DATE IN DATE DEFAULT NULL,  
INTERVAL IN VARCHAR2 DEFAULT NULL,
```

```
INSTANCE IN BINARY_INTEGER DEFAULT NULL,  
FORCE IN BOOLEAN DEFAULT FALSE )
```

この例では、INSTANCE と FORCE の 2 つのパラメータが使用されています。INSTANCE のデフォルト値は NULL で、ジョブ親和性を変更しないことを示しています。

FORCE のデフォルト値は FALSE です。指定したインスタンスが実行されていない場合はエラー ORA-23428 が、INSTANCE の数値が負数の場合はエラー ORA-23319 が表示されます。

DBMS_JOB.RUN DBMS_JOB.RUN の FORCE パラメータは FALSE にデフォルト設定されます。FORCE が TRUE の場合は、フォアグラウンド・プロセスでのジョブ実行にインスタンス親和性は無関係です。FORCE が FALSE の場合は、指定したインスタンスでのみフォアグラウンドでジョブを実行できます。FORCE が FALSE で、接続先のインスタンスが不適切なインスタンスの場合は、エラー ORA-23428 が表示されます。

```
DBMS_JOB.RUN( JOB IN BINARY_INTEGER,  
FORCE IN BOOLEAN DEFAULT FALSE)
```

関連項目： Oracle Parallel Server の詳細は、『Oracle8i Parallel Server 概要』および『Oracle8i Parallel Server 管理、配置およびパフォーマンス』を参照してください。

DBMS_LOB パッケージは、BLOB、CLOB、NCLOB、BFILE およびテンポラリ LOB を操作するサブプログラムを提供します。DBMS_LOB を使用すると、各 LOB の特定の部分または LOB 全体に対するアクセスおよび操作ができます。

DBMS_LOB では、BLOB、CLOB および NCLOB の読み込みと変更ができます。BFILE に対しては、読み取り専用操作を実行できます。LOB 操作の大部分が、このパッケージによって提供されます。

関連項目：『Oracle8i アプリケーション開発者ガイド ラージ・オブジェクト』

要件

このパッケージは、SYS (connect internal) によって作成される必要があります。このパッケージが提供する操作は、パッケージ所有者 SYS ではなく、現行のコール・ユーザーのもとで実行されます。

LOB ロケータ

すべての DBMS_LOB サブプログラムは、LOB ロケータをベースにして動作します。DBMS_LOB サブプログラムを正常に実行するためには、データベース表領域または外部ファイル・システムにすでに存在している LOB を示す入力ロケータを用意する必要があります。
『[Oracle8i アプリケーション開発者ガイド ラージ・オブジェクト](#)』の第 1 章も参照してください。

内部 LOB 内部 LOB の場合は、最初に SQL データ定義言語 (DDL) を使用して LOB 列を含んだ表を定義し、次に SQL データ操作言語 (DML) を使用して、ロケータを初期化するかまたはこの LOB 列に移入する必要があります。

外部 LOB 外部 LOB の場合は、有効な物理ディレクトリが定義済であることを示す DIRECTORY オブジェクトおよび Oracle に対する読み込み許可を伴った物理ファイルの存在を確認する必要があります。オペレーティング・システムでパス名の太文字と小文字が区別される場合は、必ず正しい形式でディレクトリを指定してください。

LOB の定義と作成後に、SELECT を実行して LOB ロケータをローカルの PL/SQL LOB 変数に割り当て、この変数を DBMS_LOB への入力パラメータとして使用し LOB 値にアクセスすることができます。

テンポラリ LOB テンポラリ LOB の場合は、OCI、PL/SQL またはその他のプログラム・インタフェースを使用して作成または操作する必要があります。テンポラリ LOB は、BLOB、CLOB または NCLOB のいずれにもできます。

データ型

DBMS_LOB サブプログラムに対するパラメータには、次のデータ型が使用されます。

表 18-1 DBMS_LOB のデータ型

BLOB	ソースまたは宛先のバイナリ LOB。
RAW	ソースまたは宛先の RAW バッファ（BLOB とともに使用されます）。
CLOB	ソースまたは宛先の文字 LOB（NCLOB を含みます）。
VARCHAR2	ソースまたは宛先の文字バッファ（CLOB および NCLOB とともに使用されます）。
INTEGER	バッファまたは LOB のサイズ、LOB へのオフセット、またはアクセス量を指定します。
BFILE	データベースの外部に格納されているラージ・バイナリ・オブジェクト。

DBMS_LOB パッケージでは特別な型を定義しません。NCLOB は、CLOB の特別なケースで、固定幅と可変幅のマルチバイト各国キャラクタ・セットに使用されます。CLOB 用の DBMS_LOB サブプログラム仕様部にある句 ANY_CS によって、CLOB または NCLOB ロケータ変数を入力として受け入れることができます。

定数

DBMS_LOB では、次の定数が定義されます。

```
file_readonly CONSTANT BINARY_INTEGER := 0;
lob_readonly  CONSTANT BINARY_INTEGER := 0;
lob_readwrite CONSTANT BINARY_INTEGER := 1;
lobmaxsize    CONSTANT INTEGER         := 4294967295;
call          CONSTANT PLS_INTEGER     := 12;
session       CONSTANT PLS_INTEGER     := 10;
```

Oracle は、最大 4GB (2^{32}) の LOB をサポートします。ただし、パッケージの amount と offset パラメータで設定できるのは、1 ～ 4294967295 ($2^{32}-1$) の範囲の値です。

PL/SQL 3.0 言語では、RAW または VARCHAR2 変数の最大サイズが 32767 バイトに指定されます。

注意： 値 32767 バイトは、以降の項では maxbufsize で表されます。

例外

表 18-2 DBMS_LOB の例外

例外	コード	説明
invalid_argval	21560	引数は NULL 以外の有効な値である必要がありますが、渡された引数値は NULL、無効または範囲外です。
access_error	22925	LOB に書き込むデータが多すぎます。LOB サイズは最大 4GB です。
noexist_directory	22285	ファイルに指定されているディレクトリが存在しません。
nopriv_directory	22286	ユーザーに、そのディレクトリ別名またはファイル（あるいはその両方）の操作に必要なアクセス権限がありません。
invalid_directory	22287	現行の操作に使用しているディレクトリ別名は、それが初めてアクセスされた別名か、または前回のアクセス以降に DBA が変更した別名である場合は無効です。
operation_failed	22288	ファイル操作に失敗しました。
unopened_file	22289	要求された操作の実行に使用するファイルがオープンされていません。
open_toomany	22290	オープン・ファイル数が最大値に達しました。

セキュリティ

無名 PL/SQL ブロックからコールされた DBMS_LOB サブプログラムは、カレント・ユーザーの権限を使用して実行されます。ストアド・プロシージャからコールされた DBMS_LOB サブプログラムは、そのストアド・プロシージャの所有者の権限を使用して実行されます。

Oracle8i では、ユーザーはプロシージャの作成時に、AUTHID を設定して定義者の権限または実行者の権限のどちらを使用するかを示すことができます。たとえば、次のようになります。

```
CREATE PROCEDURE proc1 authid definer ...
```

または

```
CREATE PROCEDURE proc1 authid current_user ....
```

関連項目： AUTHID と権限の詳細は、『Oracle8i PL/SQL ユーザーズ・ガイドおよびリファレンス』を参照してください。

DIRECTORY 機能を使用すると、BFILE に安全にアクセスできます。この機能は、『Oracle8i アプリケーション開発者ガイド ラージ・オブジェクト』と『Oracle8i SQL リファレンス』の「BFILENAME 関数」で説明されています。

規則および制限事項

- このパッケージにあるサブプログラムの仕様部に適用される規則は次のとおりです。
 - BLOB と BFILE を操作するサブプログラムの length と offset パラメータは、バイト単位で指定する必要があります。
 - CLOB を操作するサブプログラムの length と offset パラメータは、文字単位で指定する必要があります。
 - offset と amount パラメータは常に、CLOB/NCLOB の場合は文字単位、BLOB/BFILE の場合はバイト単位です。
- パラメータ値の指定時に、次の制限事項に従わなかった場合（または指定しなかった場合）は、INVALID_ARGVAL 例外が発生します。
 1. LOB データの開始位置からの正の絶対オフセットのみ許可されています。LOB の終了位置からの負のオフセットは許可されていません。
 2. amount、offset、newlen、nth など、サイズや位置を表すパラメータには、0（ゼロ）以外の正の値のみ許可されています。Oracle SQL 文字列ファンクションと演算子では、負のオフセットや範囲は許可されていません。
 3. offset、amount、newlen、nth の値は、どの DBMS_LOB サブプログラムの場合でも、最大値は lobmaxsize (4GB-1) です。
 4. 固定幅のマルチバイト・キャラクタで構成される CLOB の場合、これらのパラメータの最大値は (lobmaxsize/character_width_in_bytes) 文字です。

たとえば、CLOB が次のような 2 バイト文字で構成されているとします。

```
JA16SJISFIXED
```

この場合の amount の最大値は次のとおりです。

```
4294967295/2 = 2147483647 characters.
```

- PL/SQL 言語仕様では、DBMS_LOB サブプログラムで使用される RAW と VARCHAR2 パラメータの上限は 32767 バイト（文字数ではありません）に規定されています。たとえば、変数を次のように宣言するとします。

```
charbuf VARCHAR2(3000)
```

この場合、charbuf にはシングル・バイト文字の場合は 3000 文字、2 バイトの固定幅文字の場合は 1500 文字格納できます。これは、CLOB と NCLOB に対して DBMS_LOB サブプログラムを使用するときに特に注意してください。

- %CHARSET 句は、%CHARSET を使用するパラメータの形式が、参照先である ANY_CS パラメータの形式と一致している必要があることを示します。

たとえば、VARCHAR2 バッファ・パラメータを使用する DBMS_LOB サブプログラムでは、VARCHAR2 バッファの形式は CLOB パラメータの形式と一致している必要があります。

す。入力 LOB パラメータの型が NCLOB の場合、バッファには NCHAR データが含まれている必要があります。これに対して、入力 LOB のパラメータの型が CLOB の場合、バッファには CHAR データが含まれる必要があります。

2 つの CLOB パラメータを使用する DBMS_LOB サブプログラムの場合は、2 つの CLOB パラメータの形式が同じである必要があります。つまり、両方とも NCLOB であるか、または CLOB である必要があります。

- 更新サブプログラム（例：APPEND、COPY、TRIM、WRITE および WRITEAPPEND サブプログラム）のコールで、amount と offset を加算した値が、BLOB と BFILE の場合で 4GB（例：lobmaxsize+1）、CLOB の場合で (lobmaxsize/character_width_in_bytes) +1 を超えると、アクセス例外が発生します。

この入力条件のもとでは、READ、COMPARE、INSTR および SUBSTR などの読み込みサブプログラムでは、End of Lob/File に達するまでデータが読み込まれます。たとえば、BLOB または BFILE での READ 操作で、ユーザーが offset 値に 3GB、amount 値に 2GB を指定すると、READ では (4GB-1)-3GB バイトしか読み込まれません。

- パラメータに NULL または無効な値が入力されると、ファンクションは NULL を戻します。宛先 LOB のパラメータに NULL 値が指定されると、プロシージャでは例外が発生します。
- COMPARE、INSTR および SUBSTR など、パラメータとしてパターンが含まれている操作では、pattern パラメータまたは副文字列に、標準の式または特殊一致文字（例：SQL の LIKE 演算子の %）はサポートされていません。
- End Of LOB 状態は、READ プロシージャで NO_DATA_FOUND 例外を使用して示されます。この例外が発生するのは、ユーザーが LOB/FILE の終了位置を超えてデータを読み込もうとしたときのみです。最後に読み込んだ READ バッファは 0（ゼロ）バイトです。
- LOB 更新の一貫性を保つために、LOB データを変更するプロシージャ（ミューテータ）をコールする前に、宛先 LOB が含まれている行をロックする必要があります。
- 特に注記がない限り、offset パラメータのデフォルト値は 1 です。これは、BLOB または BFILE データの最初のバイト、および CLOB または NCLOB 値の最初の文字を示します。amount パラメータはデフォルト値が指定されていないため、値を明示的に入力する必要があります。
- LOB を変更する任意のサブプログラム（例：APPEND、COPY、ERASE、TRIM または WRITE）をコールする前に、宛先の内部 LOB が含まれている行をロックする必要があります。これらのサブプログラムでは、LOB を含んだ行のロックは暗黙的には行われません。

BFILE 特有の規則および制限事項

- サブプログラム COMPARE、INSTR、READ、SUBSTR、FILECLOSE、FILECLOSEALL および LOADFROMFILE は、オープン済の BFILE ロケータでのみ動作します。つまり、これらのサブプログラムのコールの前に、FILEOPEN コールが正常に完了している必要があります。
- ファンクション FILEEXISTS、FILEGETNAME および GETLENGTH に関しては、ファイルのオープン/クローズ状態はあまり重要ではありません。ただし、ファイルが物理的に必ず存在し、ユーザーに DIRECTORY オブジェクトとそのファイルに関する適切な権限があることが必要です。
- DBMS_LOB は、BFILE 操作に対する同時実行制御メカニズムをサポートしません。
- クローズ処理が正しく行われていない複数のオープン・ファイルがセッションで使用されている場合は、FILECLOSEALL サブプログラムを使用してセッションでオープンされたファイルをすべてクローズし、ファイル操作を最初からやり直すことができます。
- DIRECTORY の作成者である場合、またはシステム権限がある場合、SQL の CREATE OR REPLACE、DROP および REVOKE 文の使用には特に注意してください。

ユーザーまたは特定のディレクトリ・オブジェクトの権限受領者がセッション内に複数のファイルをオープンしている場合は、前述のコマンドがファイル操作に誤って影響を与える場合があります。この状況で異常終了した場合は、必ず FILECLOSEALL をコールするプログラムまたは無名ブロックを起動し、ファイルを再度オープンしてファイル操作を再開してください。

- ユーザー・セッションの間にオープンされたファイルはすべて、セッション終了時に自動的にクローズされます。ただし、BFILE の操作の正常終了と異常終了いずれの場合も、操作終了後にファイルをクローズすることをお勧めします。

プログラムが正常に終了した場合、ファイルを適切にクローズすることによって、セッションで同時にオープンしているファイル数は、必ず SESSION_MAX_OPEN_FILES 未満になります。

PL/SQL プログラムが異常終了した場合は、その PL/SQL プログラム内でオープン中のすべてのファイルをクローズする例外ハンドラを使用する必要があります。例外が発生すると、最新の状態の BFILE 変数にアクセスできるのは例外ハンドラのみであるためこの処理が必要です。

例外によってプログラム制御が PL/SQL プログラム・ブロック外に転送されると、オープン中の BFILE への参照はすべて失われます。この結果、オープン・ファイル・カウントが増加し、SESSION_MAX_OPEN_FILES 値を超える場合があります。

たとえば、BFILE 値の終了位置を超えて READ 操作を行い、NO_DATA_FOUND 例外が発生したと想定します。

```

DECLARE
    fil BFILE;
    pos INTEGER;
    amt BINARY_INTEGER;
    buf RAW(40);
BEGIN
    SELECT f_lob INTO fil FROM lob_table WHERE key_value = 21;
    dbms_lob.open(fil, dbms_lob.lob_readonly);
    amt := 40; pos := 1 + dbms_lob.getlength(fil); buf := '';
    dbms_lob.read(fil, amt, pos, buf);
    dbms_output.put_line('Read F1 past EOF: ' ||
        utl_raw.cast_to_varchar2(buf));
    dbms_lob.close(fil);
END;

```

```

ORA-01403: no data found
ORA-06512: at "SYS.DBMS_LOB", line 373
ORA-06512: at line 10

```

例外が発生した後、BFILE ロケータの変数ファイルは有効範囲外となり、その変数を使用したファイル操作は実行できません。したがって、解決方法として、次のような例外ハンドラを使用します。

```

DECLARE
    fil BFILE;
    pos INTEGER;
    amt BINARY_INTEGER;
    buf RAW(40);
BEGIN
    SELECT f_lob INTO fil FROM lob_table WHERE key_value = 21;
    dbms_lob.open(fil, dbms_lob.lob_readonly);
    amt := 40; pos := 1 + dbms_lob.getlength(fil); buf := '';
    dbms_lob.read(fil, amt, pos, buf);
    dbms_output.put_line('Read F1 past EOF: ' ||
        utl_raw.cast_to_varchar2(buf));
    dbms_lob.close(fil);
exception
    WHEN no_data_found
    THEN
        BEGIN
            dbms_output.put_line('End of File reached. Closing file');
            dbms_lob.fileclose(fil);
            -- or dbms_lob.filecloseall if appropriate
        END;
END;
/

```

```
Statement processed.  
End of File reached. Closing file
```

通常、DBMS_LOB を使用して PL/SQL ブロック内でオープンされたファイルは、ブロックの正常終了または異常終了の前にクローズしてください。

テンポラリー LOB

Oracle8i では、テンポラリー LOB の定義、作成、削除、アクセスおよび更新がサポートされています。テンポラリー LOB データは、ユーザーのテンポラリー表領域に格納されています。テンポラリー LOB はデータベースに永続的には格納されません。この目的は、主に LOB データの変換の実行です。

テンポラリー LOB は空の状態で作成されます。デフォルトではテンポラリー LOB は、それが作成されたセッションの終了時にすべて削除されます。処理が途中で停止したり、データベースがクラッシュした場合、テンポラリー LOB は削除され、その領域は解放されます。

Oracle8i には、テンポラリー LOB を論理バケットにまとめてグループ化するインタフェースもあります。テンポラリー LOB 用のこの論理記憶域は期間で表されます。各テンポラリー LOB には、CACHE/NOCACHE など、別々の記憶特性があります。各セッションごとにデフォルトの記憶域があり、ユーザーが特定の期間を指定しない場合、テンポラリー LOB はこの記憶域に格納されます。さらに、期間ごとに解放操作を実行して、期間内のすべてのコンテンツを解放できます。

テンポラリー LOB に関する、一貫性読込み (CR)、取消し、バックアップ、パラレル処理またはトランザクション管理はサポートされていません。テンポラリー LOB には、CR とロールバックがサポートされていないため、エラーが発生した場合、ユーザーはテンポラリー LOB を解放して、最初から操作をやり直す必要があります。

CR、取消しおよびバージョンが、テンポラリー LOB に対しては生成されないため、同じテンポラリー LOB に複数のロケータを割り当てると、パフォーマンスに影響する可能性があります。具体的には、各ロケータがテンポラリー LOB の独自のコピーを所有するためです。

別のロケータがテンポラリー LOB を指しているときに、その内容を変更する場合は、テンポラリー LOB のコピーが作成されます。この場合、変更操作が行われたロケータは、テンポラリー LOB の新しいコピーを指します。これ以降、他のロケータは、変更操作が行われたロケータと同じデータは参照しません。このような状況の場合、永続 LOB でディープ・コピーは行われません。これは、CR スナップショットやバージョン・ページによって、ユーザーは独自のバージョンの LOB を簡単に参照できるためです。

必要な場合は、OCI のロケータへのポインタを使用し、ロケータへの複数のポインタが同じテンポラリー LOB ロケータを指すように設定することによって、擬似 REF 構文を取得できます。PL/SQL では、1 つのテンポラリー LOB に対して複数のロケータは使用しないでください。テンポラリー LOB ロケータは、参照によって別のプロシージャに渡される可能性があります。

テンポラリー LOB は表スキーマと関連付けられていないため、行内および行外のテンポラリー LOB という概念はありません。ユーザーがテンポラリー LOB インスタンスを作成すると、エンジンによって LOB データへのロケータが作成され戻されます。PL/SQL の DBMS_LOB パッ

テージ、PRO*C、OCI およびその他のプログラム・インタフェースは、このロケータを介して、永続 LOB に対する操作と同様にテンポラリー LOB を操作します。

クライアント側のテンポラリー LOB はサポートされていません。テンポラリー LOB はすべてサーバーに常駐します。

テンポラリー LOB は、永続 LOB がサポートしている EMPTY_BLOB または EMPTY_CLOB ファンクションをサポートしません。EMPTY_BLOB ファンクションは、LOB の初期化を指定しますが、データの移入は行いません。

テンポラリー LOB インスタンスは、適切な FREETEMPORARY または OCIDurationEnd 文で OCI または DBMS_LOB パッケージを使用したときのみ破棄できます。

テンポラリー LOB インスタンスは、適切な OCI と DBMS_LOB 文を使用することによって、標準の永続内部 LOB と同様にアクセスおよび変更できます。テンポラリー LOB を永続 LOB に変更するには、OCI または DBMS_LOB の COPY コマンドを明示的に使用して、テンポラリー LOB を永続 LOB にコピーする必要があります。

セキュリティは、LOB ロケータを介して提供されます。テンポラリー LOB は、その作成ユーザーのみ参照できます。ロケータは、あるユーザーのセッションから別のユーザーのセッションに渡すことはできません。あるセッションから別のセッションにロケータを渡しても、元のセッションのテンポラリー LOB にはアクセスできません。テンポラリー LOB データ参照は、各ユーザー固有のセッションに限定されます。あるユーザーが別のセッションのロケータを使用しても、そのユーザーのセッション内で同じ LOB ID を持つ LOB にしかアクセスできません。このような処理は行わないでください。仮に実行した場合でも別のユーザーのデータを操作することはできません。

Oracle は、V\$TEMPORARY_LOBS と呼ばれる v\$ ビューにセッションごとのテンポラリー LOB を記録します。この中には、セッションごとに存在しているテンポラリー LOB 数に関する情報が含まれています。v\$ ビューは、DBA が使用するためのものです。セッションから、Oracle はどのユーザーがそのテンポラリー LOB を所有しているかを判断できます。V\$TEMPORARY_LOBS を DBA_SEGMENTS と組み合わせて使用すると、DBA は、セッションでテンポラリー LOB 用に使われている領域の量を把握できます。これらの表を使用すると、DBA はテンポラリー LOB が使用しているテンポラリー領域の緊急クリーン・アップをモニターして指示できます。

テンポラリ LOB の使用上の注意

1. 入力パラメータのいずれかが NULL の場合、DBMS_LOB のファンクションはすべて NULL を戻します。LOB ロケータが NULL で入力されると、DBMS_LOB のすべてのプロシージャで例外が発生します。
2. CLOB に基づく操作では、パラメータ（CLOB パラメータや VARCHAR2 のバッファとパターンなど）のキャラクタ・セット ID が一致しているかどうかは検証されません。この確認はユーザーが各自で行ってください。
3. データ記憶域リソースは、DBA が異なるテンポラリ表領域を作成して制御します。必要に応じて、DBA はユーザーごとに別々のテンポラリ表領域を定義できます。
4. テンポラリ LOB は値構文に準拠しています。これは、永続 LOB との一貫性を保ち、LOB 用の ANSI 規格に従うためです。このため、OCILobLocatorAssign または PL/SQL で同じ割当てが実行されるたびに、データベースではテンポラリ LOB のコピーが作成されます。

各ロケータは、それぞれ独自の LOB 値を指します。テンポラリ LOB を作成するためにあるロケータが使用され、そのロケータが、OCI の OCILobLocatorAssign を使用して、または PL/SQL の割当て操作によって別の LOB ロケータに割り当てられた場合、データベースは元のテンポラリ LOB のコピーを作成し、2 番目のロケータがそのコピーを指すようにします。

複数のユーザーが同じ LOB を変更できるようにするためには、同じロケータを使用する必要があります。OCI では、ロケータへのポインタを使用して、そのポインタが同じロケータを指すように割り当てることによって、比較的簡単にこの処理を実行できます。PL/SQL では、同様の効果を得るには、同じ LOB 変数を使用して LOB を更新する必要があります。

次の例は、コピーが作成される場合、または最低 1 回サーバーへの特別なラウンドトリップが行われる場合を示しています。

```
DECLARE
  a blob;
  b blob;
BEGIN
  dbms_lob.createtemporary(b, TRUE);
  -- the following assignment results in a deep copy
  a := b;
END;
```

PL/SQL コンパイラは、OUT または IN OUT パラメータにバインドされる実引数のテンポラリ・コピーを作成します。実パラメータがテンポラリ LOB の場合、テンポラリ・コピーはディープ（値）コピーです。

次の PL/SQL ブロックは、テンポラリ LOB を IN OUT パラメータとして渡すことによって、ディープ・コピーが行われる例を示しています。

```
DECLARE
  a blob;
  procedure foo(parm IN OUT blob) is
  BEGIN
    ...
  END;
BEGIN
  dbms_lob.createtemporary(a, TRUE);
  -- the following call results in a deep copy of the blob a
  foo(a);
END;
```

PL/SQL パラメータの受渡しのためのディープ・コピーを最小限にするためには、可能であれば NOCOPY のコンパイラ・ヒントを使用します。

関連項目： NOCOPY 構文の詳細は、『Oracle8i PL/SQL ユーザーズ・ガイドおよびリファレンス』を参照してください。

テンポラリ LOB の例外

表 18-3 DBMS_LOB パッケージの例外

例外	コード	説明
INVALID_ARGVAL	21560	引数 %s の値が無効です。
ACCESS_ERROR	22925	%s で LOB の最大サイズを超えて、読み込みまたは書き込みを実行しようとしました。
NO_DATA_FOUND		ループ読み取り操作の EndofLob の標識。ハード・エラーではありません。
VALUE_ERROR	6502	サブプログラムのパラメータの値が無効のため、PL/SQL エラーが発生しました。

サブプログラムの要約

表 18-4 DBMS_LOB のサブプログラム

サブプログラム	説明
18-14 ページの APPEND プロシージャ	ソース LOB の内容を宛先 LOB に追加します。
18-16 ページの CLOSE プロシージャ	オープンしている内部または外部 LOB をクローズします。
18-17 ページの COMPARE ファンクション	2 つの LOB 全体、または 2 つの LOB の一部を比較します。
18-20 ページの COPY プロシージャ	ソース LOB 全体または一部を宛先 LOB にコピーします。
18-22 ページの CREATETEMPORARY プロシージャ	テンポラリ BLOB または CLOB とそれに対応する索引をユーザーのデフォルト・テンポラリ表領域に作成します。
18-23 ページの ERASE プロシージャ	LOB 全体または一部を消去します。
18-25 ページの FILECLOSE プロシージャ	ファイルをクローズします。
18-26 ページの FILECLOSEALL プロシージャ	オープンしているファイルをすべてクローズします。
18-27 ページの FILEEXISTS ファンクション	ファイルがサーバー上に存在しているかどうかをチェックします。
18-29 ページの FILEGETNAME プロシージャ	ディレクトリ別名とファイル名を取得します。
18-30 ページの FILEISOPEN ファンクション	入力 BFILE ロケータを使用してファイルがオープンされたかどうかをチェックします。
18-32 ページの FILEOPEN プロシージャ	ファイルをオープンします。
18-33 ページの FREETEMPORARY プロシージャ	ユーザーのデフォルト・テンポラリ表領域にあるテンポラリ BLOB または CLOB を解放します。
18-34 ページの GETCHUNKSIZE ファンクション	LOB 値を格納する LOB チャンクの使用領域容量を戻します。
18-35 ページの GETLENGTH ファンクション	LOB 値の長さを取得します。
18-37 ページの INSTR ファンクション	LOB にあるパターン of n 番目のオカレンスのマッチング位置を戻します。

表 18-4 DBMS_LOB のサブプログラム

サブプログラム	説明
18-40 ページの ISOPEN ファンクション	LOB が入力ロケータを使用してすでにオープンされたかどうかをチェックします。
18-41 ページの ISTEMPORARY ファンクション	ロケータがテンポラリ LOB を指しているかどうかをチェックします。
18-42 ページの LOADFROMFILE プロシージャ	BFILE データを内部 LOB にロードします。
18-44 ページの OPEN プロシージャ	指定されたモードで LOB（内部、外部またはテンポラリ）をオープンします。
18-45 ページの READ プロシージャ	指定されたオフセット以降の LOB データを読み込みます。
18-49 ページの SUBSTR ファンクション	指定されたオフセット以降の LOB 値の一部を戻します。
18-52 ページの TRIM プロシージャ	LOB 値を指定された長さに切り捨てます。
18-53 ページの WRITE プロシージャ	LOB の指定されたオフセット以降にデータを書き込みます。
18-56 ページの WRITEAPPEND プロシージャ	LOB の終わり以降にバッファを書き込みます。

APPEND プロシージャ

このプロシージャは、ソース内部 LOB の内容を宛先 LOB に追加します。ソース LOB 全体を追加します。

APPEND プロシージャは 2 種類、オーバーロードされています。

構文

```
DBMS_LOB.APPEND (  
    dest_lob IN OUT NOCOPY BLOB,  
    src_lob IN BLOB);  
  
DBMS_LOB.APPEND (  
    dest_lob IN OUT NOCOPY CLOB CHARACTER SET ANY_CS,  
    src_lob IN CLOB CHARACTER SET dest_lob%CHARSET);
```

プラグマ

なし

パラメータ

表 18-5 APPEND プロシージャのパラメータ

パラメータ	説明
dest_lob	データを追加する内部 LOB のロケータ
src_lob	データを読み取る内部 LOB のロケータ

例外

表 18-6 APPEND プロシージャの例外

例外	説明
VALUE_ERROR	ソースまたは宛先 LOB のいずれかが NULL です。

例

```
CREATE OR REPLACE PROCEDURE Example_1a IS
    dest_lob, src_lob BLOB;
BEGIN
    -- get the LOB locators
    -- note that the FOR UPDATE clause locks the row
    SELECT b_lob INTO dest_lob
        FROM lob_table
        WHERE key_value = 12 FOR UPDATE;
    SELECT b_lob INTO src_lob
        FROM lob_table
        WHERE key_value = 21;
    DBMS_LOB.APPEND(dest_lob, src_lob);
    COMMIT;
EXCEPTION
    WHEN some_exception
    THEN handle_exception;
END;

CREATE OR REPLACE PROCEDURE Example_1b IS
    dest_lob, src_lob BLOB;
BEGIN
    -- get the LOB locators
    -- note that the FOR UPDATE clause locks the row
    SELECT b_lob INTO dest_lob
        FROM lob_table
        WHERE key_value = 12 FOR UPDATE;
```

```
SELECT b_lob INTO src_lob
FROM lob_table
WHERE key_value = 12;
DBMS_LOB.APPEND(dest_lob, src_lob);
COMMIT;
EXCEPTION
WHEN some_exception
THEN handle_exception;
END;
```

CLOSE プロシージャ

このプロシージャは、オープンしている内部または外部 LOB をクローズします。

構文

```
DBMS_LOB.CLOSE (
    lob_loc    IN OUT NOCOPY BLOB);

DBMS_LOB.CLOSE (
    lob_loc    IN OUT NOCOPY CLOB CHARACTER SET ANY_CS);

DBMS_LOB.CLOSE (
    file_loc   IN OUT NOCOPY BFILE);
```

プラグマ

なし

エラー

BFILE が存在していてもオープンしていない場合、エラーは戻されません。エラーは、LOB がオープンしていない場合に戻されます。

使用要件

CLOSE では、内部または外部 LOB のいずれの場合もサーバーへのラウンドトリップが必要です。内部 LOB の場合、CLOSE はクローズ・コールに依存するその他のコードをトリガーし、外部 LOB (BFILE) の場合は、サーバー側のオペレーティング・システム・ファイルを実際にクローズします。

COMPARE ファンクション

このファンクションは、2つのLOB全体、または2つのLOBの一部を比較します。同じデータ型のLOBのみ比較できます（BLOB型のLOBとその他のBLOB、CLOBとCLOB、およびBFILEとBFILEを比較できます）。BFILEの場合は、この操作を行う前に、FILEOPEN操作でファイルをあらかじめオープンしておく必要があります。

offset と amount パラメータによって指定した範囲のデータがすべて完全に一致すると0（ゼロ）が戻されます。一致しない場合は0（ゼロ）以外のINTEGERが戻されます。

固定幅の n バイトのCLOBの場合は、COMPARE に対する入力 amount の指定が $(4294967295/n)$ を超えると、 $(4294967295/n)$ または Max (length(clob1), length(clob2)) のサイズのうち、小さい方の範囲で文字を比較します。

構文

```
DBMS_LOB.COMPARE (  
    lob_1           IN BLOB,  
    lob_2           IN BLOB,  
    amount          IN INTEGER := 4294967295,  
    offset_1        IN INTEGER := 1,  
    offset_2        IN INTEGER := 1)  
RETURN INTEGER;  
  
DBMS_LOB.COMPARE (  
    lob_1           IN CLOB CHARACTER SET ANY_CS,  
    lob_2           IN CLOB CHARACTER SET lob_1%CHARSET,  
    amount          IN INTEGER := 4294967295,  
    offset_1        IN INTEGER := 1,  
    offset_2        IN INTEGER := 1)  
RETURN INTEGER;  
  
DBMS_LOB.COMPARE (  
    lob_1           IN BFILE,  
    lob_2           IN BFILE,  
    amount          IN INTEGER,  
    offset_1        IN INTEGER := 1,  
    offset_2        IN INTEGER := 1)  
RETURN INTEGER;
```

プラグマ

```
pragma restrict_references (COMPARE, WNDS, WNPS, RNDS, RNPS);
```

パラメータ

表 18-7 COMPARE ファンクションのパラメータ

パラメータ	説明
lob_1	最初の比較対象の LOB ロケータ。
lob_2	2 番目の比較対象の LOB ロケータ。
amount	比較するバイト数 (BLOB の場合) または文字数 (CLOB の場合)。
offset_1	最初の LOB の比較開始位置を示すオフセット (起点: 1)。バイト数または文字数で指定します。
offset_2	2 番目の LOB の比較開始位置を示すオフセット (起点: 1)。バイト数または文字数で指定します。

戻り値

- INTEGER: 比較が正常に行われた場合は 0 (ゼロ)、それ以外の場合は 0 (ゼロ) 以外の整数が戻されます。
- 次の場合に NULL が戻されます。
 - amount < 1
 - amount > LOBMAXSIZE
 - offset_1 または offset_2 < 1
 - * offset_1 または offset_2 > LOBMAXSIZE

例外

表 18-8 BFILE 操作に関する COMPARE ファンクションの例外

例外	説明
UNOPENED_FILE	ファイルが入力ロケータを使用してオープンされていません。
NOEXIST_DIRECTORY	ディレクトリが存在しません。
NOPRIV_DIRECTORY	ディレクトリに対する権限がありません。
INVALID_DIRECTORY	ディレクトリがファイルのオープン後に無効となりました。
INVALID_OPERATION	ファイルが存在しないか、またはファイルのアクセス権限がありません。

例

```

CREATE OR REPLACE PROCEDURE Example2a IS
    lob_1, lob_2      BLOB;
    retval            INTEGER;
BEGIN
    SELECT b_col INTO lob_1 FROM lob_table
        WHERE key_value = 45;
    SELECT b_col INTO lob_2 FROM lob_table
        WHERE key_value = 54;
    retval := dbms_lob.compare(lob_1, lob_2, 5600, 33482,
        128);
    IF retval = 0 THEN
        ; -- process compared code
    ELSE
        ; -- process not compared code
    END IF;
END;

CREATE OR REPLACE PROCEDURE Example_2b IS
    fil_1, fil_2      BFILE;
    retval            INTEGER;
BEGIN

    SELECT f_lob INTO fil_1 FROM lob_table WHERE key_value = 45;
    SELECT f_lob INTO fil_2 FROM lob_table WHERE key_value = 54;
    dbms_lob.fileopen(fil_1, dbms_lob.file_readonly);
    dbms_lob.fileopen(fil_2, dbms_lob.file_readonly);
    retval := dbms_lob.compare(fil_1, fil_2, 5600,
        3348276, 2765612);

    IF (retval = 0)
    THEN
        ; -- process compared code
    ELSE
        ; -- process not compared code
    END IF;
    dbms_lob.fileclose(fil_1);
    dbms_lob.fileclose(fil_2);
END;

```

COPY プロシージャ

このプロシージャは、ソース内部 LOB の全体または一部を宛先内部 LOB にコピーします。ソースと宛先 LOB の両方に対するオフセット、およびコピーするバイト数または文字数を指定できます。

宛先 LOB に指定したオフセットが、現在その LOB に格納されているデータの終わりを超えている場合は、宛先の BLOB または CLOB に、0（ゼロ）バイトの FILLER または空白がそれぞれ挿入されます。オフセットが宛先 LOB の現行の長さより小さい場合、既存のデータは上書きされます。

ソース LOB のデータ長を超える量を指定してもエラーにはなりません。したがって、ソース LOB のデータを、src_offset からソース LOB の終わりまでコピーする大量のコピーを指定できます。

構文

```
DBMS_LOB.COPY (
    dest_lob      IN OUT NOCOPY BLOB,
    src_lob       IN              BLOB,
    amount        IN              INTEGER,
    dest_offset   IN              INTEGER := 1,
    src_offset    IN              INTEGER := 1);

DBMS_LOB.COPY (
    dest_lob      IN OUT NOCOPY CLOB CHARACTER SET ANY_CS,
    src_lob       IN              CLOB CHARACTER SET dest_lob%CHARSET,
    amount        IN              INTEGER,
    dest_offset   IN              INTEGER := 1,
    src_offset    IN              INTEGER := 1);
```

プラグマ

なし

パラメータ

表 18-9 COPY プロシージャのパラメータ

パラメータ	説明
dest_lob	コピー先の LOB ロケータ。
src_lob	コピー元の LOB ロケータ。
amount	コピーするバイト数（BLOB の場合）または文字数（CLOB の場合）。
dest_offset	宛先 LOB のコピー開始位置を示すオフセット（起点:1）。バイト数または文字数で指定します。

表 18-9 COPY プロシージャのパラメータ

パラメータ	説明
src_offset	ソース LOB のコピー開始位置を示すオフセット（起点: 1）バイト数または文字数で指定します。

戻り値

なし

例外

表 18-10 COPY プロシージャの例外

例外	説明
VALUE_ERROR	入力パラメータのいずれかが NULL または無効です。
INVALID_ARGVAL	次のいずれかです。 - src_offset または dest_offset < 1 - src_offset または dest_offset > LOBMAXSIZE - amount < 1 - amount > LOBMAXSIZE

例

```
CREATE OR REPLACE PROCEDURE Example_3a IS
  lobd, lobs      BLOB;
  dest_offset     INTEGER := 1
  src_offset      INTEGER := 1
  amt             INTEGER := 3000;
BEGIN
  SELECT b_col INTO lobd
    FROM lob_table
   WHERE key_value = 12 FOR UPDATE;
  SELECT b_col INTO lobs
    FROM lob_table
   WHERE key_value = 21;
  DBMS_LOB.COPY(lobd, lobs, amt, dest_offset, src_offset);
  COMMIT;
EXCEPTION
  WHEN some_exception
  THEN handle_exception;
END;
```

```
CREATE OR REPLACE PROCEDURE Example_3b IS
    lobd, lobs      BLOB;
    dest_offset     INTEGER := 1
    src_offset      INTEGER := 1
    amt             INTEGER := 3000;
BEGIN
    SELECT b_col INTO lobd
    FROM lob_table
    WHERE key_value = 12 FOR UPDATE;
    SELECT b_col INTO lobs
    FROM lob_table
    WHERE key_value = 12;
    DBMS_LOB.COPY(lobd, lobs, amt, dest_offset, src_offset);
    COMMIT;
    EXCEPTION
        WHEN some_exception
        THEN handle_exception;
END;
```

CREATETEMPORARY プロシージャ

このプロシージャは、テンポラリ BLOB または CLOB とそれに対応する索引をユーザーのデフォルト・テンポラリ表領域に作成します。

構文

```
DBMS_LOB.CREATETEMPORARY (
    lob_loc IN OUT NOCOPY BLOB,
    cache   IN             BOOLEAN,
    dur      IN             PLS_INTEGER := 10);

DBMS_LOB.CREATETEMPORARY (
    lob_loc IN OUT NOCOPY CLOB CHARACTER SET ANY_CS,
    cache   IN             BOOLEAN,
    dur      IN             PLS_INTEGER := 10);
```

プラグマ

なし

パラメータ

表 18-11 CREATETEMPORARY プロシージャのパラメータ

パラメータ	説明
lob_loc	LOB ロケータ。
cache	LOB をバッファ・キャッシュに読み込むかどうかを指定します。
dur	2 つの事前定義の継続時間（SESSION または CALL）のうちの 1 つ。 テンポラリ LOB のクリーン・アップをセッション終了時に行うか、 コール終了時に行うかを指定します。 dur が省略された場合は、セッション継続時間が使用されます。

戻り値

なし

例外

なし

例

DBMS_LOB.CREATETEMPORARY (Dest_Loc, TRUE)

関連項目： NOCOPY とテンポラリ LOB をパラメータとして渡す方法は、『Oracle8i PL/SQL ユーザーズ・ガイドおよびリファレンス』を参照してください。

ERASE プロシージャ

このプロシージャは、内部 LOB 全体、または内部 LOB の一部を消去します。

注意： LOB の長さは、LOB のセクションを消去しても減少しません。LOB 値の長さを減らす方法は、18-52 ページの「[TRIM プロシージャ](#)」を参照してください。

LOB の中央部のデータが消去されると、BLOB または CLOB には、0（ゼロ）バイトの FILLER または空白がそれぞれ書き込まれます。

指定した数を消去する前に LOB の終わりに達した場合、実際に消去されたバイト数または文字数は、amount パラメータで指定した数と異なる場合があります。実際に消去された文字数またはバイト数は、amount パラメータに戻されます。

構文

```
DBMS_LOB.ERASE (
  lob_loc          IN OUT NOCOPY BLOB,
  amount           IN OUT NOCOPY INTEGER,
  offset           IN             INTEGER := 1);

DBMS_LOB.ERASE (
  lob_loc          IN OUT NOCOPY CLOB CHARACTER SET ANY_CS,
  amount           IN OUT NOCOPY INTEGER,
  offset           IN             INTEGER := 1);
```

プラグマ

なし

パラメータ

表 18-12 ERASE プロシージャのパラメータ

パラメータ	説明
lob_loc	消去する LOB のロケータ。
amount	消去するバイト数（BLOB または BFILES の場合）または文字数（CLOB または NCLOB の場合）。
offset	LOB の先頭からの絶対オフセット（起点 : 1）。BLOB の場合はバイト数、CLOB の場合は文字数で指定します。

戻り値

なし

例外

表 18-13 ERASE プロシージャの例外

例外	説明
VALUE_ERROR	入力パラメータのいずれかが NULL です。
INVALID_ARGVAL	次のいずれかです。 - amount < 1 または amount > LOBMAXSIZE - offset < 1 または offset > LOBMAXSIZE

例

```
CREATE OR REPLACE PROCEDURE Example_4 IS
    lobd          BLOB;
    amt           INTEGER := 3000;
BEGIN
    SELECT b_col INTO lobd
    FROM lob_table
    WHERE key_value = 12 FOR UPDATE;
    dbms_lob.erase(dest_lob, amt, 2000);
    COMMIT;
END;
```

関連項目： 18-52 ページの「[TRIM プロシージャ](#)」

FILECLOSE プロシージャ

このプロシージャは、入力ロケータによってすでにオープンされている BFILE をクローズします。

注意： Oracle には、BFILE に対する読取り専用アクセスしかありません。つまり、Oracle を介して BFILE に書き込むことはできません。

構文

```
DBMS_LOB.FILECLOSE (
    file_loc IN OUT NOCOPY BFILE);
```

プラグマ

なし

パラメータ

表 18-14 FILECLOSE プロシージャのパラメータ

パラメータ	説明
file_loc	クローズする BFILE のロケータ

戻り値

なし

例外

表 18-15 FILECLOSE プロシージャの例外

例外	説明
VALUE_ERROR	file_loc の入力値が NULL です。
UNOPENED_FILE	ファイルが入力ロケータを使用してオープンされていません。
NOEXIST_DIRECTORY	ディレクトリが存在しません。
NOPRIV_DIRECTORY	ディレクトリに対する権限がありません。
INVALID_DIRECTORY	ディレクトリがファイルのオープン後に無効となりました。
INVALID_OPERATION	ファイルが存在しないか、またはファイルのアクセス権限がありません。

例

```
CREATE OR REPLACE PROCEDURE Example_5 IS
    fil BFILE;
BEGIN
    SELECT f_lob INTO fil FROM lob_table WHERE key_value = 99;
    dbms_lob.fileopen(fil);
    -- file operations
    dbms_lob.fileclose(fil);
    EXCEPTION
        WHEN some_exception
        THEN handle_exception;
END;
```

関連項目： 18-32 ページの「[FILEOPEN プロシージャ](#)」 および 18-26 ページの「[FILECLOSEALL プロシージャ](#)」

FILECLOSEALL プロシージャ

このプロシージャは、セッションでオープンされたすべての BFILE をクローズします。

構文

```
DBMS_LOB.FILECLOSEALL;
```

プラグマ

なし

戻り値

なし

例外

表 18-16 FILECLOSEALL プロシージャの例外

例外	説明
UNOPENED_FILE	セッションでオープンされたファイルはありません。

例

```
CREATE OR REPLACE PROCEDURE Example_6 IS
    fil BFILE;
BEGIN
    SELECT f_lob INTO fil FROM lob_table WHERE key_value = 99;
    dbms_lob.fileopen(fil);
    -- file operations
    dbms_lob.filecloseall;
EXCEPTION
    WHEN some_exception
    THEN handle_exception;
END;
```

関連項目： 18-32 ページの「[FILEOPEN プロシージャ](#)」および 18-25 ページの「[FILECLOSE プロシージャ](#)」

FILEEXISTS ファンクション

このファンクションは、指定した BFILE ロケータが、サーバーのファイル・システムに実際に存在しているファイルを指しているかどうかを検証します。

構文

```
DBMS_LOB.FILEEXISTS (
    file_loc IN BFILE)
RETURN INTEGER;
```

プラグマ

```
pragma restrict_references(FILEEXISTS, WNDS, RNDS, WNPS, RNPS);
```

パラメータ

表 18-17 FILEEXISTS ファンクションのパラメータ

パラメータ	説明
file_loc	BFILE のロケータ

戻り値

表 18-18 FILEEXISTS ファンクションの戻り値

戻り値	説明
0	物理ファイルが存在しません。
1	物理ファイルが存在します。

例外

表 18-19 FILEEXISTS ファンクションの例外

例外	説明
NOEXIST_DIRECTORY	ディレクトリが存在しません。
NOPRIV_DIRECTORY	ディレクトリに対する権限がありません。
INVALID_DIRECTORY	ディレクトリがファイルのオープン後に無効となりました。

例

```
CREATE OR REPLACE PROCEDURE Example_7 IS
    fil BFILE;
BEGIN
    SELECT f_lob INTO fil FROM lob_table WHERE key_value = 12;
    IF (dbms_lob.fileexists(fil))
    THEN
        ; -- file exists code
    ELSE
        ; -- file does not exist code
    END IF;
    EXCEPTION
        WHEN some_exception
        THEN handle_exception;
END;
```

関連項目： [「FILEISOPEN ファンクション」](#)

FILEGETNAME プロシージャ

このプロシージャは、指定した BFILE ロケータのディレクトリ別名とファイル名を判別します。ロケータに割り当てられたディレクトリ別名とファイル名を示すのみで、物理ファイルまたはディレクトリが実際に存在しているかどうかは判別しません。

dir_alias バッファの最大値は 30 で、パス名全体の最大値は 2000 です。

構文

```
DBMS_LOB.FILEGETNAME (  
    file_loc    IN    BFILE,  
    dir_alias   OUT   VARCHAR2,  
    filename    OUT   VARCHAR2);
```

プラグマ

なし

パラメータ

表 18-20 FILEGETNAME プロシージャのパラメータ

パラメータ	説明
file_loc	BFILE のロケータ
dir_alias	ディレクトリ別名
filename	BFILE 名

戻り値

なし

例外

表 18-21 FILEGETNAME プロシージャの例外

例外	説明
VALUE_ERROR	入力パラメータのいずれかが NULL または INVALID です。
INVALID_ARGVAL	dir_alias またはファイル名が NULL です。

例

```
CREATE OR REPLACE PROCEDURE Example_8 IS
    fil BFILE;
    dir_alias VARCHAR2(30);
    name VARCHAR2(2000);
BEGIN
    IF (dbms_lob.fileexists(fil))
    THEN
        dbms_lob.filegetname(fil, dir_alias, name);
        dbms_output.put_line("Opening " || dir_alias || name);
        dbms_lob.fileopen(fil, dbms_lob.file_readonly);
        -- file operations
        dbms_output.fileclose(fil);
    END IF;
END;
```

FILEISOPEN ファンクション

このファンクションは、BFILE が特定の FILE ロケータでオープンされたかどうかを検証します。

入力 FILE ロケータが FILEOPEN プロシージャに渡されていない場合、そのファイルはこのロケータによってオープンされていないとみなされます。ただし、別のロケータがこのファイルをオープンしている可能性があります。つまり、オープンされているかどうかは特定のロケータと関連付けられています。

構文

```
DBMS_LOB.FILEISOPEN (
    file_loc IN BFILE)
RETURN INTEGER;
```

プラグマ

```
pragma restrict_references(FILEISOPEN, WNDS, RNDS, WNPS, RNPS);
```

パラメータ

表 18-22 FILEISOPEN ファンクションのパラメータ

パラメータ	説明
file_loc	BFILE のロケータ

戻り値

INTEGER: 0 = ファイルはオープンされていません。1 = ファイルはオープンされています。

例外

表 18-23 FILEISOPEN ファンクションの例外

例外	説明
NOEXIST_DIRECTORY	ディレクトリが存在しません。
NOPRIV_DIRECTORY	ディレクトリに対する権限がありません。
INVALID_DIRECTORY	ディレクトリがファイルのオープン後に無効となりました。
INVALID_OPERATION	ファイルが存在しないか、またはファイルのアクセス権限がありません。

例

```
CREATE OR REPLACE PROCEDURE Example_9 IS
DECLARE
    fil      BFILE;
    pos      INTEGER;
    pattern  VARCHAR2(20);
BEGIN
    SELECT f_lob INTO fil FROM lob_table
        WHERE key_value = 12;
    -- open the file
    IF (fileisopen(fil))
    THEN
        pos := dbms_lob.instr(fil, pattern, 1025, 6);
        -- more file operations
        dbms_lob.fileclose(fil);
    ELSE
        ; -- return error
    END IF;
END;
```

関連項目： 18-27 ページの「[FILEEXISTS ファンクション](#)」

FILEOPEN プロシージャ

このプロシージャは、BFILE を読取り専用アクセスでオープンします。BFILE は、Oracle を介して書き込むことはできません。

構文

```
DBMS_LOB.FILEOPEN (
    file_loc    IN OUT NOCOPY  BFILE,
    open_mode   IN              BINARY_INTEGER := file_readonly);
```

プラグマ

なし

パラメータ

表 18-24 FILEOPEN プロシージャのパラメータ

パラメータ	説明
file_loc	BFILE のロケータ。
open_mode	ファイル・アクセスは読取り専用です。

戻り値

なし

例外

表 18-25 FILEOPEN プロシージャの例外

例外	説明
VALUE_ERROR	file_loc または open_mode が NULL です。
INVALID_ARGVAL	open_mode が FILE_READONLY ではありません。
OPEN_TOOMANY	セッション内のオープン・ファイル数が session_max_open_files を超えています。
NOEXIST_DIRECTORY	file_loc に関連付けられたディレクトリが存在しません。
INVALID_DIRECTORY	ディレクトリがファイルのオープン後に無効となりました。
INVALID_OPERATION	ファイルが存在しないか、またはファイルのアクセス権限がありません。

例

```
CREATE OR REPLACE PROCEDURE Example_10 IS
    fil BFILE;
BEGIN
    -- open BFILE
    SELECT f_lob INTO fil FROM lob_table WHERE key_value = 99;
    IF (dbms_lob.fileexists(fil))
    THEN
        dbms_lob.fileopen(fil, dbms_lob.file_readonly);
        -- file operation
        dbms_lob.fileclose(fil);
    END IF;
    EXCEPTION
        WHEN some_exception
        THEN handle_exception;
END;
```

関連項目： 18-25 ページの「[FILECLOSE プロシージャ](#)」および 18-26 ページの「[FILECLOSEALL プロシージャ](#)」

FREETEMPORARY プロシージャ

このプロシージャは、ユーザーのデフォルト・テンポラリ表領域内のテンポラリ BLOB または CLOB を解放します。FREETEMPORARY のコール後、解放された LOB ロケータには無効のマークが設定されます。

無効の LOB ロケータが、OCI の OCILobLocatorAssign を使用して、または PL/SQL の割当て操作によって別の LOB ロケータに割り当てられている場合、割当て先も解放され、無効のマークが設定されます。

構文

```
DBMS_LOB.FREETEMPORARY (
    lob_loc IN OUT NOCOPY BLOB);

DBMS_LOB.FREETEMPORARY (
    lob_loc IN OUT NOCOPY CLOB CHARACTER SET ANY_CS);
```

プラグマ

なし

パラメータ

表 18-26 FREETEMPORARY プロシージャのパラメータ

パラメータ	説明
lob_loc	LOB ロケータ

戻り値

なし

例外

なし

例

```
DECLARE
  a blob;
  b blob;
BEGIN
  dbms_lob.createtemporary(a, TRUE);
  dbms_lob.createtemporary(b, TRUE);
  ...
  -- the following call frees lob a
  dbms_lob.freetemporary(a);
  -- at this point lob locator a is marked as invalid
  -- the following assignment frees the lob b and marks it as invalid
also
  b := a;
END;
```

GETCHUNKSIZE ファンクション

表の作成時に、Oracle ブロックの倍数でチャンク・ファクタを指定できます。これは、LOB 値のアクセスまたは変更時に LOB データ・レイヤーによって使用されるチャンク・サイズに対応します。チャンクの一部はシステム関連情報の格納に使用され、それ以外の部分に LOB 値が格納されます。

このファンクションは、LOB 値を格納する LOB チャンクで使用される領域容量を戻します。

構文

```
DBMS_LOB.GETCHUNKSIZE (
    lob_loc IN BLOB)
RETURN INTEGER;

DBMS_LOB.GETCHUNKSIZE (
    lob_loc IN CLOB CHARACTER SET ANY_CS)
RETURN INTEGER;
```

プラグマ

```
pragma restrict_references (GETCHUNKSIZE, WNDS, RNDS, WNPS, RNPS);
```

パラメータ

表 18-27 GETCHUNKSIZE ファンクションのパラメータ

パラメータ	説明
lob_loc	LOB ロケータ

戻り値

BLOB の場合に返される値はバイト単位です。CLOB の場合に返される値は文字単位です。

例外

なし

使用上の注意

このチャンク・サイズの倍数を使用して読み込み / 書き込み要求を入力するとパフォーマンスが改善されます。LOB チャンクはバージョン化されるため、書き込みの場合にはさらに利点があります。すべての書き込みがチャンクを基準に行われると、余分なバージョン分割が行われず、重複もしません。同じチャンクに対して WRITE コールを複数回送信するかわりに、1 つのチャンクのサイズ内で WRITE コールをまとめることができます。

GETLENGTH ファンクション

このファンクションは、指定された LOB 値の長さを取得します。長さは、バイト数または文字数で返されます。

BFILE の場合に返される長さには、EOF（存在している場合）が含まれます。以前に行った ERASE または WRITE 操作で LOB に挿入された 0（ゼロ）バイトまたは空白の FILLER も長さに含まれます。空の内部 LOB の長さは 0（ゼロ）です。

構文

```
DBMS_LOB.GETLENGTH (
    lob_loc    IN  BLOB)
RETURN INTEGER;

DBMS_LOB.GETLENGTH (
    lob_loc    IN  CLOB    CHARACTER SET ANY_CS)
RETURN INTEGER;

DBMS_LOB.GETLENGTH (
    lob_loc    IN  BFILE)
RETURN INTEGER;
```

プラグマ

```
pragma restrict_references (GETLENGTH, WNDS, WNPS, RNDS, RNPS);
```

パラメータ

表 18-28 GETLENGTH ファンクションのパラメータ

パラメータ	説明
lob_loc	長さが戻される LOB のロケータ

戻り値

LOB の長さが、INTEGER としてバイト数または文字数で戻されます。入力 LOB が NULL であるか、または入力 lob_loc が NULL の場合は、NULL が戻されます。BFILE の場合は、次の場合にエラーが戻されます。

- lob_loc に必要なディレクトリ権限と OS 権限がない場合
- OS 読み込みエラーのために、lob_loc を読み込むことができない場合

例外

なし

例

```

CREATE OR REPLACE PROCEDURE Example_11a IS
    lobd          BLOB;
    length         INTEGER;
BEGIN
    -- get the LOB locator
    SELECT b_lob INTO lobd FROM lob_table
        WHERE key_value = 42;
    length := dbms_lob.getlength(lob_loc);
    IF length IS NULL THEN
        dbms_output.put_line('LOB is null. ');
    ELSE
        dbms_output.put_line('The length is '
            || length);
    END IF;
END;

CREATE OR REPLACE PROCEDURE Example_11b IS
DECLARE
    len INTEGER;
    fil BFILE;
BEGIN
    SELECT f_lob INTO fil FROM lob_table WHERE key_value = 12;
    len := dbms_lob.length(fil);
END;

```

INSTR ファンクション

このファンクションは、指定されたオフセットを開始位置として、LOB におけるパターンの *n* 番目のオカレンスのマッチング位置を戻します。

VARCHAR2 バッファ (pattern パラメータ) の形式は、CLOB パラメータの形式と一致する必要があります。つまり、入力 LOB のパラメータの型が NCLOB の場合、バッファには NCHAR データが含まれる必要があります。これに対して、入力 LOB のパラメータの型が CLOB の場合、バッファには CHAR データが含まれる必要があります。

BFILE の場合は、この操作を行う前に、FILEOPEN 操作でファイルをあらかじめオープンしておく必要があります。

INSTR など、パターン・マッチング用に RAW または VARCHAR2 パラメータを受け入れる操作では、パターン・パラメータまたは副文字列において、標準の式または特殊一致文字 (例: SQL の LIKE) はサポートされていません。

構文

```
DBMS_LOB.INSTR (
    lob_loc      IN      BLOB,
    pattern      IN      RAW,
    offset       IN      INTEGER := 1,
    nth          IN      INTEGER := 1)
RETURN INTEGER;

DBMS_LOB.INSTR (
    lob_loc      IN      CLOB      CHARACTER SET ANY_CS,
    pattern      IN      VARCHAR2  CHARACTER SET lob_loc%CHARSET,
    offset       IN      INTEGER := 1,
    nth          IN      INTEGER := 1)
RETURN INTEGER;

DBMS_LOB.INSTR (
    lob_loc      IN      BFILE,
    pattern      IN      RAW,
    offset       IN      INTEGER := 1,
    nth          IN      INTEGER := 1)
RETURN INTEGER;
```

プラグマ

```
pragma restrict_references(INSTR, WNDS, WNPS, RNDS, RNPS);
```

パラメータ

表 18-29 INSTR ファクションのパラメータ

パラメータ	説明
lob_loc	検査する LOB のロケータ。
pattern	テスト対象のパターン。パターンは、BLOB の場合は RAW バイト、CLOB の場合は一連の文字列（VARCHAR2）です。パターンの最大サイズは 16383 バイトです。
offset	パターン・マッチングの開始位置を示す絶対オフセット（起点 : 1）。BLOB の場合はバイト数、CLOB の場合は文字数で指定します。
nth	オカレンス番号。1 から始まります。

戻り値

表 18-30 INSTR ファンクションの戻り値

戻り値	説明
INTEGER	一致したパターンの先頭のオフセット。バイト数または文字数で示されます。 パターンが見つからない場合は 0（ゼロ）が戻されます。
NULL	次のいずれかです。 - IN パラメータのいずれかが NULL または INVALID の場合 - offset < 1 または offset > LOBMAXSIZE - nth < 1 - nth > LOBMAXSIZE

例外

表 18-31 BFILES に関する INSTR ファンクションの例外

例外	説明
UNOPENED_FILE	ファイルが入力ロケータを使用してオープンされていません。
NOEXIST_DIRECTORY	ディレクトリが存在しません。
NOPRIV_DIRECTORY	ディレクトリに対する権限がありません。
INVALID_DIRECTORY	ディレクトリがファイルのオープン後に無効となりました。
INVALID_OPERATION	ファイルが存在しないか、またはファイルのアクセス権限がありません。

例

```
CREATE OR REPLACE PROCEDURE Example_12a IS
  lobd          CLOB;
  pattern       VARCHAR2 := 'abcde';
  position      INTEGER  := 10000;
BEGIN
  -- get the LOB locator
  SELECT b_col INTO lobd
    FROM lob_table
   WHERE key_value = 21;
  position := DBMS_LOB.INSTR(lobd,
                             pattern, 1025, 6);
```

```
IF position = 0 THEN
    dbms_output.put_line('Pattern not found');
ELSE
    dbms_output.put_line('The pattern occurs at '
        || position);
END IF;
END;
```

```
CREATE OR REPLACE PROCEDURE Example_12b IS
DECLARE
    fil BFILE;
    pattern VARCHAR2;
    pos INTEGER;
BEGIN
    -- initialize pattern
    -- check for the 6th occurrence starting from 1025th byte
    SELECT f_lob INTO fil FROM lob_table WHERE key_value = 12;
    dbms_lob.fileopen(fil, dbms_lob.file_readonly);
    pos := dbms_lob.instr(fil, pattern, 1025, 6);
    dbms_lob.fileclose(fil);
END;
```

関連項目： 18-49 ページの「[SUBSTR ファンクション](#)」

ISOPEN ファンクション

このファンクションは、LOB が入力ロケータを使用してすでにオープンされたかどうかをチェックします。このサブプログラムは、内部および外部 LOB 用です。

構文

```
DBMS_LOB.ISOPEN (
    lob_loc IN BLOB)
RETURN INTEGER;
```

```
DBMS_LOB.ISOPEN (
    lob_loc IN CLOB CHARACTER SET ANY_CS)
RETURN INTEGER;
```

```
DBMS_LOB.ISOPEN (
    file_loc IN BFILE)
RETURN INTEGER;
```

プラグマ

```
pragma restrict_references(ISOPEN, WNDS, RNDS, WNPS, RNPS);
```


パラメータ

表 18-32 ISOPEN ファンクションのパラメータ

パラメータ	説明
lob_loc	LOB ロケータ
file_loc	ファイル・ロケータ

例外

なし

使用上の注意

BFILES の場合、オープンされているかどうかはロケータと関連付けられています。入力ロケータが OPEN に渡されていない場合、その BFILE はこのロケータによってオープンされていないとみなされます。ただし、別のロケータが BFILE をオープンしている可能性があります。異なるロケータを使用すると、同じ BFILE 上で複数の OPEN を実行できます。

内部 LOB の場合、オープンされているかどうかは、ロケータではなくその LOB に関連付けられています。ロケータ 1 が LOB をオープンした場合、ロケータ 2 もその LOB はオープンしているとみなします。内部 LOB の場合は、LOB が実際にオープンしているかどうかを調べるにはサーバー上の状態をチェックするため、ISOPEN でラウンドトリップが必要です。

外部 LOB (BFILE) の場合も、サーバーに状態が保持されているため、ISOPEN でラウンドトリップが必要です。

ISTEMPORARY ファンクション

このファンクションは、ロケータがテンポラリ LOB を指しているかどうかをチェックします。

構文

```
DBMS_LOB.ISTEMPORARY (  
    lob_loc IN BLOB)  
    RETURN INTEGER;  
  
DBMS_LOB.ISTEMPORARY (  
    lob_loc IN CLOB CHARACTER SET ANY_CS)  
    RETURN INTEGER;
```

プラグマ

```
PRAGMA RESTRICT_REFERENCES(istemporary, WNDS, RNDS, WNPS, RNPS);
```

関連項目： RESTRICT_REFERENCES プラグマの使用方法は、45-2 ページの「[RESTRICT_REFERENCES プラグマ使用上のトラブルシューティング](#)」を参照してください。

パラメータ

表 18-33 ISTEMPORARY プロシージャのパラメータ

パラメータ	説明
lob_loc	LOB ロケータ。
temporary	プール値。LOB がテンポラリであるかどうかを示します。

戻り値

ロケータがテンポラリ LOB を指している場合は、temporary に TRUE が戻されます。そうでない場合は FALSE が戻されます。

例外

なし

LOADFROMFILE プロシージャ

このプロシージャは、ソース外部 LOB (BFILE) の全体または一部を宛先内部 LOB にコピーします。

ソースと宛先の LOB の両方に対するオフセット、およびソース BFILE からコピーするバイト数を指定できます。amount と src_offset は BFILE を参照するため、バイト単位で、dest_offset は BLOB の場合はバイト、CLOB の場合は文字が単位です。

注意： 入力 BFILE は、このプロシージャを使用する前にオープンしている必要があります。バイナリ BFILE データが CLOB にロードされる場合、キャラクタ・セット変換は暗黙的には実行されません。BFILE データは、データベース内の CLOB と同じキャラクタ・セットであることが必要です。これを検証するためのエラー・チェックは実行されません。

宛先 LOB に指定したオフセットが、現在その LOB に格納されているデータの終わりを超えている場合は、宛先の BLOB または CLOB に、0 (ゼロ) バイトの FILLER または空白がそれぞれ挿入されます。オフセットが宛先 LOB の現行の長さより小さい場合、既存のデータは上書きされます。

入力 amount と offset を加算した値が BFILE 内のデータの長さを超えた場合はエラーが発生します。

構文

```
DBMS_LOB.LOADFROMFILE (
    dest_lob    IN OUT NOCOPY BLOB,
    src_file    IN              BFILE,
    amount      IN              INTEGER,
    dest_offset  IN              INTEGER := 1,
    src_offset   IN              INTEGER := 1);

DBMS_LOB.LOADFROMFILE(
    dest_lob    IN OUT NOCOPY CLOB CHARACTER SET ANY_CS,
    src_file    IN              BFILE,
    amount      IN              INTEGER,
    dest_offset  IN              INTEGER := 1,
    src_offset   IN              INTEGER := 1);
```

プラグマ

なし

パラメータ

表 18-34 LOADFROMFILE プロシージャのパラメータ

パラメータ	説明
dest_lob	ロード先の LOB ロケータ。
src_file	ロード元の BFILE ロケータ。
amount	BFILE からロードするバイト数。
dest_offset	宛先 LOB のロード開始位置を示すオフセット（起点:1）。バイト数または文字数で指定します。
src_offset	ソース BFILE のロード開始位置を示すオフセット（起点:1）。バイト数で指定します。

戻り値

なし

例外

表 18-35 LOADFROMFILE プロシージャの例外

例外	説明
VALUE_ERROR	入力パラメータのいずれかが NULL または INVALID です。
INVALID_ARGVAL	次のいずれかです。 - src_offset または dest_offset < 1 - src_offset または dest_offset > LOBMAXSIZE - amount < 1 - amount > LOBMAXSIZE

例

```
CREATE OR REPLACE PROCEDURE Example_l2f IS
  lobd      BLOB;
  fils      BFILE := BFILENAME('SOME_DIR_OBJ','some_file');
  amt       INTEGER := 4000;
BEGIN
  SELECT b_lob INTO lobd FROM lob_table WHERE key_value = 42 FOR UPDATE;
  dbms_lob.fileopen(fils, dbms_lob.file_readonly);
  dbms_lob.loadfromfile(lobd, fils, amt);
  COMMIT;
  dbms_lob.fileclose(fils);
END;
```

OPEN プロシージャ

このプロシージャは、指定されたモードで、(内部または外部) LOB をオープンします。有効なモードは読取り専用と読込み書込みです。同じ LOB を 2 回オープンするとエラーになります。

注意： LOB が読取り専用モードでオープンされた場合は、その LOB に書込みを行おうとすると、エラーが戻されます。BFILE は、読取り専用モードでのみオープンできます。

Oracle8.0 では、定数 file_readonly は BFILE をオープンするときのみ有効なモードです。Oracle 8i では、lob_readonly と lob_readwrite の 2 つの新規定数が DBMS_LOB パッケージに追加されました。

構文

```
DBMS_LOB.OPEN (
    lob_loc    IN OUT NOCOPY BLOB,
    open_mode  IN              BINARY_INTEGER);

DBMS_LOB.OPEN (
    lob_loc    IN OUT NOCOPY CLOB CHARACTER SET ANY_CS,
    open_mode  IN              BINARY_INTEGER);

DBMS_LOB.OPEN (
    file_loc   IN OUT NOCOPY BFILE,
    open_mode  IN              BINARY_INTEGER := file_readonly);
```

プラグマ

なし

パラメータ

表 18-36 OPEN プロシージャのパラメータ

パラメータ	説明
lob_loc	LOB ロケータ
open_mode	オープンするモード

使用上の注意

OPEN では、内部または外部 LOB のいずれの場合もサーバーへのラウンドトリップが必要です。内部 LOB の場合、OPEN は OPEN コールに依存しているその他のコードをトリガーします。外部 LOB (BFILE) の場合は、サーバー側の実際のオペレーティング・システム・ファイルがオープンされるため、ラウンドトリップが必要です。

READ プロシージャ

このプロシージャは、LOB の一部を読み込み、LOB の先頭からの絶対オフセットから開始し、指定された量のデータを buffer パラメータに戻します。

実際に読み込まれたバイト数または文字数は、amount パラメータに戻されます。入力 offset が LOB の終わりを超えた位置を指している場合、amount は 0 (ゼロ) に設定され、NO_DATA_FOUND 例外が発生します。

構文

```
DBMS_LOB.READ (
  lob_loc      IN          BLOB,
  amount       IN OUT     NOCOPY BINARY_INTEGER,
  offset       IN          INTEGER,
  buffer       OUT        RAW);

DBMS_LOB.READ (
  lob_loc      IN          CLOB CHARACTER SET ANY_CS,
  amount       IN OUT     NOCOPY BINARY_INTEGER,
  offset       IN          INTEGER,
  buffer       OUT        VARCHAR2 CHARACTER SET lob_loc%CHARSET);

DBMS_LOB.READ (
  lob_loc      IN          BFILE,
  amount       IN OUT     NOCOPY BINARY_INTEGER,
  offset       IN          INTEGER,
  buffer       OUT        RAW);
```

プラグマ

なし

パラメータ

表 18-37 READ プロシージャのパラメータ

パラメータ	説明
lob_loc	読み込む LOB のロケータ。
amount	読み込む、または読み込まれたバイト数（BLOB の場合）または文字数（CLOB の場合）。
offset	LOB の先頭からのオフセット（起点 : 1）。BLOB の場合はバイト数、CLOB の場合は文字数で指定します。
buffer	読取り操作のアウトプット・バッファ。

戻り値

なし

例外

表 18-38 READ プロシージャの例外

例外	説明
VALUE_ERROR	lob_loc、amount または offset パラメータのいずれかが NULL です。
INVALID_ARGVAL	次のいずれかです。 - amount < 1 - amount > MAXBUFSIZE - offset < 1 - offset > LOBMAXSIZE - amount（バイト数または文字数）が buffer の容量を超えています。
NO_DATA_FOUND	LOB の終わりに達し、LOB から読み込むバイトまたは文字がありません。amount の値は 0（ゼロ）です。

BFILE に関する例外

表 18-39 BFILE に関する READ プロシージャの例外

例外	説明
UNOPENED_FILE	ファイルが入力ロケータを使用してオープンされていません。
NOEXIST_DIRECTORY	ディレクトリが存在しません。
NOPRIV_DIRECTORY	ディレクトリに対する権限がありません。
INVALID_DIRECTORY	ディレクトリがファイルのオープン後に無効となりました。
INVALID_OPERATION	ファイルが存在しないか、またはファイルのアクセス権限がありません。

使用上の注意

VARCHAR2 バッファの形式は、CLOB パラメータの形式と一致する必要があります。つまり、入力 LOB のパラメータの型が NCLOB の場合、バッファには NCHAR データが含まれる必要があります。これに対して、入力 LOB のパラメータの型が CLOB の場合、バッファには CHAR データが含まれる必要があります。

クライアントから DBMS_LOB.READ をコールすると（例：SQL*Plus 内からの BEGIN/END ブロックでのコール）、戻されるバッファにはクライアントのキャラクタ・セットのデータが含まれます。Oracle は、バッファをユーザーに戻す前に、サーバーのキャラクタ・セットからクライアントのキャラクタ・セットに LOB 値を変換します。

例

```
CREATE OR REPLACE PROCEDURE Example_13a IS
    src_lob          BLOB;
    buffer           RAW(32767);
    amt              BINARY_INTEGER := 32767;
    pos              INTEGER := 2147483647;
BEGIN
    SELECT b_col INTO src_lob
    FROM lob_table
    WHERE key_value = 21;
    LOOP
        dbms_lob.read (src_lob, amt, pos, buffer);
        -- process the buffer
        pos := pos + amt;
    END LOOP;
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            dbms_output.put_line('End of data');
END;
```



```
CREATE OR REPLACE PROCEDURE Example_13b IS
    fil BFILE;
    buf RAW(32767);
    amt BINARY_INTEGER := 32767;
    pos INTEGER := 2147483647;
BEGIN
    SELECT f_lob INTO fil FROM lob_table WHERE key_value = 21;
    dbms_lob.fileopen(fil, dbms_lob.file_readonly);
    LOOP
        dbms_lob.read(fil, amt, pos, buf);
        -- process contents of buf
        pos := pos + amt;
    END LOOP;
    EXCEPTION
        WHEN NO_DATA_FOUND
        THEN
            BEGIN
                dbms_output.putline ('End of LOB value reached');
                dbms_lob.fileclose(fil);
            END;
END;
```


ストリーム I/O よりもブロック I/O の方が OS 上で効率的に実行される I/O の例

```
CREATE OR REPLACE PROCEDURE Example_13c IS
    fil BFILE;
    amt BINARY_INTEGER := 1024; -- or n x 1024 for reading n
    buf RAW(1024); -- blocks at a time
    tmpamt BINARY_INTEGER;
BEGIN
    SELECT f_lob INTO fil FROM lob_table WHERE key_value = 99;
    dbms_lob.fileopen(fil, dbms_lob.file_readonly);
    LOOP
        dbms_lob.read(fil, amt, pos, buf);
        -- process contents of buf
        pos := pos + amt;
    END LOOP;
    EXCEPTION
        WHEN NO_DATA_FOUND
        THEN
            BEGIN
                dbms_output.putline ('End of data reached');
                dbms_lob.fileclose(fil);
            END;
END;
```

SUBSTR ファンクション

このファンクションは、LOB の先頭からの絶対 offset から開始し、LOB の amount バイトまたは文字を戻します。

固定幅の n バイトの CLOB の場合、SUBSTR に対する入力 amount の指定が $(32767/n)$ を超えると、長さ $(32767/n)$ の文字バッファまたは CLOB の長さのうち、小さい方が戻されます。

構文

```
DBMS_LOB.SUBSTR (
    lob_loc      IN      BLOB,
    amount       IN      INTEGER := 32767,
    offset       IN      INTEGER := 1)
RETURN RAW;

DBMS_LOB.SUBSTR (
    lob_loc      IN      CLOB CHARACTER SET ANY_CS,
    amount       IN      INTEGER := 32767,
    offset       IN      INTEGER := 1)
RETURN VARCHAR2 CHARACTER SET lob_loc%CHARSET;
```

```
DBMS_LOB.SUBSTR (  
    lob_loc      IN      BFILE,  
    amount       IN      INTEGER := 32767,  
    offset       IN      INTEGER := 1)  
RETURN RAW;
```

プラグマ

```
pragma restrict_references (SUBSTR, WNDS, WNPS, RNDS, RNPS);
```

パラメータ

表 18-40 SUBSTR ファンクションのパラメータ

パラメータ	説明
lob_loc	読み込む LOB のロケータ。
amount	読み込むバイト数（BLOB の場合）または文字数（CLOB の場合）。
offset	LOB の先頭からのオフセット（起点 : 1）。BLOB の場合はバイト数、CLOB の場合は文字数で指定します。

戻り値

表 18-41 SUBSTR ファンクションの戻り値

戻り値	説明
RAW	ファンクション・オーバーロード。パラメータに BLOB または BFILE を使用します。
VARCHAR2	CLOB のバージョン。
NULL	次のいずれかです。 - 入力パラメータのいずれかが NULL です。 - amount < 1 - amount > 32767 - offset < 1 - offset > LOBMAXSIZE

例外

表 18-42 BFILE 操作に関する SUBSTR ファンクションの例外

例外	説明
UNOPENED_FILE	ファイルが入力ロケータを使用してオープンされていません。
NOEXIST_DIRECTORY	ディレクトリが存在しません。
NOPRIV_DIRECTORY	ディレクトリに対する権限がありません。
INVALID_DIRECTORY	ディレクトリがファイルのオープン後に無効となりました。
INVALID_OPERATION	ファイルが存在しないか、またはファイルのアクセス権限がありません。

使用上の注意

VARCHAR2 バッファの形式は、CLOB パラメータの形式と一致する必要があります。つまり、入力 LOB のパラメータの型が NCLOB の場合、バッファには NCHAR データが含まれる必要があります。これに対して、入力 LOB のパラメータの型が CLOB の場合、バッファには CHAR データが含まれる必要があります。

クライアントから DBMS_LOB.SUBSTR をコールすると（例：SQL*Plus 内からの BEGIN/END ブロックでのコール）、戻されるバッファにはクライアントのキャラクタ・セットのデータが含まれます。Oracle は、バッファをユーザーに戻す前に、サーバーのキャラクタ・セットからクライアントのキャラクタ・セットに LOB 値を変換します。

例

```
CREATE OR REPLACE PROCEDURE Example_14a IS
    src_lob          CLOB;
    pos              INTEGER := 2147483647;
    buf              VARCHAR2(32000);
BEGIN
    SELECT c_lob INTO src_lob FROM lob_table
        WHERE key_value = 21;
    buf := DBMS_LOB.SUBSTR(src_lob, 32767, pos);
    -- process the data
END;

CREATE OR REPLACE PROCEDURE Example_14b IS
    fil BFILE;
    pos INTEGER := 2147483647;
    pattern RAW;
```

```
BEGIN
    SELECT f_lob INTO fil FROM lob_table WHERE key_value = 21;
    dbms_lob.fileopen(fil, dbms_lob.file_readonly);
    pattern := dbms_lob.substr(fil, 255, pos);
    dbms_lob.fileclose(fil);
END;
```

関連項目： 18-37 ページの「[INSTR ファンクション](#)」 および 18-45 ページの「[READ プロシージャ](#)」

TRIM プロシージャ

このプロシージャは、内部 LOB の値を newlen パラメータで指定された長さに切り捨てます。BLOB の場合はバイト数、CLOB の場合は文字数で長さを指定します。

注意： TRIM プロシージャは、LOB の長さを newlen パラメータで指定された値に減らします。

空の LOB に対して TRIM を実行すると、何も処理は行われずエラーは戻されません。newlen で指定した新しい長さが LOB のサイズよりも大きい場合は、例外が発生します。

構文

```
DBMS_LOB.TRIM (
    lob_loc          IN OUT NOCOPY BLOB,
    newlen           IN          INTEGER);

DBMS_LOB.TRIM (
    lob_loc          IN OUT NOCOPY CLOB CHARACTER SET ANY_CS,
    newlen           IN          INTEGER);
```

プラグマ

なし

パラメータ

表 18-43 TRIM プロシージャのパラメータ

パラメータ	説明
lob_loc	長さを切り捨てる内部 LOB のロケータ。
newlen	切り捨て後の新しい LOB 値の長さ。BLOB の場合はバイト数、CLOB の場合は文字数で指定します。

戻り値

なし

例外

表 18-44 TRIM プロシージャの例外

例外	説明
VALUE_ERROR	lob_loc が NULL です。
INVALID_ARGVAL	次のいずれかです。 - new_len < 0 - new_len > LOBMAXSIZE

例

```
CREATE OR REPLACE PROCEDURE Example_15 IS
    lob_loc          BLOB;
BEGIN
    -- get the LOB locator
    SELECT b_col INTO lob_loc
    FROM lob_table
    WHERE key_value = 42 FOR UPDATE;
    dbms_lob.trim(lob_loc, 4000);
    COMMIT;
END;
```

関連項目： 18-23 ページの「[ERASE プロシージャ](#)」 および 18-56 ページの「[WRITEAPPEND プロシージャ](#)」

WRITE プロシージャ

このプロシージャは、LOB の先頭からの絶対オフセットから開始し、指定された量のデータを内部 LOB に書き込みます。データは、buffer パラメータから書き込まれます。

WRITE は、オフセット位置以降 LOB にすでに存在しているデータを、指定した長さ分置換（上書き）します。

入力 amount がバッファのデータより多い場合はエラーが発生します。入力 amount がバッファのデータより少ない場合は、その量のバイト数または文字数のみバッファから LOB に書き込まれます。指定したオフセットが現在その LOB に格納されているデータの終わりを超えている場合は、BLOB または CLOB に、0（ゼロ）バイトの FILLER または空白がそれぞれ挿入されます。

構文

```
DBMS_LOB.WRITE (
    lob_loc  IN OUT NOCOPY  BLOB,
    amount   IN              BINARY_INTEGER,
    offset   IN              INTEGER,
    buffer   IN              RAW);

DBMS_LOB.WRITE (
    lob_loc  IN OUT NOCOPY CLOB  CHARACTER SET ANY_CS,
    amount   IN              BINARY_INTEGER,
    offset   IN              INTEGER,
    buffer   IN              VARCHAR2 CHARACTER SET lob_loc%CHARSET);
```

プラグマ

なし

パラメータ

表 18-45 WRITE プロシージャのパラメータ

パラメータ	説明
lob_loc	書き込み先の内部 LOB のロケータ。
amount	書き込む、または書き込まれたバイト数（BLOB の場合）または文字数（CLOB の場合）。
offset	書き込み操作開始位置を示す LOB の先頭からのオフセット（起点:1）。BLOB の場合はバイト数、CLOB の場合は文字数で指定します。
buffer	書き込み用の入力バッファ。

戻り値

なし

例外

表 18-46 WRITE プロシージャの例外

例外	説明
VALUE_ERROR	lob_loc、amount または offset パラメータのいずれかが NULL、 範囲外または INVALID です。
INVALID_ARGVAL	次のいずれかです。 - amount < 1 - amount > MAXBUFSIZE - offset < 1 - offset > LOBMAXSIZE

使用上の注意

VARCHAR2 バッファの形式は、CLOB パラメータの形式と一致する必要があります。つまり、
入力 LOB のパラメータの型が NCLOB の場合、バッファには NCHAR データが含まれる必要が
あります。これに対して、入力 LOB のパラメータの型が CLOB の場合、バッファには CHAR
データが含まれる必要があります。

クライアントから DBMS_LOB.WRITE をコールするときは（例：SQL*Plus 内からの
BEGIN/END ブロック内でコール）、バッファにクライアントのキャラクタ・セットのデータ
が含まれている必要があります。Oracle は、バッファ・データを LOB に書き込む前に、クラ
イアント側のキャラクタ・セットをサーバー側のキャラクタ・セットに変換します。

例

```
CREATE OR REPLACE PROCEDURE Example_16 IS
  lob_loc      BLOB;
  buffer       RAW;
  amt          BINARY_INTEGER := 32767;
  pos          INTEGER := 2147483647;
  i            INTEGER;
BEGIN
  SELECT b_col INTO lob_loc
    FROM lob_table
   WHERE key_value = 12 FOR UPDATE;
  FOR i IN 1..3 LOOP
    dbms_lob.write (lob_loc, amt, pos, buffer);
    -- fill in more data
    pos := pos + amt;
  END LOOP;
```

```
EXCEPTION4
    WHEN some_exception
    THEN handle_exception;
END;
```

関連項目： 18-14 ページの「[APPEND プロシージャ](#)」および 18-20 ページの「[COPY プロシージャ](#)」

WRITEAPPEND プロシージャ

このプロシージャは、指定された量のデータを内部 LOB の後ろに書き込みます。データは、buffer パラメータから書き込まれます。

入力 amount がバッファのデータより多い場合はエラーが発生します。入力 amount がバッファのデータより少ない場合は、その量のバイト数または文字数のみバッファから LOB の後ろに書き込まれます。

構文

```
DBMS_LOB.WRITEAPPEND (
    lob_loc IN OUT NOCOPY BLOB,
    amount IN             BINARY_INTEGER,
    buffer IN             RAW);

DBMS_LOB.WRITEAPPEND (
    lob_loc IN OUT NOCOPY CLOB CHARACTER SET ANY_CS,
    amount IN             BINARY_INTEGER,
    buffer IN             VARCHAR2 CHARACTER SET lob_loc%CHARSET);
```

プラグマ

なし

パラメータ

表 18-47 WRITEAPPEND プロシージャのパラメータ

パラメータ	説明
lob_loc	書き込み先の内部 LOB のロケータ
amount	書き込む、または書き込まれたバイト数（BLOB の場合）または文字数（CLOB の場合）
buffer	書き込み用の入力バッファ

例外

表 18-48 WRITEAPPEND プロシージャの例外

例外	説明
VALUE_ERROR	lob_loc、amount または offset パラメータのいずれかが NULL、 範囲外または INVALID です。
INVALID_ARGVAL	次のいずれかです。 - amount < 1 - amount > MAXBUFSIZE

使用上の注意

VARCHAR2 バッファの形式は、CLOB パラメータの形式と一致する必要があります。つまり、入力 LOB のパラメータの型が NCLOB の場合、バッファには NCHAR データが含まれる必要があります。これに対して、入力 LOB のパラメータの型が CLOB の場合、バッファには CHAR データが含まれる必要があります。

クライアントから DBMS_LOB.WRITEAPPEND をコールするときは（例：SQL*Plus 内からの BEGIN/END ブロック内でコール）、バッファにクライアントのキャラクタ・セットのデータが含まれている必要があります。Oracle は、バッファ・データを LOB に書き込む前に、クライアント側のキャラクタ・セットをサーバー側のキャラクタ・セットに変換します。

例

```
CREATE OR REPLACE PROCEDURE Example_17 IS
    lob_loc    BLOB;
    buffer     RAW;
    amt        BINARY_INTEGER := 32767;
    i          INTEGER;
BEGIN
    SELECT b_col INTO lob_loc
    FROM lob_table
    WHERE key_value = 12 FOR UPDATE;
    FOR i IN 1..3 LOOP
        -- fill the buffer with data to be written to the lob
        dbms_lob.writeappend (lob_loc, amt, buffer);
    END LOOP;
END;
```

関連項目： 18-14 ページの「[APPEND プロシージャ](#)」、18-20 ページの「[COPY プロシージャ](#)」および 18-53 ページ「[WRITE プロシージャ](#)」

アプリケーションに対する Oracle ロック・マネージメント・サービスは、DBMS_LOCK パッケージにあるプロシージャを介して使用できます。特定モードのロックを要求したり、同一または別のインスタンスにある別のプロシージャ内で識別できる一意の名前をロックに付けたり、ロック・モードの変更およびロックの解放を行うことができます。

確保されているユーザー・ロックは Oracle ロックと同一であるため、デッドロック検出など、Oracle ロックの全機能を備えています。分散トランザクションで使用するユーザー・ロックが COMMIT 時に解放されることを確認してください。解放されない場合、デッドロックが検出されない可能性があります。

ユーザー・ロックは、接頭辞 "UL" で識別されているため、Oracle ロックと競合することはありません。これらのロックは、Enterprise Manager ロック・モニター・スクリーンまたは適切な固定ビューを使用して表示できます。ユーザー・ロックは、セッションの終了時に自動的に解放されます。

ロック識別子は、0 ～ 1073741823 の番号です。

ユーザー・ロックを使用して、次のことができます。

- 端末などのデバイスに排他的アクセスを提供します。
- アプリケーションレベルの読み込みロックを施行します。
- ロックの解放時期およびアプリケーション終了後のクリーン・アップを検出します。
- アプリケーションを同期化し、順次処理を施行します。

要件

DBMS_LOCK は、セッション当りのロック数を 200 ～ 300 に制限すると、最も効率的です。プロシージャ間で同一のロックを使用して競合が発生しないように、ロックの使用について標準規則を作成することをお勧めします。たとえば、ロック名の一部に会社名を含める方法があります。

セキュリティ

使用可能なロックの最大数について、オペレーティング・システム固有の制限がある場合があります。これは、ロックの使用時またはこのパッケージを他のユーザーに対して使用可能にするときに考慮する必要があります。特定のユーザーまたはロールに対してのみ EXECUTE 権限の付与を検討してください。

使用するロック数を制限するカバークパッケージを作成してから特定のユーザーに EXECUTE 権限を付与することをお勧めします。カバークパッケージの例は、DBMSLOCK.SQL パッケージ仕様部ファイルに記載されています。

ロックの表示と監視

Oracle は、2 つの機能を提供して、インスタンス内で進行中のトランザクションに関するロック情報を表示します。

Enterprise Manager モニター（ロックとラッチ・モニター） Oracle Enterprise Manager のモニター機能では、インスタンスのロック情報を表示するための 2 つのモニターが提供されます。Enterprise Manager のモニターの詳細は、Oracle Enterprise Manager のドキュメントを参照してください。

UTLLOCKT.SQL UTLLOCKT.SQL スクリプトは、簡単な文字形式のロック待ち状態グラフをツリー構造で表示します。非定型の SQL ツール（SQL*Plus など）を使用してスクリプトを実行すると、システム内のブロックされている側のセッションと対応するブロックしている側のセッションを出力します。このスクリプト・ファイルの場所は、オペレーティング・システムによって異なります。（CATBLOCK.SQL スクリプトを実行してから UTLLOCKT.SQL を使用する必要があります。）

定数

```
nl_mode  constant integer := 1;
ss_mode  constant integer := 2;      -- Also called 'Intended Share'
sx_mode  constant integer := 3;      -- Also called 'Intended Exclusive'
s_mode   constant integer := 4;
ssx_mode constant integer := 5;
x_mode   constant integer := 6;
```

いくつかのロック・モードがあります (NL -> "NULL"、SS -> " 半共有 (Sub Shared) "、SX -> " 半排他 (Sub eXclusive) "、S -> " 共有 (Shared) "、SSX -> " 共有半排他 (Shared Sub eXclusive) "、X -> " 排他 (eXclusive) ")。

半共有ロックを集計オブジェクトで使用して、共有ロックがオブジェクトのサブパーツに対して作動中であることを示すことができます。同様に、半排他ロックを集計オブジェクトで使用して、排他ロックがオブジェクトのサブパーツに対して作動中であることを示すことができます。共有半排他ロックは、集計オブジェクト全体で共有ロックを使用している一方、一部のサブパートではさらに排他ロックを使用していることを示します。

ロックの互換性ルール 別のプロセスが " 保持 " 状態のときに、" 取得 " を試みると次のようになります。

表 19-1 ロックの互換性

保持モード	NL の取得	SS の取得	SX の取得	S の取得	SSX の取得	X の取得
NL	成功	成功	成功	成功	成功	成功
SS	成功	成功	成功	成功	成功	失敗
SX	成功	成功	成功	失敗	失敗	失敗
S	成功	成功	失敗	成功	失敗	失敗
SSX	成功	成功	失敗	失敗	失敗	失敗
X	成功	失敗	失敗	失敗	失敗	失敗

```
maxwait  constant integer := 32767;
```

定数 maxwait は、無期限に待機します。

サブプログラムの要約

表 19-2 DBMS_LOCK パッケージのサブプログラム

サブプログラム	説明
19-4 ページの ALLOCATE_UNIQUE プロシージャ	名前付きロックに一意のロック ID を割り当てます。
19-6 ページの REQUEST ファンクション	特定のモードのロックを要求します。
19-7 ページの CONVERT ファンクション	ロックを別のモードに変換します。
19-9 ページの RELEASE ファンクション	ロックを解放します。
19-10 ページの SLEEP プロシージャ	プロシージャを指定時間だけスリープさせます。

ALLOCATE_UNIQUE プロシージャ

このプロシージャは、一意のロック識別子（1073741824 ~ 1999999999 の範囲内）を指定のロック名に割り当てます。アプリケーションは、ロック識別子を使用してロックの使用方法を調整できます。これは、アプリケーションでロックの使用方法を調整するには、ロック番号よりロック名に基づいた方が容易になるためです。

ロックを名前で識別する場合は、ALLOCATE_UNIQUE を使用して、名前付きロックに一意のロック識別番号を生成できます。

新規ロック名で ALLOCATE_UNIQUE をコールする最初のセッションで、一意のロック ID が生成され、dbms_lock_allocated 表に格納されます。次のコール（通常は他のセッションによる）では、最初のセッションで生成されたロック ID が戻ります。

ロック名は、指定のロック名で ALLOCATE_UNIQUE を最後にコールした後、少なくとも expiration_secs（デフォルトは 10 日間）の間は、戻されたロック ID に関連付けられています。この期間を経過すると、このロック名の dbms_lock_allocated 表にある行は、領域リカバリのために削除される場合があります。ALLOCATE_UNIQUE プロシージャはコミットを実行します。

注意： Oracle では、SQL を使用して指定の名前に関連付けられたロックを判断するため、名前付きユーザー・ロックを使用すると効率が悪くなる場合があります。

構文

```
DBMS_LOCK.ALLOCATE_UNIQUE (  
    lockname          IN  VARCHAR2,  
    lockhandle        OUT VARCHAR2,  
    expiration_secs   IN   INTEGER    DEFAULT 864000);
```

パラメータ

表 19-3 ALLOCATE_UNIQUE プロシージャのパラメータ

パラメータ	説明
lockname	一意の ID を生成するロックの名前。 ORA\$ で始まるロック名は使用しないでください。これはオラクル社の製品用に予約されています。
lockhandle	ALLOCATE_UNIQUE が生成したロック ID にハンドルを戻します。 このハンドルは、REQUEST、CONVERT および RELEASE への次のコールで使用できます。 ハンドルは実際のロック ID のかわりに戻され、プログラミング・エラーにより不適切であるが有効なロック ID が作成されないようにします。これにより、このパッケージを使用するアプリケーション間の独立性が高まります。 LOCKHANDLE は、VARCHAR2 (128) まで可能です。 ALLOCATE_UNIQUE が同じロック名で戻したロック・ハンドルを使用しているすべてのセッションは、同じロックを参照します。したがって、ロック・ハンドルを別のセッションに渡さないでください。
expiration_specs	指定のロックに最後の ALLOCATE_UNIQUE を実行した後、DBMS_LOCK_ALLOCATED 表からロックの削除を許可するまで待機する秒数。 デフォルトの待機期間は 10 日間です。この表からロックを削除しないでください。ALLOCATE_UNIQUE への次のコールで、期限切れのロックを削除し、領域をリカバリできます。

エラー

ORA-20000, ORU-10003: dbms_lock_allocated カタログに <lockname> ロックを挿入または検出できません。

例外

なし

REQUEST ファンクション

このファンクションは、指定のモードのロックを要求します。REQUEST はオーバーロードされたファンクションで、ユーザー定義のロック識別子または ALLOCATE_UNIQUE プロシージャが戻すロック・ハンドルのいずれかを受け入れます。

構文

```
DBMS_LOCK.REQUEST(  
  id                IN  INTEGER ||  
  lockhandle        IN  VARCHAR2,  
  lockmode          IN  INTEGER DEFAULT X_MODE,  
  timeout           IN  INTEGER DEFAULT MAXWAIT,  
  release_on_commit IN  BOOLEAN DEFAULT FALSE,  
  RETURN INTEGER;
```

X_MODE や MAXWAIT などの現行のデフォルト値は、DBMS_LOCK パッケージ仕様部で定義されています。

パラメータ

表 19-4 REQUEST ファンクションのパラメータ

パラメータ	説明
id or lockhandle	変更するロック・モードのユーザー割当てロック識別子（0 ～ 1073741823）または ALLOCATE_UNIQUE が戻すロック・ハンドル。
lockmode	要求するロックのモード。 使用可能なモードと関連する整数識別子は次のとおりです。カッコ内の略称は、V\$ ビューと Enterprise Manager モニターに表示される時のロックの略称です。 1 - NULL モード 2 - 行共有モード（ULRS） 3 - 行排他モード（ULRX） 4 - 共有モード（ULS） 5 - 共有行排他モード（ULRSX） 6 - 排他モード（ULX） 各ロック・モードの説明は、『Oracle8i 概要』を参照してください。

表 19-4 REQUEST ファンクションのパラメータ

パラメータ	説明
timeout	ロックの付与を待つ秒数。 この時間内にロックが付与できない場合、コールは値 1 (timeout) を戻します。
release_on_commit	このパラメータを TRUE に設定すると、ロックをコミットまたはロールバック時に解放します。 そうでない場合は、ロックが明示的に解放されるかセッションが終了するまで、ロックは解放されません。

戻り値

表 19-5 REQUEST ファンクションの戻り値

戻り値	説明
0	成功。
1	タイムアウト。
2	デッドロック。
3	パラメータ・エラー。
4	id または lockhandle で指定されたロックをすでに所有しています。
5	不正なロック・ハンドル。

例外

なし

CONVERT ファンクション

このファンクションは、ロックを別のモードに変換します。CONVERT はオーバーロードされたファンクションで、ユーザー定義のロック識別子または ALLOCATE_UNIQUE プロシージャが戻すロック・ハンドルのいずれかを受け入れます。

構文

```
DBMS_LOCK.CONVERT(  
    id          IN INTEGER ||  
    lockhandle  IN VARCHAR2,  
    lockmode    IN INTEGER,  
    timeout     IN NUMBER DEFAULT MAXWAIT)  
RETURN INTEGER;
```

パラメータ

表 19-6 CONVERT ファンクションのパラメータ

パラメータ	説明
id or lockhandle	変更するロック・モードのユーザー割当てロック識別子（0 ～ 1073741823）または ALLOCATE_UNIQUE が戻すロック・ハンドル。
lockmode	<p>指定のロックに割り当てる新規モード。</p> <p>使用可能なモードと関連する整数識別子は次のとおりです。カッコ内の略称は、VS ビューと Enterprise Manager モニターに表示されるときのロックの略称です。</p> <p>1 - NULL モード</p> <p>2 - 行共有モード（ULRS）</p> <p>3 - 行排他モード（ULRX）</p> <p>4 - 共有モード（ULS）</p> <p>5 - 共有行排他モード（ULRSX）</p> <p>6 - 排他モード（ULX）</p> <p>各ロック・モードの説明は、『Oracle8i 概要』を参照してください。</p>
timeout	<p>ロック・モードの変更を待つ秒数。</p> <p>この時間内にロックを変換できない場合、コールは値 1（timeout）を戻します。</p>

戻り値

表 19-7 CONVERT ファンクションの戻り値

戻り値	説明
0	成功。
1	タイムアウト。
2	デッドロック。
3	パラメータ・エラー。
4	id または lockhandle が指定したロックを所有していません。
5	不正なロック・ハンドル。

例外

なし

RELEASE ファンクション

このファンクションは、REQUEST ファンクションを使用して前に取得したロックを明示的に解放します。ロックは、セッションの終了時に自動的に解放されます。RELEASE はオーバーロードされたファンクションで、ユーザー定義のロック識別子または ALLOCATE_UNIQUE プロシージャが戻すロック・ハンドルのいずれかを受け入れます。

構文

```
DBMS_LOCK.RELEASE (  
    id          IN INTEGER)  
RETURN INTEGER;  
  
DBMS_LOCK.RELEASE (  
    lockhandle IN VARCHAR2)  
RETURN INTEGER;
```

パラメータ

表 19-8 RELEASE ファンクションのパラメータ

パラメータ	説明
id or lockhandle	変更するロック・モードのユーザー割当てロック識別子（0 ～ 1073741823）または ALLOCATE_UNIQUE が戻すロック・ハンドル。

戻り値

表 19-9 RELEASE ファンクションの戻り値

戻り値	説明
0	成功。
3	パラメータ・エラー。
4	id または lockhandle が指定するロックを所有していません。
5	不正なロック・ハンドル。

例外

なし

SLEEP プロシージャ

このプロシージャは、指定時間だけセッションを中断します。

構文

```
DBMS_LOCK.SLEEP (  
    seconds IN NUMBER);
```

パラメータ

表 19-10 SLEEP プロシージャのパラメータ

パラメータ	説明
seconds	セッションを中断する時間（秒）。 入力できる最小増分値は 100 分の 1 秒です。たとえば、1.95 は有効な時間値です。

例

この Pro*COBOL プリコンパイラの例では、複数のユーザーが単一のデバイスにアクセスするとき、ロックを使用して競合が発生しないようにする方法を示します。

小切手の印刷

レジ係が顧客の返品に対して返金を行うとします。50 ドル未満の返金は現金で、それ以上の場合は小切手で返金します。次のコードにより、小切手を印刷します。小切手の印刷のつどプリンタをオープンしてクローズする負担を避けるために、1 台のプリンタはすべてのレジ係に対してオープンしています。このため、プリンタへの排他アクセスを確保しないと、複数のレジ係からの出力ラインがインターリーブ状態となる可能性があります。DBMS_LOCK パッケージを使用して、排他アクセスを確保します。

CHECK-PRINT

プリンタ・ロックに対するロック " ハンドル " を取得します。

```
MOVE "CHECKPRINT" TO LOCKNAME-ARR.  
MOVE 10 TO LOCKNAME-LEN.  
EXEC SQL EXECUTE  
    BEGIN DBMS_LOCK.ALLOCATE_UNIQUE ( :LOCKNAME, :LOCKHANDLE );  
    END; END-EXEC.
```

プリンタを排他モード（デフォルトのモード）にロックします。

```
EXEC SQL EXECUTE  
    BEGIN DBMS_LOCK.REQUEST ( :LOCKHANDLE );  
    END; END-EXEC.
```

これでプリンタを排他的に使用できるので、小切手を印刷します。

...

プリンタのロックを解除して、他のユーザーが使用できるようにします。

```
EXEC SQL EXECUTE  
    BEGIN DBMS_LOCK.RELEASE ( :LOCKHANDLE );  
  
    END; END-EXEC.
```

DBMS_LOGMNR

DBMS_LOGMNR は、ツールの初期化に必要なファイル名リストと SCN を持つログ分析ツールを提供します。このプロシージャを完了すると、サーバーは、V\$LOGMNR_CONTENTS ビューに対して SELECT を処理する用意ができます。

REDO ログ・データを使用して、データベースがいつ破損したかを正確に調べることができるので、REDO ログ・データはデータ・リカバリのために特に重要です。REDO ログの情報をを使用して、データベースが破損する前の状態にリカバリできます。

関連項目：『Oracle8i 管理者ガイド』および『Oracle8i バックアップおよびリカバリ・ガイド』

LogMiner の使用方法

DBMS_LOGMNR_D でディクショナリ・ファイルを作成した後、アーカイブ REDO ログの分析を開始できます。

1. マウントまたはアンマウントされたデータベースで Oracle インスタンスを起動します。
2. DBMS_LOGMNR.ADD_LOGFILE プロシージャを実行して、読み込むログ・ファイルまたはファイルを指定します。V\$LOGMNR_LOGS を使用して、指定したログ・ファイルの情報を表示できます。
3. DBMS_LOGMNR.START_LOGMNR プロシージャを使用して、ログ・リーダーを起動します。START_LOGMNR コマンドで開始と終了の SCN、および時間パラメータを設定して、分析する REDO レコードにフィルタをかけることができます。V\$LOGMNR_PARAMETERS ビューを問い合わせ、パラメータを表示できます。
4. V\$LOGMNR_CONTENTS 表から出力を表示します。LogMiner は、SCN 順序（メディア・リカバリに適用される順序と同じ）ですべての行を戻します。

関連項目： 20-10 ページの「例」

定数

ADD_LOGFILE オプション・フラグの定数

NEW	DBMS_LOGMNR.NEW は、ログ・ファイルの既存リストがある場合、これをパージします。指定したログ・ファイルを、分析用のログ・ファイルのリストに設定します。
ADDFILE	DBMS_LOGMNR.ADDFILE は、ログ・ファイルを分析用のログ・ファイルのリストに追加します。
REMOVEFILE	DBMS_LOGMNR.REMOVEFILE は、分析用のログ・ファイルのリストからログ・ファイルを削除します。以前に追加されていないファイルを削除しようとすると、例外（ORA-1290）が発生します。

START_LOGMNR オプション・フラグの定数

USE_COLMAP	DBMS_LOGMNR.USE_COLMAP は、logmnr.opt ファイルで指定した列マップを使用します。このファイルは、DictFileName で指定したディクショナリ・ファイルと同じディクショナリであることが必要です。
------------	---

`SKIP_CORRUPTION` LogMiner に、破損 REDO ブロックをスキップして、処理を継続するように指示します。このオプションは、REDO ブロック（REDO ログ・ファイルのヘッダーは除く）が破損している場合のみ機能します。コー側は、`V$LOGMNR_CONTENTS` ビューの `INFO` 列をチェックして、LogMiner がスキップした破損ブロックを確認する必要があります。

プレース・ホルダ列の使用法

`V$LOGMNR_CONTENTS` 表には、プレース・ホルダ列の複数のセットが含まれています。各プレース・ホルダ列セットは、名前列、REDO 値列および UNDO 値列で構成されています。各プレース・ホルダ列セットは、オプションの LogMiner 代入ファイル (`logmnr.opt`) を介して表と列に割り当てられます。プレース・ホルダ列を割り当てた後は、そのプレース・ホルダ列を使用して、割り当てられた列と表への変更を REDO ログ・ストリームから選択できます。

たとえば、割り当て `"colmap = SCOTT EMP (1, EMPNO);"` は、PH1 プレース・ホルダ列セットを表（つまり `SCOTT.EMP` と列 `EMPNO`）に割り当てます。割り当てた後は、`EMP` 表の `EMPNO` 列に対する変更を REDO ストリームから選択できます。

```
SELECT scn FROM V$LOGMNR_CONTENTS
WHERE ph1_name='EMPNO'
AND ph1_redo='12345';
```

REDO ストリームが処理され、`EMP` 表の `EMPNO` 列に値 12345 を設定した変更が戻されます。

各プレース・ホルダ列セットについて、複数の割当てを行うことができます。次に例を示します。

```
colmap = SCOTT EMP (1, EMPNO);
```

続いて次のように指定します。

```
colmap = ACCOUNTING CUSTOMER (1, CUSTID);
```

この場合、PH1 プレース・ホルダ列セットには2つの割当てがあります。ユーザーは、次の問合せで示されているように、プレース・ホルダ列を使用して、`EMP` 表または `CUSTOMER` 表いずれかへの変更を選択できます。

```
SELECT scn FROM V$LOGMNR_CONTENTS
WHERE seg_name = 'EMP'
AND ph1_name='EMPNO'
AND ph1_redo='12345';
```

または

```
SELECT scn FROM V$LOGMNR_CONTENTS
WHERE seg_name = 'CUSTOMER'
AND phl_name='CUSTID'
AND phl_redo='12345';
```

logmnr.opt プレース・ホルダ列の使用方法

logmnr.opt ファイルは、DBMS_LOGMNR.START_LOGMNR が実行され、Options が USE_COLMAP に設定 (Options=USE_COLMAP) された場合に処理されます。Options で USE_COLMAP を設定すると、LogMiner は logmnr.opt ファイルを読み込んで処理します。logmnr.opt ファイルは、LogMiner ディクショナリ・ファイルと同じディレクトリにある必要があります。

プレース・ホルダ列の代入ファイル (logmnr.opt) の処理後、V\$LOGMNR_CONTENTS 表から次に選択する内容は、すべて割当て済プレース・ホルダ列を使用できます。割当てを変更するには、logmnr.opt ファイルを更新して、LogMiner を再度開始します。

logmnr.opt ファイルが処理されるとき、割当て済の列は現行の LogMiner ディクショナリに対して検証されます。割当て済の列が存在しない場合、開始処理は失敗します。

logmnr.opt の構文ルール

```
line = 'colmap' <sp> '=' <sp> <schema> <sp> <table> <sp> '(' map ')' ' ';
map = <num> ',' <colname> [ ',' <num> ',' <colname>]
```

<sp> 空白

引用符で囲まれたワードは、固定の記号です。

<num> 任意の番号 (プレース・ホルダ列セットの番号に限りです)。

<table> 表の名前。

<schema> スキーマ名。

<colname> 指定した <schema>.<table> の列名。

V\$LOGMNR_CONTENTS 表にあるプレース・ホルダ列の数まで、カッコで囲まれた <num> ' ' <colname> を繰り返すことができます。

<table>、<schema> および <colname> は、すべて大文字で記述する必要があります。

有効な logmnr.opt 構文

- ユーザーがいずれかの表の列をプレース・ホルダに入れる場合は、次のように指定します。

```
colmap = SCOTT EMP (1, EMPNO, 2, SAL, 3, JOB, 4, MGR, 5, COMM);
colmap = SCOTT DEPT (1, DEPTNO);
```

- colmap は、重複する値を含むことができます。次の例では、最初の ph1_redo、ph1_undo および ph4_redo、ph4_undo のみ入力されます。

```
colmap = SCOTT EMP (1, EMPNO, 2, EMPNO, 3, EMPNO, 4, MGR);
```

無効な logmnr.opt 構文

- 構文エラー: "colmap" と "=" の間に空白がありません。

```
colmap= SCOTT EMP (1, EMPNO, 2, SAL);
```

- 構文エラー: マップが正しく指定されていません。

```
colmap = SCOTT EMP (1, EMPNO, 2);
```

- 構文エラー: 文が ';' で終了していません。

```
colmap = SCOTT EMP (1, EMPNO)
```

- エラー: スキーマが小文字です。

```
colmap = scott EMP (1, EMPNO, 2, SAL);
```

- エラー: 表名が小文字です。

```
colmap = SCOTT emp (1, EMPNO, 2, SAL);
```

- エラー: マップのエントリが5つを超えています。

```
colmap = SCOTT EMP (1, EMPNO, 2, SAL, 3, JOB, 4, MGR, 5, COMM, 6, ENAME);
```

- エラー: 列 REGION が表にありません。

```
colmap = SCOTT EMP (1, EMPNO, 2, SAL, 3, JOB, 4, REGION);
```

サブプログラムの要約

表 20-1 DBMS_LOGMNR パッケージのサブプログラム

サブプログラム	説明
20-6 ページの ADD_LOGFILE プロシージャ	ファイルを既存または新規作成した処理対象のアーカイブ・ファイルのリストに追加します。
20-7 ページの START_LOGMNR プロシージャ	ログ分析ツールを初期化します。
20-10 ページの END_LOGMNR プロシージャ	セッションを終了します。

ADD_LOGFILE プロシージャ

このプロシージャは、ファイルを既存または新規作成した処理対象のアーカイブ・ファイルのリストに追加します。

V\$LOGMNR_CONTENTS ビューから情報を検索するために、LogMiner セッションはいくつかの情報を使用して設定する必要があります。このプロシージャは、LogMiner セッションに分析するログ・ファイルのリストを通知します。

注意： 5つのログ・ファイルを分析する場合は、ADD_LOGFILE プロシージャを 5 回コールする必要があります。

構文

```
DBMS_LOGMNR.ADD_LOGFILE(  
  LogFileName      IN VARCHAR2,  
  Options          IN BINARY_INTEGER default ADDFILE );
```

パラメータ

表 20-2 ADD_LOGFILE プロシージャのパラメータ

パラメータ	説明
LogFileName	このセッションによる分析のために、ログ・ファイルのリストに追加する必要があるログ・ファイル名。
Options	次のいずれかです。 - 新規リスト（DBMS_LOGMNR.NEW）を開始します。 - 既存のリスト（DBMS_LOGMNR.ADDFILE）にファイルを追加します。または、 - ログ・ファイル（DBMS_LOGMNR.REMOVEFILE）を削除します。 20-2 ページの「 ADD_LOGFILE オプション・フラグの定数 」を参照してください。

例外

- ORA-1284: ログ・ファイルをオープンできません。ログ・ファイルまたはディレクトリが存在していないか、またはアクセスできません。
- ORA-1286: 指定したログ・ファイルは、分析用に追加した他のログ・ファイルを生成したデータベースからのファイルではありません。
- ORA-1287: 指定したログ・ファイルは、別のデータベース再現によるファイルです。
- ORA-1289: 重複するログ・ファイルを追加しようとしてしました。
- ORA-1290: リストにないログ・ファイルを削除しようとしてしました。

START_LOGMNR プロシージャ

このプロシージャは、LogMiner セッションを開始します。

注意： ADD_LOGFILE プロシージャによって、分析するログ・ファイルのリストがあらかじめ指定されていない場合、このプロシージャは失敗します。

構文

```
DBMS_LOGMNR.START_LOGMNR (  
  startScn          IN NUMBER default 0,  
  endScn            IN NUMBER default 0,  
  startTime         IN DATE default '01-jan-1988',  
  endTime           IN DATE default '01-jan-2988',  
  DictFileName      IN VARCHAR2 default '',  
  Options            IN BINARY_INTEGER default 0 );
```

パラメータ

表 20-3 START_LOGMNR プロシージャのパラメータ

パラメータ	説明
startScn	指定した startSCN 以上の SCN を持つ REDO レコードのみ処理対象にします。SCN 範囲（V\$LOGMNR_LOGS ビューに表示されるログ・ファイルに関連する LOW_SCN と NEXT_SCN）に startScn を含むログ・ファイルがない場合、このプロシージャは失敗します。
endScn	指定した endSCN 以下の SCN を持つ REDO レコードのみ処理対象にします。SCN 範囲（V\$LOGMNR_LOGS ビューに表示されるログ・ファイルに関連する LOW_SCN と NEXT_SCN）に endScn を含んだログ・ファイルがない場合、このプロシージャは失敗します。

表 20-3 START_LOGMNR プロシージャのパラメータ

パラメータ	説明
startTime	指定した startTime 以上のタイムスタンプを持つ REDO レコードのみ処理対象にします。時間範囲 (V\$LOGMNR_LOGS ビューに表示されるログ・ファイルに関連する LOW_TIME と HIGH_TIME) に startTime を含んだログ・ファイルがない場合、このプロシージャは失敗します。startScn が指定されている場合、このパラメータは無視されます。
endTime	指定した endTime 以下のタイムスタンプを持つ REDO レコードのみ処理対象にします。時間範囲 (V\$LOGMNR_LOGS ビューに表示されるログ・ファイルに関連する LOW_TIME と HIGH_TIME) に endTime を含んだログ・ファイルがない場合、このプロシージャは失敗します。endScn が指定されている場合、このパラメータは無視されます。
DictFileName	このフラット・ファイルには、データベース・カタログのスナップショットが含まれています。完全に変換された SEG_NAME、SEG_OWNER、SEG_TYPE_NAME および TABLE_SPACE 列とともに、再構成された SQL_REDO と SQL_UNDO 列を V\$LOGMNR_CONTENTS で参照する場合、このパラメータは指定する必要があります。ディクショナリ・ファイルの完全修飾パス名を指定する必要があります (このファイルは、DBMS_LOGMNR_D.BUILD プロシージャですでに作成されている必要があります)。
Options	DBMS_LOGMNR.USE_COLMAP: logmnr.opt ファイルで指定した列マップを使用します。このファイルは、DictFileName で指定したディクショナリ・ファイルと同じディクショナリである必要があります。 20-2 ページの「START_LOGMNR オプション・フラグの定数」および 20-4 ページの「logmnr.opt プレース・ホルダ列の使用方法」を参照してください。

START_LOGMNR() プロシージャの実行後、V\$LOGMNR_DICTIONARY ビューを問い合わせると、ディクショナリ・ファイルに関する現在の情報を取得できます。LogMiner セッションに関する情報は、V\$LOGMNR_PARAMETERS ビューを問い合わせます。

例外

- プロシージャは、LogMiner で内部エラーが発生すると ORA-1280 で失敗します。
- ORA-1281: endScn が startScn 未満です。
- ORA-1282: endDate が startDate より前の日付です。
- ORA-1283: 指定したオプションは無効です。
- プロシージャは、次の理由から ORA-1293 で失敗します。
 1. 指定した startScn を含む範囲 (LOW_SCN、NEXT_SCN) を持つログ・ファイルがない場合。
 2. 指定した endScn を含む範囲 (LOW_SCN、NEXT_SCN) を持つログ・ファイルがない場合。
 3. 指定した startTime を含む範囲 (LOW_TIME、HIGH_TIME) を持つログ・ファイルがない場合。
 4. 指定した endTime を含む範囲 (LOW_TIME、HIGH_TIME) を持つログ・ファイルがない場合。
- ORA-1294: 指定したディクショナリ・ファイルが破損しています。
- ORA-1295: 指定したディクショナリが、分析対象のログ・ファイルを生成した同じデータベースに対応していません。
- ORA-1301: DBMS_LOGMNR.USE_COLMAP が、ディクショナリ・ファイルなしで使用されました。
- ORA-1302: logmnr.opt ファイルに構文エラーがあります。
- ORA-1303: logmnr.opt ファイルに指定されているスキーマが存在しません。
- ORA-1304: logmnr.opt ファイルに指定されている表が存在しません。
- ORA-1305: logmnr.opt ファイルに指定されている列が存在しません。

END_LOGMNR プロシージャ

このプロシージャは、セッションを終了します。

構文

```
DBMS_LOGMNR.END_LOGMNR;
```

パラメータ

なし

例外

- ORA-1307: logminer セッションはアクティブではありません。END_LOGMNR プロシージャが、ログ・ファイルの追加なしにコールされました。

例

```
EXECUTE DBMS_LOGMNR.ADD_LOGFILE(  
    LogFileName => '/oracle/logs/log1.f',  
    Options => dbms_logmnr.NEW);  
  
EXECUTE DBMS_LOGMNR.ADD_LOGFILE(  
    LogFileName => '/oracle/logs/log2.f',  
    Options => dbms_logmnr.ADDFILE);  
  
EXECUTE DBMS_LOGMNR.START_LOGMNR(  
    DictFileName => '/oracle/dictionary.ora');  
  
SELECT sql_redo  
FROM V$logmnr_contents  
EXECUTE DBMS_LOGMNR.END_LOGMNR();
```

DBMS_LOGMNR_D

DBMS_LOGMNR_D には LogMnr プロシージャが含まれ、LogMnr ディクショナリ・ファイルを作成します。このプロシージャは、現行データベースのディクショナリ表を問い合わせ、その内容を格納したテキスト・ベースのファイルを作成します。外部ディクショナリ・ファイルは、LogMnr ツールを使用する今後のログ・ファイル分析のために作成されます。

関連項目：『Oracle8i 管理者ガイド』および『Oracle8i バックアップおよびリカバリ・ガイド』

ディクショナリ・ファイルの作成

ディクショナリ・ファイルは、データベースをマウントし、ディクショナリ情報を外部ファイルに抽出して作成します。ディクショナリ・ファイルは、分析するログ・ファイルを生成した同じデータベースから作成する必要があります。ディクショナリ・ファイルが作成されると、ログ・ファイルの分析に使用できます。

1. 分析するファイルがあるデータベースをマウントしてオープンします。
2. PL/SQL プロシージャ DBMS_LOGMNR_D.BUILD を実行します。このプロシージャにより、ログ・ファイルの分析に使用するディクショナリ・ファイルを作成します。

サブプログラムの要約

DBMS_LOGMNR_D には、1 つのプロシージャ BUILD が含まれています。このプロシージャは、ログ・ファイルの分析に使用するディクショナリ・ファイルを作成します。

BUILD プロシージャ

このプロシージャは、現行データベースのディクショナリ表を問い合わせ、その内容を格納したテキスト・ベースのファイルを作成します。

構文

```
DBMS_LOGMNR_D.BUILD (  
    dictionary_filename IN VARCHAR2,  
    dictionary_location IN VARCHAR2);
```

パラメータ

表 21-1 BUILD プロシージャのパラメータ

パラメータ	説明
dictionary_filename	ディクショナリ・ファイルの名前
dictionary_location	ディクショナリ・ファイルへのパス

使用上の注意

ディクショナリ・ファイルは、すべてのディクショナリをデータベースに変更した後、分析するログ・ファイルを作成する前に作成してください。

BUILD プロシージャは、init.ora で UTL_FILE_DIR パラメータの設定が必要な UTL_FILE パッケージを使用します。

SET SERVEROUTPUT ON により、ディクショナリ作成の進捗をモニターします。

ディクショナリ・ファイルに書き込まれた表の一部は、8i より古いバージョンのデータベースには存在しません。この場合、ディクショナリ作成中に 1 つ以上のエラー (ORA: 942) が発生する場合があります。これは予期した動作です。

例

次の例では、ディクショナリ・ファイルを作成します。

```
/oracle/database/l_dictionary.ora
```

```
SQLPLUS>EXECUTE dbms_logmnr_d.build('l_dictionary.ora',  
SQLPLUS>' /oracle/database/');
```

関連項目： 第 20 章「DBMS_LOGMNR」

例外

- ORA-1308: 初期化パラメータ UTL_FILE_DIR が設定されていません。

次の場合に、プロシージャは ORA-1309 エラーをレポートします。

1. Dictionary_location が存在しない場合
2. UTL_FILE_DIR に dictionary_location へのアクセス権限が設定されていない場合
3. Dictionary_file が読取り専用の場合

DBMS_MVIEW

DBMS_SNAPSHOT のシノニムとしての DBMS_MVIEW

DBMS_MVIEW を使用すると、同じリフレッシュ・グループおよびパージ・ログの一部ではないマテリアライズド・ビューをリフレッシュできます。DBMS_MVIEW は、DBMS_SNAPSHOT のシノニムであるため、DBMS_MVIEW の詳細は、[第 48 章「DBMS_SNAPSHOT」](#)を参照してください。

DBMS_OBFUSCATION_TOOLKIT

DBMS_OBFUSCATION_TOOLKIT パッケージは、文字列入力およびロー入力に対して 2 つのデータ暗号化規格（Data Encryption Standard: DES）プロシージャを提供します。

- [DESEncrypt プロシージャ](#)
- [DESDecrypt プロシージャ](#)

このパッケージは SYS スキーマにインストールされます。必要に応じて、既存のユーザーとロールに、パッケージのアクセス権限を付与できます。

DESEncrypt プロシージャ

DESEncrypt プロシージャは、入力データの暗号化フォームを生成します。DESEncrypt 構文の例は、この章の最後にあります。

パラメータの説明

表 23-1 と表 23-2 は、DESEncrypt 構文のパラメータ、モード、型および説明の一覧です。

表 23-1 ロー・データ用の DESEncrypt パラメータ

パラメータ名	モード	型	説明
input	IN	RAW	暗号化するデータ
key	IN	RAW	暗号キー
encrypted_data	OUT	RAW	暗号化されたデータ

表 23-2 文字列データ用の DESEncrypt パラメータ

パラメータ名	モード	型	説明
input_string	IN	VARCHAR2	暗号化する文字列
key_string	IN	VARCHAR2	暗号キー文字列
encrypted_string	OUT	VARCHAR2	暗号化された文字列

PL/SQL DESEncrypt ファンクションに渡された入力データまたはキーに値が指定されていない場合は、ORA エラー 28231 「Obfuscation ツールキットへの入力が無効です。」が発生します。

DESEncrypt ファンクションに渡された入力データが 8 バイトの倍数でない場合は、ORA エラー 28232 「Obfuscation ツールキットに対する入力サイズが無効です。」が発生します。

DESEncrypt ファンクションを使用して暗号化データを暗号化しようとした場合は、ORA エラー 28233 「Double encryption not supported by DESEncrypt in Obfuscation toolkit」が発生します。

暗号化プロシージャの制限事項

DESEncryption プロシージャには 2 つの制限事項があります。1 つは、暗号化用の DES キーの長さが 56 ビットに固定されていることです。このキーの長さは変更できません。

もう 1 つは、暗号化の多重パスを実行できないことです。つまり、ファンクションを 2 回コールして、すでに暗号化されているデータを再度暗号化することはできません。

注意： キーの長さの制限と多重暗号パスの防止は、暗号化製品の輸出を統括している米国の条項における要件です。

DESDecrypt プロシージャ

DESDecrypt プロシージャの目的は、入力データの復号化フォームを生成することにあります。DESDecrypt 構文の例は、この章の最後にあります。

パラメータの説明

表 23-3 と表 23-4 は、DESDecrypt 構文のパラメータ、モード、型および説明の一覧です。

表 23-3 ロー・データ用の DESDecrypt パラメータ

パラメータ名	モード	型	説明
input	IN	RAW	復号化するデータ
key	IN	RAW	復号キー
decrypted_data	OUT	RAW	復号化されたデータ

表 23-4 文字列データ用の DESDecrypt パラメータ

パラメータ名	モード	型	説明
input_string	IN	VARCHAR2	復号化する文字列
key_string	IN	VARCHAR2	復号キー文字列
decrypted_string	OUT	VARCHAR2	復号化された文字列

PL/SQL DESDecrypt ファンクションに渡された入力データまたはキーに値が指定されていない場合は、ORA エラー 28231 「Obfuscation ツールキットへの入力が無効です。」が発生します。

DESDecrypt ファンクションに指定された入力データが 8 バイトの倍数でない場合は、ORA エラー 28232 「Obfuscation ツールキットに対する入力サイズが無効です。」が発生します。

注意： ORA-28233 は、DESDecrypt ファンクションには適用されません。

DESDecryption プロシージャの制限事項

復号化用の DES キーの長さは 56 ビットに固定されています。このキーの長さは変更できません。

注意： キーの長さの制限事項は、暗号化製品の輸出を統括している米国の条項における要件です。

DESEncryption と DESDecryption のコード例

次のコードは、参照用のサンプル PL/SQL プログラムです。コードのセグメントに番号を付け、そのコードの部分の説明するテキストを挿入しています。

```
DECLARE
    input_string          VARCHAR2(16) := 'tigertigertigert';
    raw_input             RAW(128) := UTL_RAW.CAST_TO_RAW(input_string);
    key_string            VARCHAR2(8) := 'scottscsco';
    raw_key               RAW(128) := UTL_RAW.CAST_TO_RAW(key_string);
    wrong_input_string    VARCHAR2(25) := 'not_a_multiple_of_8_bytes';
    wrong_raw_input       RAW(128) := UTL_RAW.CAST_TO_RAW(wrong_input_string);
    wrong_key_string      VARCHAR2(8) := 'scottscsco';
    wrong_raw_key         RAW(128) := UTL_RAW.CAST_TO_RAW(wrong_key_string);
    encrypted_raw         RAW(2048);
    encrypted_string      VARCHAR2(2048);
    double_encrypted_raw  RAW(2048);
    double_encrypted_string VARCHAR2(2048);
    decrypted_raw         RAW(2048);
    decrypted_string      VARCHAR2(2048);
    error_in_input_buffer_length EXCEPTION;
    PRAGMA EXCEPTION_INIT(error_in_input_buffer_length, -28232);
    INPUT_BUFFER_LENGTH_ERR_MSG VARCHAR2(100) :=
        '*** DES INPUT BUFFER NOT A MULTIPLE OF 8 BYTES - IGNORING EXCEPTION ***';
    double_encrypt_not_permitted EXCEPTION;
    PRAGMA EXCEPTION_INIT(double_encrypt_not_permitted, -28233);
    DOUBLE_ENCRYPTION_ERR_MSG VARCHAR2(100) :=
        '*** CANNOT DOUBLE ENCRYPT DATA - IGNORING EXCEPTION ***';

-- 1. Begin testing raw data encryption and decryption
BEGIN
    dbms_output.put_line('> ===== BEGIN TEST RAW DATA =====');
    dbms_output.put_line('> Raw input                                : ' ||
        UTL_RAW.CAST_TO_VARCHAR2(raw_input));
```

```

BEGIN
    dbms_obfuscation_toolkit.DSEncrypt(input => raw_input,
        key => raw_key, encrypted_data => encrypted_raw );
    dbms_output.put_line('> encrypted hex value          : ' ||
        rawtohex(encrypted_raw));
    dbms_obfuscation_toolkit.DESDecrypt(input => encrypted_raw,
        key => raw_key, decrypted_data => decrypted_raw);
    dbms_output.put_line('> Decrypted raw output          : ' ||
        UTL_RAW.CAST_TO_VARCHAR2(decrypted_raw));
    dbms_output.put_line('> ');
    if UTL_RAW.CAST_TO_VARCHAR2(raw_input) =
        UTL_RAW.CAST_TO_VARCHAR2(decrypted_raw) THEN
        dbms_output.put_line('> Raw DES Encryption and Decryption successful');
    END if;
EXCEPTION
    WHEN error_in_input_buffer_length THEN
        dbms_output.put_line('> ' || INPUT_BUFFER_LENGTH_ERR_MSG);
END;
dbms_output.put_line('> ');

-- 2. Begin testing raw data double encryption prevention
BEGIN
    dbms_output.put_line('> testing double encryption prevention');
    dbms_output.put_line('> ');
    dbms_obfuscation_toolkit.DSEncrypt(input => raw_input,
        key => raw_key, encrypted_data => encrypted_raw );
    dbms_output.put_line('> input hex value              : ' ||
        rawtohex(encrypted_raw));
    dbms_obfuscation_toolkit.DSEncrypt(
        input => encrypted_raw,
        key => raw_key,
        encrypted_data => double_encrypted_raw );
    dbms_output.put_line('> double encrypted hex value      : ' ||
        rawtohex(double_encrypted_raw));
EXCEPTION
    WHEN double_encrypt_not_permitted THEN
        dbms_output.put_line('> ' || DOUBLE_ENCRYPTION_ERR_MSG);
END;
dbms_output.put_line('> ');

-- 3. Begin testing wrong raw input length values for encrypt operation
BEGIN
    dbms_output.put_line('> Wrong Raw input for encryption : ' ||
        UTL_RAW.CAST_TO_VARCHAR2(wrong_raw_input));
    dbms_obfuscation_toolkit.DSEncrypt(
        input => wrong_raw_input,
        key => raw_key,

```

```

        encrypted_data => encrypted_raw );
        dbms_output.put_line('> encrypted hex value          : ' ||
                               rawtohex(encrypted_raw));
EXCEPTION
    WHEN error_in_input_buffer_length THEN
        dbms_output.put_line('> ' || INPUT_BUFFER_LENGTH_ERR_MSG);
END;
dbms_output.put_line('> ');

-- 4. Begin testing wrong raw input length values for decrypt operation
BEGIN
    -- testing wrong input values for decrypt operation
    dbms_output.put_line('> Wrong Raw input for decryption  : ' ||
                           UTL_RAW.CAST_TO_VARCHAR2(wrong_raw_input));
    dbms_obfuscation_toolkit.DESDecrypt
        (input => wrong_raw_input,
         key => raw_key,
         decrypted_data => decrypted_raw );
    dbms_output.put_line('> decrypted hex value  : '
                          || rawtohex(decrypted_raw));
EXCEPTION
    WHEN error_in_input_buffer_length THEN
        dbms_output.put_line('> ' || INPUT_BUFFER_LENGTH_ERR_MSG);
END;
dbms_output.put_line('> ');
dbms_output.put_line('> ===== END TEST RAW DATA =====');

-- 5. Begin testing string data encryption and decryption
dbms_output.put_line('> ===== BEGIN TEST STRING DATA =====');

BEGIN
    dbms_output.put_line('> input string                      : '
                          || input_string);
    dbms_obfuscation_toolkit.DSEncrypt(
        input_string => input_string,
        key_string => key_string,
        encrypted_string => encrypted_string );
    dbms_output.put_line('> encrypted hex value          : ' ||
                           rawtohex(UTL_RAW.CAST_TO_RAW(encrypted_string)));
    dbms_obfuscation_toolkit.DESDecrypt(
        input_string => encrypted_string,
        key_string => key_string,
        decrypted_string => decrypted_string );
    dbms_output.put_line('> decrypted string output      : ' ||
                           decrypted_string);
    if input_string = decrypted_string THEN
        dbms_output.put_line('> String DES Encryption and Decryption successful');
    end if;
END;

```

```

        END if;
    EXCEPTION
        WHEN error_in_input_buffer_length THEN
            dbms_output.put_line(' ' || INPUT_BUFFER_LENGTH_ERR_MSG);
    END;
    dbms_output.put_line('> ');

-- 6. Begin testing string data double encryption prevention
BEGIN
    dbms_output.put_line('> testing double encryption prevention');
    dbms_output.put_line('> ');
    dbms_obfuscation_toolkit.DESEncrypt(
        input_string => input_string,
        key_string => key_string,
        encrypted_string => encrypted_string );
    dbms_output.put_line('> input hex value          : ' ||
        rawtohex(UTL_RAW.CAST_TO_RAW(encrypted_string)));
    dbms_obfuscation_toolkit.DESEncrypt(
        input_string => encrypted_string,
        key_string => key_string,
        encrypted_string => double_encrypted_string );
    dbms_output.put_line('> double encrypted hex value      : ' ||
        rawtohex(UTL_RAW.CAST_TO_RAW(double_encrypted_string)));
    EXCEPTION
        WHEN double_encrypt_not_permitted THEN
            dbms_output.put_line('> ' || DOUBLE_ENCRYPTION_ERR_MSG);
    END;
    dbms_output.put_line('> ');

-- 7. Begin testing wrong string input length values for encrypr operation
BEGIN
    dbms_output.put_line('> testing wrong input values for encrypr operation');
    dbms_output.put_line('> Wrong Raw input for encryption  : ' ||
        wrong_input_string);
    dbms_obfuscation_toolkit.DESEncrypt(
        input_string => wrong_input_string,
        key_string => wrong_key_string,
        encrypted_string => encrypted_string );
    dbms_output.put_line('> encrypted hex value          : ' ||
        rawtohex(UTL_RAW.CAST_TO_RAW(encrypted_string)));
    EXCEPTION
        WHEN error_in_input_buffer_length THEN
            dbms_output.put_line('> ' || INPUT_BUFFER_LENGTH_ERR_MSG);
    END;
    dbms_output.put_line('> ');

-- 8. Begin testing wrong string input length values for decrypt operation

```

```

BEGIN
    -- testing wrong input values for decrypt operation
    dbms_output.put_line('> Wrong Raw input for encryption   : ' ||
        wrong_input_string);
    dbms_obfuscation_toolkit.DESDecrypt(
        input_string => wrong_input_string,
        key_string => wrong_key_string,
        decrypted_string => decrypted_string );
    dbms_output.put_line('> decrypted string output : ' || decrypted_string);
EXCEPTION
    WHEN error_in_input_buffer_length THEN
        dbms_output.put_line('> ' || INPUT_BUFFER_LENGTH_ERR_MSG);
END;
dbms_output.put_line('> ');
dbms_output.put_line('> ===== END TEST STRING DATA =====');
END;
/

```

DBMS_OFFLINE_OG

DBMS_OFFLINE_OG パッケージには、マスター・グループのオフライン・インスタンスエーションのためのパブリック API が含まれています。

注意： このプロシージャは、マルチマスター・レプリケーション環境でマスター表のオフライン・インスタンスエーションを実行するときに使用します。

[DBMS_OFFLINE_SNAPSHOT](#) パッケージにあるプロシージャ（スナップショットのオフライン・インスタンスエーションの実行に使用）または [DBMS_REPCAT_INSTANTIATE](#) パッケージにあるプロシージャ（配置テンプレートのインスタンス化に使用）とこのプロシージャを混同しないでください。それぞれのパッケージの使用方法は、該当の章を参照してください。

DBMS_OFFLINE_OG パッケージ

サブプログラムの要約

表 24-1 DBMS_OFFLINE_OG パッケージのサブプログラム

サブプログラム	説明
24-3 ページの BEGIN_INSTANTIATION プロシージャ	レプリケート・マスター・グループのオフライン・インスタンスエーションを開始します。
24-4 ページの BEGIN_LOAD プロシージャ	オフライン・インスタンスエーションの一部としてデータを新規マスター・サイトにインポートする間、トリガーを使用禁止にします。
24-5 ページの END_INSTANTIATION プロシージャ	レプリケート・マスター・グループのオフライン・インスタンスエーションを完了します。
24-7 ページの END_LOAD プロシージャ	オフライン・インスタンスエーションの一部としてデータを新規マスター・サイトにインポートした後、トリガーを再び使用可能にします。
24-8 ページの RESUME_SUBSET_OF_MASTERS プロシージャ	レプリケート・マスター・グループのオフライン・インスタンスエーション中に、新規サイトを除く既存のすべてのサイトでレプリケーション・アクティビティを再開します。

BEGIN_INSTANTIATION プロシージャ

このプロシージャは、レプリケート・マスター・グループのオフライン・インスタンスエーションを開始します。このプロシージャは、マスター定義サイトからコールする必要があります。

注意： このプロシージャは、マルチマスター・レプリケーション環境でマスター表のオフライン・インスタンスエーションを実行するときに使用します。

DBMS_OFFLINE_SNAPSHOT パッケージにあるプロシージャ（スナップショットのオフライン・インスタンスエーションの実行に使用）または DBMS_REPCAT_INSTANTIATE パッケージにあるプロシージャ（配置テンプレートのインスタンス化に使用）とこのプロシージャを混同しないでください。それぞれのパッケージの使用方法は、該当の章を参照してください。

構文

```
DBMS_OFFLINE_OG.BEGIN_INSTANTIATION (
    gname      IN   VARCHAR2,
    new_site   IN   VARCHAR2
    fname      IN   VARCHAR2);
```

パラメータ

表 24-2 BEGIN_INSTANTIATION プロシージャのパラメータ

パラメータ	説明
gname	新規サイトにレプリケートするオブジェクト・グループの名前。
new_site	オブジェクト・グループをレプリケートする新規サイトの完全修飾データベース名。
fname	内部使用のためのパラメータ。オラクル社カスタマ・サポート・センターから指示がない限り、このパラメータは設定しないでください。

例外

表 24-3 BEGIN_INSTANTIATION プロシージャの例外

例外	説明
badargument	オブジェクト・グループまたは新規マスター・サイト名が NULL または指定されていません。
dbms_repcat. nonmasterdef	このプロシージャは、マスター定義サイトからコールする必要があります。
sitealreadyexists	指定のサイトは、すでにこのオブジェクト・グループのマスター・サイトです。
wrongstate	マスター定義サイトのステータスは、QUIESCED である必要があります。
dbms_repcat. missingrepgroup	gname がレプリケート・マスター・グループとして存在しません。
dbms_repcat.missing_ flavor	この例外が発生した場合は、オラクル社カスタマ・サポート・センターに連絡してください。

BEGIN_LOAD プロシージャ

このプロシージャは、オフライン・インスタンスエーションの一部としてデータを新規マスター・サイトにインポートする間、トリガーを使用禁止にします。このプロシージャは、新規マスター・サイトからコールする必要があります。

注意： このプロシージャは、マルチマスター・レプリケーション環境でマスター表のオフライン・インスタンスエーションを実行するときに使用します。

DBMS_OFFLINE_SNAPSHOT パッケージにあるプロシージャ（スナップショットのオフライン・インスタンスエーションの実行に使用）または DBMS_REPCAT_INSTANTIATE パッケージにあるプロシージャ（配置テンプレートのインスタンス化に使用）とこのプロシージャを混同しないでください。それぞれのパッケージの使用方法は、該当の章を参照してください。

構文

```
DBMS_OFFLINE_OG.BEGIN_LOAD (  
  gname      IN   VARCHAR2,  
  new_site   IN   VARCHAR2);
```

パラメータ

表 24-4 BEGIN_LOAD プロシージャのパラメータ

パラメータ	説明
gname	インポートするメンバーのオブジェクト・グループ名
new_site	オブジェクト・グループのメンバーをインポートする新規サイトの完全修飾データベース名

例外

表 24-5 BEGIN_LOAD プロシージャの例外

例外	説明
badargument	オブジェクト・グループまたは新規マスター・サイト名が NULL または指定されていません。
wrongsite	このプロシージャは、新規マスター・サイトからコールする必要があります。
unknownsite	指定のサイトがオブジェクト・グループで認識されません。
wrongstate	新規マスター・サイトのステータスは、QUIESCED である必要があります。
dbms_repcat. missingrepgroup	gname がレプリケート・マスター・グループとして存在しません。

END_INSTANTIATION プロシージャ

このプロシージャは、レプリケート・マスター・グループのオフライン・インスタンスエーションを完了します。このプロシージャは、マスター定義サイトからコールする必要があります。

注意： このプロシージャは、マルチマスター・レプリケーション環境でマスター表のオフライン・インスタンスエーションを実行するときに使用します。

DBMS_OFFLINE_SNAPSHOT パッケージにあるプロシージャ（スナップショットのオフライン・インスタンスエーションの実行に使用）または DBMS_REPCAT_INSTANTIATE パッケージにあるプロシージャ（配置テンプレートのインスタンス化に使用）とこのプロシージャを混同しないでください。それぞれのパッケージの使用方法は、該当の章を参照してください。

構文

```
DBMS_OFFLINE_OG.END_INSTANTIATION (  
    gname      IN  VARCHAR2,  
    new_site   IN  VARCHAR2);
```

パラメータ

表 24-6 END_INSTANTIATION プロシージャのパラメータ

パラメータ	説明
gname	新規サイトにレプリケートするオブジェクト・グループの名前
new_site	オブジェクト・グループをレプリケートする新規サイトの完全修飾データベース名

例外

表 24-7 END_INSTANTIATION プロシージャの例外

例外	説明
badargument	オブジェクト・グループまたは新規マスター・サイト名が NULL または指定されていません。
dbms_repcat. nonmasterdef	このプロシージャは、マスター定義サイトからコールする必要があります。
unknownsite	指定のサイトがオブジェクト・グループで認識されません。
wrongstate	マスター定義サイトのステータスは、QUIESCED である必要があります。
dbms_repcat. missingrepgroup	gname がレプリケート・マスター・グループとして存在しません。

END_LOAD プロシージャ

このプロシージャは、オフライン・インスタンスエーションの一部としてデータを新規マスター・サイトにインポートした後、トリガーを再び使用可能にします。このプロシージャは、新規マスター・サイトからコールする必要があります。

注意： このプロシージャは、マルチマスター・レプリケーション環境でマスター表のオフライン・インスタンスエーションを実行するときに使用します。

DBMS_OFFLINE_SNAPSHOT パッケージにあるプロシージャ（スナップショットのオフライン・インスタンスエーションの実行に使用）または DBMS_REPCAT_INSTANTIATE パッケージにあるプロシージャ（配置テンプレートのインスタンス化に使用）とこのプロシージャを混同しないでください。それぞれのパッケージの使用方法は、該当の章を参照してください。

構文

```
DBMS_OFFLINE_OG.END_LOAD (  
    gname      IN   VARCHAR2,  
    new_site   IN   VARCHAR2  
    fname      IN   VARCHAR2);
```

パラメータ

表 24-8 END_LOAD プロシージャのパラメータ

パラメータ	説明
gname	インポートを終了したメンバーのオブジェクト・グループの名前。
new_site	オブジェクト・グループのメンバーをインポートした新規サイトの完全修飾データベース名。
fname	内部使用のためのパラメータ。オラクル社カスタマ・サポート・センターから指示がない限り、このパラメータは設定しないでください。

例外

表 24-9 END_LOAD プロシージャの例外

例外	説明
badargument	オブジェクト・グループまたは新規マスター・サイト名が NULL または指定されていません。
wrongsite	このプロシージャは、新規マスター・サイトからコールする必要があります。
unknownsite	指定のサイトがオブジェクト・グループで認識されません。
wrongstate	新規マスター・サイトのステータスは、QUIESCED である必要があります。
dbms_repcat. missingrepgroup	gname がレプリケート・マスター・グループとして存在しません。
dbms_repcat.flavor_ noobject	この例外が発生した場合は、オラクル社カスタマ・サポート・センターに連絡してください。
dbms_repcat.flavor_ contains	この例外が発生した場合は、オラクル社カスタマ・サポート・センターに連絡してください。

RESUME_SUBSET_OF_MASTERS プロシージャ

このプロシージャは、レプリケート・マスター・グループのオフライン・インスタンスエーション中に、新規サイトを除く既存のすべてのサイトでレプリケーション・アクティビティを再開します。このプロシージャは、マスター定義サイトからコールする必要があります。

注意： このプロシージャは、マルチマスター・レプリケーション環境でマスター表のオフライン・インスタンスエーションを実行するときに使用します。

DBMS_OFFLINE_SNAPSHOT パッケージにあるプロシージャ（スナップショットのオフライン・インスタンスエーションの実行に使用）または DBMS_REPCAT_INSTANTIATE パッケージにあるプロシージャ（配置テンプレートのインスタンス化に使用）とこのプロシージャを混同しないでください。それぞれのパッケージの使用方法は、該当の章を参照してください。

構文

```
DBMS_OFFLINE_OG.RESUME_SUBSET_OF_MASTERS (  
  gname      IN  VARCHAR2,  
  new_site   IN  VARCHAR2  
  override   IN  BOOLEAN := FALSE);
```

パラメータ

表 24-10 RESUME_SUBSET_OF_MASTERS プロシージャのパラメータ

パラメータ	説明
gname	新規サイトにレプリケートするオブジェクト・グループの名前。
new_site	オブジェクト・グループをレプリケートする新規サイトの完全修飾データベース名。
override	TRUE の場合、保留中の RepCat 管理要求は無視され、通常のレプリケーション・アクティビティが各マスターで可能な限り早くリストアされます。override パラメータは、緊急の状況でのみ TRUE に指定してください。 このパラメータが FALSE の場合は、各マスターの gname に対する保留中の RepCat 管理要求がない場合のみ、各マスターで通常のレプリケーション・アクティビティがリストアされます。

例外

表 24-11 RESUME_SUBSET_OF_MASTERS プロシージャの例外

例外	説明
badargument	オブジェクト・グループまたは新規マスター・サイト名が NULL または指定されていません。
dbms_repcat. nonmasterdef	このプロシージャは、マスター定義サイトからコールする必要があります。
unknownsite	指定のサイトがオブジェクト・グループで認識されません。
wrongstate	マスター定義サイトのステータスは、QUIESCED である必要があります。
dbms_repcat. missingrepgroup	gname がレプリケート・マスター・グループとして存在しません。

DBMS_OFFLINE_SNAPSHOT

DBMS_OFFLINE_SNAPSHOT パッケージには、スナップショットのオフライン・インスタンス化のためのパブリック API が含まれています。

注意： このプロシージャは、スナップショットのオフライン・インスタンス化の実行時に使用されます。

[DBMS_OFFLINE_OG](#) パッケージにあるプロシージャ（マスター表のオフライン・インスタンス化の実行に使用）または [DBMS_REPCAT_INSTANTIATE](#) パッケージにあるプロシージャ（配置テンプレートのインスタンス化に使用）とこのプロシージャを混同しないでください。それぞれのパッケージの使用方法は、該当の章を参照してください。

DBMS_OFFLINE_SNAPSHOT パッケージ

サブプログラムの要約

表 25-1 DBMS_OFFLINE_SNAPSHOT パッケージのサブプログラム

サブプログラム	説明
25-3 ページの BEGIN_LOAD プロシージャ	オフライン・インスタンスエーションの一部として新規スナップ ショットをインポートするために、スナップショット・サイトを準備 します。
25-5 ページの END_LOAD プロシージャ	スナップショットのオフライン・インスタンスエーションを完了しま す。

BEGIN_LOAD プロシージャ

このプロシージャは、オフライン・インスタンスエーションの一部として新規スナップショットをインポートするために、スナップショット・サイトを準備します。このプロシージャは、新規スナップショットのスナップショット・サイトからコールする必要があります。

注意： このプロシージャは、スナップショットのオフライン・インスタンスエーションの実行時に使用されます。

DBMS_OFFLINE_OG パッケージにあるプロシージャ（マスター表のオフライン・インスタンスエーションの実行に使用）または DBMS_REPCAT_INSTANTIATE パッケージにあるプロシージャ（配置テンプレートのインスタンス化に使用）とこのプロシージャを混同しないでください。それぞれのパッケージの使用方法は、該当の章を参照してください。

構文

```
DBMS_OFFLINE_SNAPSHOT.BEGIN_LOAD (  
    gname                IN    VARCHAR2,  
    sname                IN    VARCHAR2,  
    master_site          IN    VARCHAR2,  
    snapshot_onsame      IN    VARCHAR2,  
    storage_c            IN    VARCHAR2 := '',  
    comment              IN    VARCHAR2 := '',  
    min_communication    IN    BOOLEAN  := TRUE);
```

パラメータ

表 25-2 BEGIN_LOAD プロシージャのパラメータ

パラメータ	説明
gname	オフライン・インスタンスエーションを使用して作成するスナップショットのオブジェクト・グループ名。
sname	新規スナップショットに対するスキーマの名前。
master_site	スナップショットのマスター・サイトの完全修飾データベース名。
snapshot_otype	マスター・サイトで作成される一時スナップショット名。
storage_c	スナップショット・サイトで新規スナップショットを作成するときに使用する記憶域オプション。
comment	ユーザー・コメント。
min_communication	TRUE の場合は、更新文で列を変更する場合に限り、更新トリガーによって列の新しい値が送信されます。また、その列がキー列または変更された列グループ内の列の場合、更新トリガーはその列の元の値も送信します。

例外

表 25-3 BEGIN_LOAD プロシージャの例外

例外	説明
badargument	オブジェクト・グループ、スキーマ、マスター・サイトまたはスナップショット名が NULL または指定されていません。
dbms_repcat. missingrepgroup	gname がレプリケート・マスター・グループとして存在しません。
missingremotesnap	指定のスナップショットが指定のマスター・サイトで見つかりません。
dbms_repcat. missingschema	指定のスキーマは存在しません。
snaptabmismatch	マスターにあるスナップショットのベース表名とスナップショットが合致しません。

END_LOAD プロシージャ

このプロシージャは、スナップショットのオフライン・インスタンスエーションを完了します。このプロシージャは、新規スナップショットのスナップショット・サイトからコールする必要があります。

注意： このプロシージャは、スナップショットのオフライン・インスタンスエーションの実行時に使用されます。

DBMS_OFFLINE_OG パッケージにあるプロシージャ（マスター表のオフライン・インスタンスエーションの実行に使用）または DBMS_REPCAT_INSTANTIATE パッケージにあるプロシージャ（配置テンプレートのインスタンス化に使用）とこのプロシージャを混同しないでください。それぞれのパッケージの使用方法は、該当の章を参照してください。

構文

```
DBMS_OFFLINE_SNAPSHOT.END_LOAD (
    gname          IN  VARCHAR2,
    sname          IN  VARCHAR2,
    snapshot_ename IN  VARCHAR2);
```

パラメータ

表 25-4 END_LOAD プロシージャのパラメータ

パラメータ	説明
gname	オフライン・インスタンスエーションを使用して作成するスナップショットのオブジェクト・グループ名
sname	新規スナップショットに対するスキーマの名前
snapshot_ename	スナップショット名

例外

表 25-5 END_LOAD プロシージャの例外

例外	説明
badargument	オブジェクト・グループ、スキーマまたはスナップショット名が NULL または指定されていません。
dbms_repcat. missingrepgroup	gname がレプリケート・マスター・グループとして存在しません。
dbms_repcat. nonsnapshot	このプロシージャは、スナップショット・サイトからコールする必要があります。

DBMS_OLAP パッケージは、マテリアライズド・ビュー分析の収集および任意の PL/SQL プログラムからコールできるアドバイザ機能を提供します。

関連項目： これらの機能の使用方法は、『Oracle8i パフォーマンスのための設計およびチューニング』マニュアルを参照してください。

要件

DBMS_OLAP は、構造化された統計情報（ファクト表やディメンション表のカーディナリティ、およびディメンション・レベル列、結合キー列、ファクト表キー列の別個のカーディナリティ）を利用し、必要に応じて、Oracle Trace で収集したマテリアライズド・ビューの管理イベントについてのワークロード統計を利用します。Oracle Trace で生成したワークロード表がある場合は、現行のデフォルト・スキーマに格納する必要があります。

エラー・メッセージ

表 26-1 DBMS_OLAP エラー・メッセージ

エラー・コード	説明
ORA-30476	PLAN_TABLE はユーザーのスキーマには存在しません。
ORA-30477	入力 select_clause の指定が正しくありません。
ORA-30478	指定されたディメンションは存在しません。
ORA-30479	サマリー・アドバイザ・エラーです。詳細は <QSM メッセージを参照。
QSM-00501	サマリー・アドバイザ環境の初期化ができません。
QSM-00502	OCI エラー。
QSM-00503	メモリー不足です。
QSM-00504	内部エラー。
QSM-00505	<parse_entitiy_name> - <error_description> 構文エラー。
QSM-00506	ファクト表が見つかりません。
QSM-00507	ディメンションが見つかりません。
QSM-00508	統計情報が表 / 列にありません。
QSM-00509	無効なパラメータ <parameter_name>。
QSM-00510	統計情報がサマリーにありません。
QSM-00511	無効なファクト・フィルタで指定されています。
QSM-00512	無効なサマリーがリテンション・リストに指定されています。
QSM-00513	ワークロード表が 1 つまたは両方ありません。

サブプログラムの要約

表 26-2 DBMS_OLAP パッケージのサブプログラム

サブプログラム	説明
26-3 ページの ESTIMATE_SUMMARY_SIZE プロシージャ	作成するマテリアライズド・ビューのサイズを、バイトと行で見積ります。
26-4 ページの EVALUATE_UTILIZATION プロシージャ	既存の各マテリアライズド・ビューの使用率を測定します。
26-4 ページの EVALUATE_UTILIZATION_W プロシージャ	既存の各マテリアライズド・ビューの使用率を測定します。
26-5 ページの RECOMMEND_MV プロシージャ	作成、保持または削除するマテリアライズド・ビューについて、推奨事項のセットを生成します。
26-7 ページの RECOMMEND_MV_W プロシージャ	作成、保持または削除するマテリアライズド・ビューについて、推奨事項のセットを生成します。
26-9 ページの VALIDATE_DIMENSION プロシージャ	ディメンションで指定した関連が正しいことを検証します。

ESTIMATE_SUMMARY_SIZE プロシージャ

このプロシージャでは、作成するマテリアライズド・ビューのサイズをバイトと行で見積ります。

構文

```
DBMS_OLAP.ESTIMATE_SUMMARY_SIZE (  
    stmt_id      IN  VARCHAR2,  
    select_clause IN  VARCHAR2,  
    num_rows     OUT NUMBER,  
    num_bytes    OUT NUMBER);
```

パラメータ

表 26-3 ESTIMATE_SIZE プロシージャのパラメータ

パラメータ	説明
stmt_id	EXPLAIN PLAN で文を識別するために使用する任意の文字列
select_clause	分析用の SELECT 文
num_rows	推測カーディナリティ
num_bytes	推測バイト数

EVALUATE_UTILIZATION プロシージャ

このプロシージャでは、仮定されるワークロードからのマテリアライズド・ビューの使用統計に基づいて、既存の各マテリアライズド・ビューの使用率を測定します。このプロシージャでは、明示しないパラメータが必要です。

出力は [MVIEW\\$_EVALUATIONS](#) 表に含まれます。この表にすでに行が含まれていた場合は、最初に切り捨てられます。

構文

```
DBMS_OLAP.EVALUATE_UTILIZATION;
```

パラメータ

表 26-4 EVALUATE_UTILIZATION プロシージャのパラメータ

パラメータ	説明
MVIEW\$_EVALUATIONS	既存の各マテリアライズド・ビューの使用率について評価します。 すでに行が含まれている場合は、最初に切り捨てられます。 このパラメータは、プロシージャがコールされるとそのプロシージャに提供されるため、暗黙的なパラメータです。

EVALUATE_UTILIZATION_W プロシージャ

このプロシージャでは、ワークロードから収集したマテリアライズド・ビューの使用統計に基づいて、既存の各マテリアライズド・ビューの使用率を測定します。次に説明するように、ワークロードはデフォルト・スキーマにある表に含める必要があります。

このプロシージャは、[WORK\\$_IDEAL_MVIEW](#) と [WORK\\$_MVIEW_USAGE](#) のビューも作成します。

関連項目： 26-10 ページの「[DBMS_OLAP インタフェース表](#)」

構文

```
DBMS_OLAP.EVALUATE_UTILIZATION_W;
```

パラメータ

表 26-5 EVALUATE_UTILIZATION_W プロシージャのパラメータ

パラメータ	I/O	説明
MVIEW\$_EVALUATIONS	OUT	既存の各マテリアライズド・ビューの使用率についての評価を戻します。すでに行が含まれている場合は、最初に切り捨てられます。 このパラメータは、プロシージャがコールされるとそのプロシージャに提供されるため、暗黙的なパラメータです。
V_192216243_F_5_E_14_8_1	IN	Oracle Trace でロギングされたワークロード要求の表。 このパラメータは、プロシージャがコールされるとそのプロシージャに提供されるため、暗黙的なパラメータです。
V_192216243_F_5_E_15_8_1	IN	Oracle Trace でロギングされたマテリアライズド・ビューの使用方法の表。 このパラメータは、プロシージャがコールされるとそのプロシージャに提供されるため、暗黙的なパラメータです。

RECOMMEND_MV プロシージャ

このプロシージャは、作成、保持または削除するマテリアライズド・ビューについて推奨事項のセットを生成します。これは ANALYZE で収集された表と列カーディナリティ統計情報の分析に基づいて行われます。

この推奨事項は、仮定されるワークロードに基づいています。この負荷では、データ・ウェアハウスで可能なすべての問合せに等しいウエイトが置かれています。このプロシージャでは、Oracle Trace で収集したワークロード統計表を必要としたり、使用することはありませんが、表が存在してもプロシージャは機能します。

ディメンションがすでに作成されている必要があり、ディメンションとファクト表をリンクする外部キー制約が存在している必要があります。

仮定されるワークロードを持つマテリアライズド・ビューの推奨事項は、使用方法が簡単であることが不可欠な DBA なしの環境に対しては適切ですが、デフォルト・スキーマでワークロードが参照できる場合は、それを使用する必要があります。

関連項目： ワークロードによる分析については、26-7 ページの「[RECOMMEND_MV_W プロシージャ](#)」を参照してください。

構文

```
DBMS_OLAP.RECOMMEND_MV (  
    fact_table_filter IN VARCHAR2,  
    storage_in_bytes  IN NUMBER,  
    retention_list     IN VARCHAR2,  
    retention_pct      IN NUMBER := 50);
```

パラメータ

表 26-6 RECOMMEND_MV プロシージャのパラメータ

パラメータ	説明
fact_table_filter	分析するファクト表名のカンマで区切られたリスト。すべてのファクト表を分析する場合は NULL です。
storage_in_bytes	マテリアライズド・ビューの格納に使用できる最大記憶域（バイト）。 負の数は指定できません。
retention_list	マテリアライズド・ビュー表名のカンマで区切られたリスト。 このリストに表示されるマテリアライズド・ビューに対して、削除の推奨は行いません。
retention_pct	実際または仮定されるワークロードに基づき、保持する必要がある既存のマテリアライズド・ビュー記憶域を 0 ～ 100 パーセントで指定します。 使用率で順位付けした累積領域が指定の保存しきい値内の場合（または retention_list に明示的にリストされている場合）、マテリアライズド・ビューは保持されます。使用率が NULL のマテリアライズド・ビュー（たとえば、非ディメンショナルのマテリアライズド・ビュー）は、常に保持されます。

パラメータ

表 26-7 RECOMMEND_MV プロシージャのパラメータ

パラメータ	説明
MVIEWS\$_RECOMMENDATION	マテリアライズド・ビューの作成に必要なサイズ見積りと SQL を含めて、作成された推奨事項を戻します。 このパラメータは、プロシージャがコールされるとそのプロシージャに提供されるため、暗黙的なパラメータです。

RECOMMEND_MV_W プロシージャ

このプロシージャは、作成、保持または削除するマテリアライズド・ビューについて推奨事項のセットを生成します。これは、ワークロード（Oracle Trace で収集）の情報と ANALYZE で収集された表と列カーディナリティ統計情報の分析に基づいて行われます。

RECOMMEND_MV_W プロシージャでは、ユーザーが ANALYZE を実行して表と列カーディナリティ統計情報を収集し、ワークロード統計情報を収集してフォーマットし、ディメンションを作成していることが要求されます。

ワークロードはそのワークロードでの各要求件数を判断するために集計されます。この件数は最適化プロセスでウエイト付け要因として使用されます。

指定されたファクト表を含むすべてのディメンショナル・マテリアライズド・ビューの領域は、ワークロード全体のパフォーマンスを最適化するマテリアライズド・ビューのセットを示します。

このプロシージャは、[WORK\\$_IDEAL_MVIEW](#) と [WORK\\$_MVIEW_USAGE](#) のビューも作成します。

関連項目： 26-10 ページの「[DBMS_OLAP インタフェース表](#)」

構文

```
DBMS_OLAP.RECOMMEND_MV_W (  
    fact_table_filter IN  VARCHAR2,  
    storage_in_bytes  IN  NUMBER,  
    retention_list     IN  VARCHAR2,  
    retention_pct      IN  NUMBER := 80);
```

パラメータ

表 26-8 RECOMMEND_MV_W プロシージャのパラメータ

パラメータ	説明
fact_table_filter	分析するファクト表名のカンマで区切られたリスト。すべてのファクト表を分析する場合は NULL になります。
storage_in_bytes	マテリアライズド・ビューの格納に使用できる最大記憶域（バイト）。負の数は指定できません。
retention_list	マテリアライズド・ビュー表名のカンマで区切られたリスト。このリストに表示されるマテリアライズド・ビューに対して、削除の推奨は行いません。
retention_pct	実際または仮定されるワークロードに基づき、保持する必要がある既存のマテリアライズド・ビュー記憶域を 0 ～ 100 パーセントで指定します。 使用率で順位付けした累積領域が指定の保存しきい値内の場合（または retention_list に明示的にリストされている場合）、マテリアライズド・ビューは保持されます。使用率が NULL のマテリアライズド・ビュー（たとえば、非ディメンショナルのマテリアライズド・ビュー）は、常に保持されます。

パラメータ

表 26-9 RECOMMEND_MV_W プロシージャのパラメータ

パラメータ	I/O	説明
MVIEW\$_RECOMMENDATION	OUT	マテリアライズド・ビューの作成に必要なサイズ見積りと SQL を含めて、作成された推奨事項を戻します。 このパラメータは、プロシージャがコールされるとそのプロシージャに提供されるため、暗黙的なパラメータです。
V_192216243_F_5_E_14_8_1 (required)	IN	Oracle Trace でロギングされたワークロード要求の表。 このパラメータは、プロシージャがコールされるとそのプロシージャに提供されるため、暗黙的なパラメータです。
V_192216243_F_5_E_15_8_1 (required)	IN	Oracle Trace でロギングされたマテリアライズド・ビューの使用方法の表。 このパラメータは、プロシージャがコールされるとそのプロシージャに提供されるため、暗黙的なパラメータです。

VALIDATE_DIMENSION プロシージャ

このプロシージャでは、既存のディメンション・オブジェクトで指定されている階層関係と属性関係、および結合関係が正しいことを検証します。これにより、参照整合性が保守されていることを迅速に確認できます。

構文

```
DBMS_OLAP.VALIDATE_DIMENSION (  
    dimension_name  VARCHAR2,  
    incremental     BOOLEAN := TRUE,  
    check_nulls     BOOLEAN := FALSE);
```

パラメータ

表 26-10 VALIDATE_DIMENSION プロシージャのパラメータ

パラメータ	説明
dimension_name	分析するディメンション名。
incremental	TRUE の場合は、このディメンションの表の sumdelta\$ 表で指定された行に対してのみ、テストが実行されます。そうでない場合は、すべての行をチェックします。
check_nulls	TRUE の場合は、すべてのレベルの列が NULL でないことを検証します。そうでない場合、このチェックは省略されます。 NOT NULL 制約などの別の方法で NULL でないことを保証できる場合は、FALSE を指定します。

パラメータ

表 26-11 VALIDATE_DIMENSION プロシージャのパラメータ

パラメータ	I/O	説明
MVIEW\$_EXCEPTIONS	OUT	名前付きディメンションで参照し、ディメンションの整合性に違反している表の行を戻します。 各行は、例外に関する表と ROWID を識別します。 このパラメータは、プロシージャがコールされるとそのプロシージャに提供されるため、暗黙的なパラメータです。

DBMS_OLAP インタフェース表

MVIEW\$_RECOMMENDATION

この表は、RECOMMEND_MV プロシージャまたは RECOMMEND_MV_W プロシージャで作成された推奨事項を示します。

表 26-12 MVIEW\$_RECOMMENDATION

列名	型	制約	説明
recommendation_number	INTEGER	主キー > 0	この推奨事項に対する一意の識別子。
recommended_action	VARCHAR2 (6)	CREATE、RETAIN または DROP	この行で記述され、マテリアライズド・ビューを使用して行う処理。
summary_owner	VARCHAR (30)	アクションが CREATE の場合は未定義	DROP または RETAIN を行う既存のマテリアライズド・ビューの所有者。
summary_name	VARCHAR2 (30)	アクションが CREATE の場合は未定義	DROP または RETAIN を行う既存のマテリアライズド・ビュー名。
group_by_columns	VARCHAR2 (2000)	アクションが CREATE 以外は NULL	マテリアライズド・ビューの GROUP BY 句に記述される列参照のカンマで区切られたリスト。 この列参照は、SELECT リストにも記述されています。
where_clause	VARCHAR2 (2000)	アクションが CREATE 以外は NULL	内部等価結合の AND で区切られたリスト。内部等価結合は、マテリアライズド・ビューの WHERE 句に記述されます。
from_clause	VARCHAR2 (2000)	アクションが CREATE 以外は NULL	リレーション名のカンマで区切られたリスト。

表 26-12 MVIEW\$_RECOMMENDATION

列名	型	制約	説明
measures_list	VARCHAR2 (2000)	アクションが DROP の場合は NULL、CREATE の 場合は NULL 以外	<p><groupingFunction> (<expression>) のカンマで区 切られたリスト。マテリアライ ズド・ビューの SELECT リスト に記述されます。</p> <p>recommended_action が CREATE の場合、このフィール ドは、作成したマテリアライズ ド・ビューに記述される必要が あるメジャー・リストです。</p> <p>recommended_action が RETAIN で、このフィールドが NULL 以外の場合、このフィール ドはマテリアライズド・ビュー に追加される必要があるメ ジャー・リストです。</p>
storage_in_bytes	NUMBER	>= 0	実際または見積りした記憶域 (バイト)。
pct_performance_ gain	NUMBER		以前の推奨事項をすべて受け入 れたと仮定し、当初の状態と比 較して、この推奨事項を受け入 れることによって期待できるパ フォーマンスの改善率。不明な 場合は NULL。
benefit_to_cost_ ratio	NUMBER		マテリアライズド・ビューのサ イズ (バイト) について、パ フォーマンスの改善率。不明な 場合は NULL。

各行には、推奨されるアクション (CREATE、RETAIN または DROP)、および次のいずれかの形式によるマテリアライズド・ビューの説明が含まれます。

- マテリアライズド・ビューがすでに存在する場合は (推奨されるアクションが RETAIN または DROP)、マテリアライズド・ビュー名。または、
- マテリアライズド・ビューを作成するための SQL SELECT 式 (推奨されるアクションが CREATE の場合)。SQL SELECT 式は、次の 4 つの部分で構成されています。

1. マテリアライズド・ビューの GROUP BY 句に記述される列名のカンマで区切られたリスト。この列は、SELECT リストにも記述されています。
2. WHERE 句の内容で、内部等価結合の AND で区切られたリスト。
3. FROM 句の内容で、リレーション名のカンマで区切られたリスト。
4. メジャーのリストで、<grouping function> (<expression>) のカンマで区切られたリスト。マテリアライズド・ビューの SELECT リストに記述されています。

次に、SQL SELECT 式の作成方法の例を示します。

```
SELECT <group by columns>, <measures list>
  FROM <from clause>
  WHERE <where clause>
  GROUP BY <group by columns>;
```

各行には、既存のマテリアライズド・ビューが占有する領域（バイト）を示す `storage_in_bytes` フィールドも含まれます。新規に推奨されるマテリアライズド・ビューの場合、このフィールドは、マテリアライズド・ビューが占有する領域サイズの見積りを示します。

各マテリアライズド・ビューについて、次の 2 つのパフォーマンス・メトリックが提供されます。

<code>pct_performance_gain</code>	以前の推奨事項をすべて受け入れたと仮定し、この推奨事項の受入れによって期待できるパフォーマンスの改善率
<code>benefit_to_cost_ratio</code>	マテリアライズド・ビューのサイズについて、パフォーマンスの改善率

推奨事項は、`recommendation_number` 別に、最も改善度が高いものから順に並べられます。そして、段階的な改善度は、リストにある以前の推奨事項をすべて受け入れたと仮定して計算されます。

MVIEW\$_EVALUATIONS

この表は、[EVALUATE_UTILIZATION](#) プロシージャで作成された評価を示します。この表の行数は、現行のデータベースにあるマテリアライズド・ビューの数と同じです。

表 26-13 MVIEW\$_EVALUATIONS

列名	型	制約	説明
summary_owner	VARCHAR2 (30)	主キー	このデータベースにある既存のマテリアライズド・ビューの所有者。
summary_name	VARCHAR2 (30)	主キー	このデータベースにある既存のマテリアライズド・ビュー名。
rank	INTEGER	≥ 1	benefit_to_cost_ratio の降順で示されるこのマテリアライズド・ビューの順位。
storage_in_bytes	NUMBER	≥ 0	マテリアライズド・ビューのサイズ (バイト)。
frequency	INTEGER	≥ 0 、または NULL	このマテリアライズド・ビューがワークロードに記述される回数。
cumulative_benefit	NUMBER	≥ 0 、または NULL	マテリアライズド・ビューがクエリー・リライトで使用されるたびに、マテリアライズド・ビューとその改善度がワークロードにロギングされます。 このフィールドで、各マテリアライズド・ビューに関する改善度 (純縮小要因) を合計します。
benefit_to_cost_ratio	NUMBER		storage_in_bytes > 0 の場合は、cumulative_benefit 対 storage_in_bytes の割合を示し、それ以外の場合は NULL です。

注意： この表の benefit_to_cost_ratio で使用する方法は、[MVIEW\\$_RECOMMENDATION](#) 表の benefit_to_cost_ratio を計算する方法と類似していますが、同一ではありません。

WORK\$_IDEAL_MVIEW

このビューは、実際のまたは可能性があるクエリー・リライトに対応した表 V_192216243_F_5_E_14_8_1 に基づいています。

表 26-14 WORK\$_IDEAL_MVIEW

列名	型	制約	説明
sql_text_hash	NUMBER	NULL 以外	SQL 文署名。
lib_cache_addr	VARCHAR2 (16)	NULL 以外	マテリアライズド・ビューの使用を特定のカーソルに関連付けます。
group_by_columns	VARCHAR2 (2000)	NULL 以外	架空のマテリアライズド・ビューの GROUP BY 句に記述されている修飾列参照のカンマで区切られたリスト。この列参照は、SELECT リストにも記述されています。
where_clause	VARCHAR2 (2000)		内部等価結合の 'AND' で区切られたリスト。内部等価結合は、架空のマテリアライズド・ビューの WHERE 句に記述されています。
from_clause	VARCHAR2 (2000)	NULL 以外	所有者と修飾リレーション名、および別名のカンマで区切られたリスト。
measure	VARCHAR2 (2000)		<groupingFunction> (<expression>) のカンマで区切られたリスト。架空のマテリアライズド・ビューの SELECT リストに記述されています。
idl_sum_flags	NUMBER		フラグ・ベクタ (4 バイト)。現在、詳細は未定義です。
summary_owner	VARCHAR2 (30)		クエリー・リライトで使用する既存のマテリアライズド・ビューの所有者。所有者がない場合は NULL。
summary_name	VARCHAR2 (30)		クエリー・リライトで使用する既存のマテリアライズド・ビュー名。ビュー名がない場合は NULL。
actual_benefit	NUMBER	>0	このマテリアライズド・ビューの使用による実際のパフォーマンスの改善度。

WORK\$_MVIEW_USAGE

このビューは、Oracle Trace の format 操作で生成される表 V_1992216243_F_5_E_15_8_1 に基づいています。この表の各行は、実際のクエリー・リライトに対応しています。

列名	型	制約	説明
sql_text_hash	NUMBER	NULL 以外	SQL 文署名。
lib_cache_addr	VARCHAR2(16)	NULL 以外	マテリアライズド・ビューの使用を特定のカーソルに関連付けます。

DBMS_ORACLE_TRACE_AGENT

DBMS_ORACLE_TRACE_AGENT パッケージは、システム・レベルのユーティリティを提供します。

セキュリティ

このパッケージは、sys 権限を持つユーザーのみアクセスできるようにデフォルト設定されています。ルーチンへのアクセスは、権限があるユーザーに実行を許可することにより制御できます。

注意： このパッケージは、DBA または Oracle TRACE Collection Agent にのみ権限を付与してください。

サブプログラムの要約

このパッケージに含まれているサブプログラムは、SET_ORACLE_TRACE_IN_SESSION のみです。

SET_ORACLE_TRACE_IN_SESSION プロシージャ

このプロシージャでは、ユーザー所有以外のデータベース・セッションに関する Oracle Trace データを収集します。Oracle TRACE は、(sid、serial#) で識別されるセッションで使用可能になります。これらの値は、v\$session から取得されます。

構文

```
DBMS_ORACLE_TRACE_AGENT.SET_ORACLE_TRACE_IN_SESSION (  
  sid                NUMBER    DEFAULT 0,  
  serial#            NUMBER    DEFAULT 0,  
  on_off IN          BOOLEAN    DEFAULT false,  
  collection_name IN VARCHAR2  DEFAULT '',  
  facility_name      IN VARCHAR2 DEFAULT '');
```

パラメータ

表 27-1 SET_ORACLE_TRACE_IN_SESSION プロシージャのパラメータ

パラメータ	説明
sid	セッション ID。
serial#	セッションのシリアル番号。
on_off	TRUE または FALSE。トレースをオンまたはオフにします。
collection_name	使用する Oracle TRACE コレクション名。
facility_name	使用する Oracle TRACE 機能名。

使用上の注意

コレクションが発生しない場合は、次の事項をチェックしてください。

- <facility_name> で識別されるサーバー・イベント・セット・ファイルの存在を確認してください。ファイルがフィールドに完全に指定されていない場合、ファイルは、初期化ファイルにある ORACLE_TRACE_FACILITY_PATH で識別されるディレクトリに配置されている必要があります。
- REGID.DAT、PROCESS.DAT および COLLECT.DAT の各ファイルが、Oracle Trace 管理ディレクトリに存在している必要があります。存在していない場合は、OTRCCREF 実行可能ファイルを実行してファイルを作成してください。

注意： Oracle8 では、PROCESS.DAT が FACILITY.DAT に変更されました。

- ストアド・プロシージャ・パッケージがデータベースに存在している必要があります。存在していない場合は、OTRCSVR.SQL ファイル（Oracle Trace または RDBMS 管理ディレクトリ内にあります）を実行してパッケージを作成します。
- ストアド・プロシージャに対する EXECUTE 権限がユーザーにあるかどうかをチェックします。

例

```
EXECUTE DBMS_ORACLE_TRACE_AGENT.SET_ORACLE_TRACE_IN_SESSION  
(8,12,TRUE, 'NEWCOLL', 'oracled');
```

DBMS_ORACLE_TRACE_USER

DBMS_ORACLE_TRACE_USER は、Oracle TRACE 機能へのパブリック・アクセスをコール側ユーザーに提供します。Oracle Trace ストアド・プロシージャを使用すると、ユーザー自身のセッションまたは別のセッションに対して Oracle Trace コレクションを起動できます。

サブプログラムの要約

このパッケージに含まれているサブプログラムは、SET_ORACLE_TRACE のみです。

SET_ORACLE_TRACE プロシージャ

このプロシージャは、ユーザー自身のデータベース・セッションに関する Oracle Trace データを収集します。

構文

```
DBMS_ORACLE_TRACE_USER.SET_ORACLE_TRACE (  
  on_off           IN BOOLEAN  DEFAULT false,  
  collection_name  IN VARCHAR2 DEFAULT '',  
  facility_name    IN VARCHAR2 DEFAULT '');
```

パラメータ

表 28-1 SET_ORACLE_TRACE プロシージャのパラメータ

パラメータ	説明
on_off	TRUE または FALSE。トレースをオンまたはオフにします。
collection_name	使用する Oracle TRACE コレクション名。
facility_name	使用する Oracle TRACE 機能名。

例

```
EXECUTE DBMS_ORACLE_TRACE_USER.SET_ORACLE_TRACE  
(TRUE, 'MYCOLL', 'oracle');
```

DBMS_OUTPUT

DBMS_OUTPUT パッケージによって、ストアド・プロシージャ、パッケージおよびトリガーからメッセージを送信できます。

このパッケージにある PUT と PUT_LINE プロシージャによって、別のトリガー、プロシージャまたはパッケージが読み込めるバッファに情報を設定できます。別の PL/SQL プロシージャまたは無名ブロックでは、GET_LINE プロシージャをコールして、バッファに入れた情報を表示できます。

GET_LINE をコールしない場合、または SQL*Plus や Enterprise Manager のスクリーンにメッセージを表示しない場合、バッファに入れたメッセージは無視されます。DBMS_OUTPUT パッケージは、PL/SQL デバッグ情報の表示に特に役立ちます。

注意： DBMS_OUTPUT を使用して送信するメッセージは、送信サブプログラムまたはトリガーが完了するまでは実際に送信されません。プロシージャの実行中に出力をフラッシュするメカニズムはありません。

セキュリティ

このスクリプトの最後にパブリック・シノニム（DBMS_OUTPUT）が作成され、このパッケージに対する EXECUTE 許可がパブリックに付与されます。

エラー

DBMS_OUTPUT サブプログラムでアプリケーション・エラー ORA-20000 が発生すると、出力プロシージャは次のエラーを戻すことができます。

表 29-1 DBMS_OUTPUT エラー

エラー	説明
ORU-10027:	Buffer overflow
ORU-10028:	Line length overflow

型

CHARARR 型は表の型です。

DBMS_OUTPUT の使用方法

トリガーによってデバッグ情報を印刷する場合があります。これを行うには、次のようにトリガーを指定します。

```
DBMS_OUTPUT.PUT_LINE('I got here:'||:new.col||' is the new value');
```

DBMS_OUTPUT パッケージを使用可能にした場合、この PUT_LINE はバッファに入れられ、トリガー文（トリガーを起動させる INSERT、DELETE または UPDATE が想定できます）の実行後、情報行を戻すことができます。次に例を示します。

```
BEGIN
  DBMS_OUTPUT.GET_LINE(:buffer, :status);
END;
```

これでスクリーンにバッファを表示できます。ステータスが 0（ゼロ）以外に戻るまで、GET_LINE へのコールを繰り返します。パフォーマンス向上のためには、行の配列を戻すことのできる GET_LINES へのコールを使用してください。

Enterprise Manager と SQL*Plus は SET SERVEROUTPUT ON コマンドをインプリメントして、INSERT、UPDATE、DELETE または無名の PL/SQL コール（これらのコールのみが、トリガーまたはストアド・プロシージャを実行させることができます）の発行後に、GET_LINE へのコールを行うかどうかを判断します。

サブプログラムの要約

表 29-2 DBMS_OUTPUT パッケージのサブプログラム

サブプログラム	説明
29-3 ページの ENABLE プロシージャ	メッセージの出力を使用可能にします。
29-4 ページの DISABLE プロシージャ	メッセージの出力を使用禁止にします。
29-5 ページの PUT および PUT_LINE プロシージャ	行をバッファに設定します。
29-5 ページの PUT および PUT_LINE プロシージャ	行の一部をバッファに設定します。
29-6 ページの NEW_LINE プロシージャ	PUT を使用して作成した行を終了します。
29-7 ページの GET_LINE および GET_LINES プロシージャ	バッファから 1 行、または行の配列を取り出します。

ENABLE プロシージャ

このプロシージャは、PUT、PUT_LINE、NEW_LINE、GET_LINE および GET_LINES へのコールを使用可能にします。DBMS_OUTPUT パッケージが使用可能でない場合は、これらのプロシージャへのコールは無視されます。

注意： Enterprise Manager または SQL*Plus の SERVEROUTPUT オプションを使用する場合は、このプロシージャをコールする必要はありません。

ENABLE へのコールが複数の場合、buffer_size は最大値が指定されます。最大サイズは 1,000,000 で、最小サイズは 2,000 です。

構文

```
DBMS_OUTPUT.ENABLE (  
    buffer_size IN INTEGER DEFAULT 20000);
```

パラメータ

表 29-3 ENABLE プロシージャのパラメータ

パラメータ	説明
buffer_size	バッファに設定する情報量（バイト）。

プラグマ

```
pragma restrict_references (enable,WNDS,RNDS);
```

エラー

表 29-4 ENABLE プロシージャのエラー

エラー	説明
ORA-20000:, ORU-10027:	バッファのオーバーフロー。バッファは <buffer_limit> バイトに制限されています。

DISABLE プロシージャ

このプロシージャは、PUT、PUT_LINE、NEW_LINE、GET_LINE および GET_LINES へのコールを使用禁止にして、残っている情報のバッファをパージします。

ENABLE と同様に、Enterprise Manager または SQL*Plus の SERVEROUTPUT オプションを使用する場合は、このプロシージャをコールする必要はありません。

構文

```
DBMS_OUTPUT.DISABLE;
```

パラメータ

なし

プラグマ

```
pragma restrict_references (disable,WNDS,RNDS);
```


PUT および PUT_LINE プロシージャ

PUT_LINE をコールして情報行全体をバッファに設定するか、または PUT を複数回コールして個別に情報行を作成することができます。両方のプロシージャはオーバーロードされていて、VARCHAR2、NUMBER または DATE 型の項目を受け入れてバッファに設定します。

すべての項目は、取り出されるときに VARCHAR2 に変換されます。NUMBER または DATE 型の項目を渡す場合は、その項目が取り出されるとき、デフォルトのフォーマットを使用して TO_CHAR でフォーマットされます。別のフォーマットを使用する場合は、その項目を VARCHAR2 として渡し、明示的にフォーマットする必要があります。

PUT_LINE をコールすると、指定した項目には行端マーカが自動的に付きます。PUT をコールして行を作成する場合は、NEW_LINE をコールして行端マーカを追加する必要があります。GET_LINE と GET_LINES は、改行文字で終了していない行は戻しません。

行がバッファ制限を超えた場合は、エラー・メッセージが発生します。

注意： PUT または PUT_LINE で作成した出力はバッファに入れられません。この出力は、それをバッファに入れた PL/SQL プログラム・ユニットがコール側に戻るまで取り出せません。

たとえば、Enterprise Manager または SQL*Plus は、PL/SQL プログラムが完了するまで DBMS_OUTPUT メッセージを表示しません。PL/SQL プログラム内で DBMS_OUTPUT バッファをフラッシュするメカニズムはありません。次に例を示します。

```
SQL> SET SERVER OUTPUT ON
SQL> BEGIN
      2 DBMS_OUTPUT.PUT_LINE ('hello');
      3 DBMS_LOCK.SLEEP (10);
      4 END;
```

構文

```
DBMS_OUTPUT.PUT      (item IN NUMBER);
DBMS_OUTPUT.PUT      (item IN VARCHAR2);
DBMS_OUTPUT.PUT      (item IN DATE);
DBMS_OUTPUT.PUT_LINE (item IN NUMBER);
DBMS_OUTPUT.PUT_LINE (item IN VARCHAR2);
DBMS_OUTPUT.PUT_LINE (item IN DATE);
```

パラメータ

表 29-5 PUT および PUT_LINE プロシージャのパラメータ

パラメータ	説明
item	バッファに設定する項目

エラー

表 29-6 PUT および PUT_LINE プロシージャのエラー

エラー	説明
ORA-20000, ORU-10027:	バッファのオーバーフロー。バッファは <buf_limit> バイトに制限されています。
ORA-20000, ORU-10028:	行の長さのオーバーフロー。1 行につき 255 バイトに制限されています。

NEW_LINE プロシージャ

このプロシージャは、行端マーカを設定します。GET_LINE は、改行で区切られた行を戻します。PUT_LINE または NEW_LINE へのすべてのコールによって、GET_LINE で戻される行が生成されます。

構文

```
DBMS_OUTPUT.NEW_LINE;
```

パラメータ

なし

エラー

表 29-7 NEW_LINE プロシージャのエラー

エラー	説明
ORA-20000, ORU-10027:	バッファのオーバーフロー。バッファは <buf_limit> バイトに制限されています。
ORA-20000, ORU-10028:	行の長さのオーバーフロー。1 行につき 255 バイトに制限されています。

GET_LINE および GET_LINES プロシージャ

単一行または行の配列をバッファから取り出すことができます。バッファに入れられた単一行の情報を取り出すには、GET_LINE プロシージャをコールします。サーバーへのコール数を減らすには、GET_LINES プロシージャをコールして、バッファから行の配列を取り出します。

Enterprise Manager または SQL*Plus を使用している場合は、特別の SET SERVEROUTPUT ON コマンドを使用して、この情報を自動的に表示できます。

GET_LINE または GET_LINES をコールしてから、次に PUT、PUT_LINE または NEW_LINE をコールする前に取り出されなかった行は、次のメッセージとの混同を避けるために廃棄されます。

構文

```
DBMS_OUTPUT.GET_LINE (  
    line    OUT VARCHAR2,  
    status  OUT INTEGER);
```

パラメータ

表 29-8 GET_LINE プロシージャのパラメータ

パラメータ	説明
line	最後の改行文字を除いて、バッファに入れられた単一行の情報を返します。最大長は 255 バイトです。
status	コールが正常に完了すると、ステータス 0（ゼロ）が戻ります。バッファにこれ以上行がない場合は、ステータス 1 が戻ります。

構文

```
DBMS_OUTPUT.GET_LINES (  
    lines      OUT CHARARR,  
    numlines  IN OUT INTEGER);
```

CHARARR は、VARCHAR2(255) の表です。

パラメータ

表 29-9 GET_LINES プロシージャのパラメータ

パラメータ	説明
lines	バッファに入れられた情報の行の配列を戻します。 配列内の各行の最大長は 255 バイトです。
numlines	バッファから取り出す行数。 プロシージャは、指定の行数を取り出した後、実際に取り出した 行数を戻します。この数が要求した行数より少ない場合は、バッ ファにそれ以上行がない場合です。

例

例 1 で示すように、DBMS_OUTPUT パッケージは、一般的に、ストアド・プロシージャとトリガーをデバッグするために使用されます。また、29-9 ページの例 2 で示すように、このパッケージを使用して、オブジェクトの情報を取り出し、その出力をフォーマットすることができます。

例 1 これは、従業員表を問い合せて、指定部門の給与合計を戻すファンクション例です。このファンクションには、次のように PUT_LINE プロシージャへのコールがいくつか含まれます。

```
CREATE FUNCTION dept_salary (dnum NUMBER) RETURN NUMBER IS
  CURSOR emp_cursor IS
    SELECT sal, comm FROM emp WHERE deptno = dnum;
  total_wages    NUMBER(11, 2) := 0;
  counter        NUMBER(10) := 1;
BEGIN

  FOR emp_record IN emp_cursor LOOP
    emp_record.comm := NVL(emp_record.comm, 0);
    total_wages := total_wages + emp_record.sal
      + emp_record.comm;
    DBMS_OUTPUT.PUT_LINE('Loop number = ' || counter ||
      ' ; Wages = ' || TO_CHAR(total_wages)); /* Debug line */
    counter := counter + 1; /* Increment debug counter */
  END LOOP;
  /* Debug line */
  DBMS_OUTPUT.PUT_LINE('Total wages = ' ||
    TO_CHAR(total_wages));
  RETURN total_wages;

END dept_salary;
```

EMP 表には、次の行が含まれているとします。

EMPNO	SAL	COMM	DEPT
1002	1500	500	20
1203	1000		30
1289	1000		10
1347	1000	250	20

ユーザーは、Enterprise Manager の SQL ワークシート入力ペインで次の文を実行するとします。

```
SET SERVEROUTPUT ON
VARIABLE salary NUMBER;
EXECUTE :salary := dept_salary(20);
```

出力ペインには、次の情報が表示されます。

```
Loop number = 1; Wages = 2000
Loop number = 2; Wages = 3250
Total wages = 3250
```

PL/SQL procedure successfully executed.

例 2 この例で、ユーザーはすでに EXPLAIN PLAN コマンドを使用して、ある文の実行計画に関する情報を取り出し、その情報を PLAN_TABLE に格納してあります。また、ユーザーはこの文に文 ID を割り当ててあります。EXPLAIN_OUT プロシージャの例では、表から情報を取り出し、出力をネスト形式でフォーマットして、SQL 文を処理するステップの順序を厳密に記述しています。

```

/*****
/* Create EXPLAIN_OUT procedure. User must pass STATEMENT_ID to */
/* to procedure, to uniquely identify statement.                */
*****/
CREATE OR REPLACE PROCEDURE explain_out
(statement_id IN VARCHAR2) AS

-- Retrieve information from PLAN_TABLE into cursor EXPLAIN_ROWS.
```

```
CURSOR explain_rows IS
    SELECT level, id, position, operation, options,
           object_name
    FROM plan_table
    WHERE statement_id = explain_out.statement_id
    CONNECT BY PRIOR id = parent_id
           AND statement_id = explain_out.statement_id
    START WITH id = 0
    ORDER BY id;

BEGIN

    -- Loop through information retrieved from PLAN_TABLE:

    FOR line IN explain_rows LOOP

        -- At start of output, include heading with estimated cost.

        IF line.id = 0 THEN
            DBMS_OUTPUT.PUT_LINE ('Plan for statement '
                                   || statement_id
                                   || ', estimated cost = ' || line.position);
        END IF;

        -- Output formatted information. LEVEL determines indention level.

        DBMS_OUTPUT.PUT_LINE (lpad(' ', 2*(line.level-1)) ||
                               line.operation || ' ' || line.options || ' ' ||
                               line.object_name);
    END LOOP;

END;
```

関連項目： [第 60 章「UTL_FILE」](#)

DBMS_PCLXUTIL

DBMS_PCLXUTIL パッケージは、パーティション単位でローカル索引を作成するためのパーティション内並列性を提供します。

関連項目： パーティションと索引に関しては、いくつかのルールがあります。詳細は、『Oracle8i 概要』と『Oracle8i 管理者ガイド』を参照してください。

パーティションごとに 1 つのスレーブ処理しか利用できません。DBMS_PCLXUTIL は、ローカル索引の作成でパーティション数によって並列度が制限されることを回避します。

DBMS_PCLXUTIL は DBMS_JOB パッケージを使用して、パーティション表のローカル索引を作成するための高い並列度を提供します。これは、バックグラウンド・プロセスを使用した (DBMS_JOB を使用) 非同期のパーティション間並列性と、パラレル問合せスレーブ処理を使用したパーティション内並列性の組合せで達成されます。

DBMS_PCLXUTIL は、レンジ・パーティション化とレンジ・ハッシュ・コンポジット・パーティション化の両方で機能します。

注意： レンジ・パーティション化の最小互換モードは 8.0 で、レンジ・ハッシュ・コンポジット・パーティション化の最小互換モードは 8i です。

DBMS_PCLXUTIL の使用方法

DBMS_PCLXUTIL パッケージは、次の DBA タスク中に使用できます。

1. ローカル索引の作成

プロシージャ BUILD_PART_INDEX は、ローカル索引のディクショナリ情報がすでに存在していることを前提としています。これは、UNUSABLE オプションを持つ CREATE INDEX SQL コマンドを発行して行います。

```
CREATE INDEX <idx_name> on <tab_name>(...) local(...) unusable;
```

これにより、索引作成で時間がかかるディクショナリ・エントリの作成を、索引自体を作成せずに行うことができます。そして、プロシージャ BUILD_PART_INDEX を起動することによって、指定の並列度でローカル索引の同時作成を行います。

```
EXECUTE dbms_pclxutil.build_part_index(4,4,<tab_name>,<idx_name>,FALSE);
```

コンポジット・パーティションについて、プロシージャは、そのコンポジット表のすべてのサブパーティションに対するローカル索引を自動的に作成します。

2. ローカル索引のメンテナンス

目的のパーティションを使用可能または使用禁止にマークすることによって、BUILD_PART_INDEX プロシージャもローカル索引の選択的な再作成を可能にします。force_opt パラメータは、これを上書きし、すべてのパーティションに対するローカル索引の作成方法を提供します。

```
ALTER INDEX <idx_name> local(...) unusable;
```

目的の（サブ）パーティション（使用禁止とマークされています）のみを再作成します。

```
EXECUTE dbms_pclxutil.build_part_index(4,4,<tab_name>,<idx_name>,FALSE);
```

force_opt = TRUE を使用して、すべての（サブ）パーティションを再作成します。

```
EXECUTE dbms_pclxutil.build_part_index(4,4,<tab_name>,<idx_name>,TRUE);
```

進捗レポートが作成され、プログラムの終了時に出力がスクリーンに表示されます（DBMS_OUTPUT パッケージは、最初にメッセージをバッファに書き込み、そのバッファをプログラムの終了時のみスクリーンにフラッシュするためです）。

制限事項

DBMS_PCLXUTIL パッケージは DBMS_JOB パッケージを使用しているため、DBMS_JOB に関する次の制限事項があることに注意してください。

- `job_queue_processes` と `job_queue_interval` 初期化パラメータについて、適切な値を決める必要があります。`BUILD_PART_INDEX()` をコールする前にジョブ・プロセスが開始されていない場合、パッケージは正しく機能しません。バックグラウンド・プロセスは、次の `init.ora` パラメータで指定されます。

```
job_queue_processes=n    #the number of background processes = n
job_queue_interval=m     #the processes wake-up every m seconds
```

- キューに入る同時ジョブの数に上限があり、この上限は、SNP[0..9] および SNP[A..Z] のマークが付けられたバックグラウンド・プロセスの数で決まります。上限は 36 です。

関連項目：『Oracle8i 管理者ガイド』

したがって、`jobs_per_batch` の上限は `MIN(#partitions, #job_queue_processes)` です。`jobs_per_batch` のデフォルト値は 1 で、これは、一度に 1 つのパーティションに対して索引が作成されることを示します。

- 障害の状態はトレース・ファイルにのみレポートされ（DBMS_JOB の制限事項）、ユーザーへの対話的なフィードバックはできません。このパッケージは、単に失敗時にメッセージを印刷して未完了のジョブを削除し、`snp*.trc` トレース・ファイルを調べるようにユーザーに要求するのみです。
- 前述のポイントの主な問題点は、大きな索引を作成するためには、ユーザーに Oracle のチューニング方法（特に様々な記憶域パラメータの設定方法）の知識が必要なことです。このパッケージは、そのようなチューニング・プロセスの支援を目的としていません。

サブプログラムの要約

DBMS_PCLXUTIL には、プロシージャ `BUILD_PART_INDEX` のみ含まれています。

BUILD_PART_INDEX プロシージャ

構文

```
DBMS_PCLXUTIL.build_part_index (
    jobs_per_batch  IN NUMBER    DEFAULT 1,
    procs_per_job   IN NUMBER    DEFAULT 1,
    tab_name        IN VARCHAR2  DEFAULT NULL,
    idx_name        IN VARCHAR2  DEFAULT NULL,
    force_opt       IN BOOLEAN   DEFAULT FALSE);
```

パラメータ

表 30-1 BUILD_PART_INDEX プロシージャのパラメータ

パラメータ	説明
jobs_per_batch	同時に作成するローカル索引数 (1 <= jobs_per_batch <= パーティション数)。
procs_per_job	ローカル索引作成ごとに利用するパラレル問合せスレーブ数 (1 <= procs_per_job <= max_slaves)。
tab_name	パーティション表の名前 (表が存在しない、またはパーティション化されていない場合は例外が発生します)。
idx_name	ローカル索引に指定する名前 (ローカル索引が表 tab_name に作成されていない場合は例外が発生します)。
force_opt	TRUE の場合は、すべてのパーティション索引の再作成を強制します。そうでない場合は、'UNUSABLE' にマークされたパーティションのみ再作成します。

例

PROJ001 と PROJ002 の 2 つのパーティション、およびローカル索引 IDX を持つ表 PROJECT が作成されるとします。

プロシージャ BUILD_PART_INDEX(2,4,'PROJECT','IDX',TRUE) へのコールによって、次の出力が作成されます。

```
SQLPLUS> EXECUTE dbms_pclxutil.build_part_index(2,4,'PROJECT','IDX',TRUE);
Statement processed.
INFO: Job #21 created for partition PROJ002 with 4 slaves
INFO: Job #22 created for partition PROJ001 with 4 slaves
```

DBMS_PIPE

DBMS_PIPE パッケージによって、同じインスタンスにある複数のセッションの通信を行います。Oracle パイプは、UNIX で使用するパイプと概念は似ていますが、オペレーティング・システムのパイプ・メカニズムを使用してインプリメントされません。

Oracle パイプを介して送信する情報は、システム・グローバル領域（SGA）にバッファリングされます。パイプ内のすべての情報は、インスタンスがシャットダウンすると失われます。

セキュリティ要件に応じて、パブリック・パイプまたはプライベート・パイプのいずれかを使用できます。

注意： パイプはトランザクションから独立しています。トランザクション制御に影響を与える可能性がある場合は、注意してパイプを使用してください。

パブリック・パイプ

パブリック・パイプは、暗黙的または明示的に作成できます。暗黙的なパブリック・パイプは、そのパイプの最初の参照時に自動的に作成され、データがなくなると消去されます。パイプ記述子は SGA に格納されるため、空のパイプがキャッシュから削除されるまで、スペースを使用するオーバーヘッドが若干あります。

明示的なパブリック・パイプを作成するためには、`private` フラグに `FALSE` を設定した `CREATE_PIPE` ファンクションをコールします。明示的に作成したパイプは、`REMOVE_PIPE` ファンクションをコールして割当て解除する必要があります。

パブリック・パイプのドメインは、明示的または暗黙的にパイプが作成されたスキーマです。

パイプへの書込みと読み込み

各パブリック・パイプは非同期に動作します。スキーマ・ユーザーが `DBMS_PIPE` パッケージに対する `EXECUTE` 許可を持ち、パブリック・パイプ名を知っている場合に限り、任意の数のスキーマ・ユーザーがパブリック・パイプへの書込みを行うことができます。ただし、バッファに入っている情報を 1 人のユーザーが読み込むと、その情報はバッファから消去されるため、同じパイプの他のユーザーは読み込むことができません。

送信側セッションでは、`PACK_MESSAGE` プロシージャに 1 つ以上コールを行ってメッセージを作成します。このプロシージャは、セッションのローカル・メッセージ・バッファにメッセージを追加します。このバッファ内の情報は、メッセージ送信に使用するパイプ名を指定した `SEND_MESSAGE` ファンクションをコールして送信されます。`SEND_MESSAGE` をコールすると、ローカル・バッファにスタックされたすべてのメッセージが送信されます。

メッセージを受信するプロセスは、メッセージを受信するパイプ名を指定した `RECEIVE_MESSAGE` ファンクションをコールします。次に、`UNPACK_MESSAGE` プロシージャをコールして、メッセージ内の各項目にアクセスします。

プライベート・パイプ

`CREATE_PIPE` ファンクションをコールして、プライベート・パイプを明示的に作成します。プライベート・パイプは、一度作成されると、`REMOVE_PIPE` ファンクションをコールして明示的に割当て解除するまで共有メモリに存続します。プライベート・パイプは、データベース・インスタンスがシャットダウンしたときにも割当て解除されます。

メモリに暗黙的なパイプが存在し、そのパイプと作成しようとするプライベート・パイプが同じ名前の場合、プライベート・パイプは作成できません。この場合は、`CREATE_PIPE` からエラーが戻されます。

プライベート・パイプへのアクセスは、次のように限定されます。

- パイプ作成者と同じユーザー ID で実行中のセッション
- パイプ作成者と同じユーザー ID 権限ドメインで実行中のストアド・サブプログラム
- SYSDBA または INTERNAL として接続したユーザー

これ以外のユーザーがパイプ上のメッセージの送受信やパイプの削除を試みると、即時にエラーが発生します。別のユーザーが同じ名前のパイプを作成しようとしても、エラーが発生します。

パブリック・パイプと同様に、SEND_MESSAGE をコールする前に PACK_MESSAGE へのコールを使用して、最初にメッセージを作成する必要があります。同様に、RECEIVE_MESSAGE をコールしてメッセージを取り出してから、UNPACK_MESSAGE をコールしてメッセージ内の項目にアクセスする必要があります。

パイプの使用方法

パイプ機能には、次のような潜在的な応用例があります。

- 外部サービス・インタフェース：RDBMS 外部にあるユーザー記述のサービスと通信することができます。この通信は、マルチスレッド方法で（効率的に）行うことができるため、サービスの複数インスタンスが同時に実行されます。さらに、サービスは非同期で使用できます。サービスのリクエストは、待機応答をブロックする必要がなく、（タイムアウトに関係なく）後でチェックできます。サービスは、Oracle がサポートする任意の 3GL 言語で記述できます。
- 独立したトランザクション：パイプは、操作を独立したトランザクション（トリガーで検出するセキュリティ違反のロギングなど）で実行できる個別のセッションと通信できます。
- アラート（トランザクション以外）：ポーリングするための待機プロセスを要求せずに、別のプロセスを転記できます。アプリケーションにアラートを通知する "AFTER 行" または "AFTER 文" トリガーがある場合、アプリケーションは、このアラートをデータが変更された可能性を示すためのアラートとして処理します。アプリケーションはそのデータを読み込み、現行の値を取得します。これは "AFTER" トリガーなので、アプリケーションは、"SELECT FOR UPDATE" を行って正しいデータを読み込んだことを確認します。
- デバッグ：トリガーとストアド・プロシージャは、デバッグ情報をパイプに送信できます。別のセッションは、パイプからの読み込みを続行し、その内容をスクリーンに表示したりファイルに書き込むことができます。
- コンセントレータ：これは、少数のネットワーク接続に対して多数のユーザーがいる場合にユーザーを多重化したり、複数のユーザー・トランザクションを 1 つの DBMS トランザクションに集中化する場合のパフォーマンス改善に役立ちます。

セキュリティ

セキュリティは、DBMS_PIPE パッケージの実行権限の付与を使用してアーカイブできます。これは、CREATE_PIPE ファンクションの private パラメータを使用してパイプを作成し、特定の機能、特定のユーザーやロールへのパイプ名を表示するのみのカバー・パッケージを記述することによって行われます。

定数

```
maxwait    constant integer := 86400000; /* 1000 days */
```

メッセージの送受信を行うための最大待機時間です。

エラー

DBMS_PIPE パッケージ・サブプログラムは、次のエラーを戻すことができます。

表 31-1 DBMS_PIPE エラー

エラー	説明
ORA-23321:	パイプ名には NULL を指定できません。このエラーは、CREATE_PIPE ファンクションまたはパイプ名をパラメータとして使用しているサブプログラムによって戻されます。
ORA-23322:	パイプへのアクセス権限が不十分です。このエラーは、パラメータ・リストにあるプライベート・パイプを参照するサブプログラムによって戻されます。

サブプログラムの要約

表 31-2 DBMS_PIPE パッケージのサブプログラム

サブプログラム	説明
31-5 ページの CREATE_PIPE ファンクション	パイプを明示的に作成します（プライベート・パイプに必要です）。
31-7 ページの PACK_MESSAGE プロシージャ	ローカル・バッファにメッセージを作成します。
31-8 ページの SEND_MESSAGE ファンクション	名前付きパイプにメッセージを送信します。名前付きパイプが存在しない場合は、パブリック・パイプが暗黙的に作成されます。
31-10 ページの RECEIVE_MESSAGE ファンクション	名前付きパイプからローカル・バッファにメッセージをコピーします。

表 31-2 DBMS_PIPE パッケージのサブプログラム

サブプログラム	説明
31-12 ページの NEXT_ITEM_TYPE ファンクション	バッファにある次の項目のデータ型を戻します。
31-13 ページの UNPACK_MESSAGE プロシージャ	バッファにある次の項目にアクセスします。
31-14 ページの REMOVE_PIPE ファンクション	名前付きパイプを削除します。
31-15 ページの PURGE プロシージャ	名前付きパイプの内容をパージします。
31-16 ページの RESET_BUFFER プロシージャ	ローカル・バッファの内容をパージします。
31-17 ページの UNIQUE_SESSION_NAME ファンクション	一意のセッション名を戻します。

CREATE_PIPE ファンクション

このファンクションは、パブリック・パイプまたはプライベート・パイプを明示的に作成します。private フラグが TRUE の場合、パイプ作成者がそのプライベート・パイプの所有者として割り当てられます。

明示的に作成されたパイプは、REMOVE_PIPE をコールするか、またはインスタンスをシャットダウンすることによってのみ削除できます。

構文

```
DBMS_PIPE.CREATE_PIPE (  
    pipename      IN VARCHAR2,  
    maxpipesize   IN INTEGER DEFAULT 8192,  
    private        IN BOOLEAN DEFAULT TRUE)  
RETURN INTEGER;
```

プラグマ

```
pragma restrict_references (create_pipe,WNDS,RNDS);
```

パラメータ

表 31-3 CREATE_PIPE ファンクションのパラメータ

パラメータ	説明
pipename	作成するパイプの名前。 SEND_MESSAGE および RECEIVE_MESSAGE をコールするときは、この名前を使用する必要があります。この名前は、インスタンス間で一意である必要があります。 注意:ORA\$ で始まるパイプ名を使用しないでください。これは、オラクル社が提供する製品用に予約されています。パイプ名は 128 バイト以下で指定し、大 / 小文字区別があります。現時点では、名前に NLS 文字を含めることはできません。
maxpipesize	パイプの最大サイズ（単位はバイト）。 パイプ上のすべてのメッセージの合計サイズは、この数を超えることはできません。この最大値を超えると、そのメッセージはブロックされます。デフォルトの maxpipesize は 8192 バイトです。 パイプの maxpipesize はパイプ特性の一部となり、パイプが存続する限り有効です。これより大きいパイプ・サイズで SEND_MESSAGE をコールすると、maxpipesize の値が大きくなります。これより小さいサイズでコールした場合は、より大きい既存の値を使用します。
private	デフォルトの TRUE を使用して、プライベート・パイプを作成します。 パブリック・パイプは、SEND_MESSAGE をコールして、暗黙的に作成できます。

戻り値

表 31-4 CREATE_PIPE ファンクションの戻り値

戻り値	説明
0	成功。 パイプが存在し、パイプを作成するユーザーにそのパイプの使用が認可されている場合、Oracle は 0（ゼロ）を戻して成功であることを示し、パイプ内のデータはそのまま残ります。 SYSDBA/SYSOPER として接続したユーザーがパイプを再作成した場合、Oracle はステータス 0 を戻しますが、そのパイプの所有者は変更されないままです。

表 31-4 CREATE_PIPE ファンクションの戻り値

戻り値	説明
ORA-23322	命名競合のために失敗。 同じ名前のパイプが存在し、別のユーザーがそのパイプを作成した場合、Oracle ではエラー ORA-23322 を通知して命名競合であることを示しています。

例外

表 31-5 CREATE_PIPE ファンクションの例外

例外	説明
パイプ名が NULL	アクセス権エラー。同名のパイプが存在するため使用できません。

PACK_MESSAGE プロシージャ

このプロシージャは、ユーザーのメッセージをローカル・メッセージ・バッファに作成します。

メッセージを送信するためには、最初に PACK_MESSAGE を 1 回以上コールします。次に、SEND_MESSAGE をコールして、ローカル・バッファにあるメッセージを名前付きパイプに送信します。

PACK_MESSAGE プロシージャは、VARCHAR2 型、NUMBER 型または DATE 型の項目を受け入れるためにオーバーロードされています。バッファ内の各項目には、データ・バイトの他に、項目の型を示すための 1 バイトと長さを格納するための 2 バイトが必要です。さらに、メッセージを終了するために 1 バイトが必要です。VARCHAR 以外のすべての型のオーバーヘッドは 4 バイトです。

Oracle8 では、キャラクタ・セット ID (2 バイト) とキャラクタ・セット・フォーム (1 バイト) が各データ項目に格納されています。したがって、Oracle8 を使用するときのオーバーヘッドは 7 バイトです。

SEND_MESSAGE をコールしてメッセージを送信するときは、メッセージを送信するパイプの名前を示す必要があります。パイプがすでに存在する場合は、そのパイプにアクセスするための十分な権限が必要です。パイプが存在しない場合は、自動的に作成されます。

構文

```
DBMS_PIPE.PACK_MESSAGE      (item IN VARCHAR2);
DBMS_PIPE.PACK_MESSAGE      (item IN NCHAR);
DBMS_PIPE.PACK_MESSAGE      (item IN NUMBER);
DBMS_PIPE.PACK_MESSAGE      (item IN DATE);
DBMS_PIPE.PACK_MESSAGE_RAW  (item IN RAW);
DBMS_PIPE.PACK_MESSAGE_ROWID (item IN ROWID);
```

注意： PACK_MESSAGE プロシージャは、VARCHAR2、NCHAR、NUMBER または DATE 型の項目を受け入れるためにオーバーロードされています。さらに、RAW と ROWID 項目をパックするための 2 つのプロシージャがあります。

プラグマ

```
pragma restrict_references(pack_message,WNDS,RNDS);
pragma restrict_references(pack_message_raw,WNDS,RNDS);
pragma restrict_references(pack_message_rowid,WNDS,RNDS);
```

パラメータ

表 31-6 PACK_MESSAGE プロシージャのパラメータ

パラメータ	説明
item	ローカル・メッセージ・バッファにパックする項目

例外

メッセージ・バッファ（現在は 4096 バイト）がオーバーフローした場合は、ORA-06558 が出されます。バッファ内の各項目には、実際のデータに加えて、型を示すために 1 バイト、長さを示すために 2 バイトが必要です。さらに、メッセージを終了するために 1 バイトが必要です。

SEND_MESSAGE ファンクション

このファンクションは、メッセージを名前付きパイプに送信します。

PACK_MESSAGE へのコールで入力されたメッセージは、ローカル・メッセージ・バッファに格納されます。パイプは CREATE_PIPE を使用して明示的に作成されます。そうでない場合は、暗黙的に作成されます。

構文

```
DBMS_PIPE.SEND_MESSAGE (  
    pipename      IN VARCHAR2,  
    timeout       IN INTEGER DEFAULT MAXWAIT,  
    maxpipesize   IN INTEGER DEFAULT 8192)  
RETURN INTEGER;
```

プラグマ

```
pragma restrict_references (send_message, WNDS, RNDS);
```

パラメータ

表 31-7 SEND_MESSAGE ファンクションのパラメータ

パラメータ	説明
pipename	<p>メッセージを設定するパイプの名前。</p> <p>明示的なパイプを使用している場合、この名前は、CREATE_PIPE をコールしたときに指定した名前です。</p> <p>注意: 'ORA\$' で始まるパイプ名を使用しないでください。この名前は、オラクル社が提供する製品用に予約されています。パイプ名は 128 バイト以下で指定し、大/小文字区別があります。現時点では、名前に NLS 文字を含めることはできません。</p>
timeout	<p>パイプにメッセージを設定する間の待機時間（単位は秒）。</p> <p>デフォルト値は定数 MAXWAIT で、86400000（1000 日）に定義されています。</p>
maxpipesize	<p>パイプの最大サイズ（単位はバイト）。</p> <p>パイプ上のすべてのメッセージの合計サイズは、この数を超えることはできません。この最大値を超えると、そのメッセージはブロックされます。デフォルトは 8192 バイトです。</p> <p>パイプの maxpipesize はパイプ特性の一部となり、パイプが存続する限り有効です。これより大きいパイプ・サイズで SEND_MESSAGE をコールすると、maxpipesize の値が大きくなります。これより小さいサイズでコールした場合は、より大きい既存の値を使用します。</p> <p>SEND_MESSAGE プロシージャの一部として maxpipesize を指定すると、別のコールでパイプをオープンする必要がなくなります。明示的にパイプを作成した場合は、オプションの maxpipesize パラメータを使用して、作成したパイプのサイズ仕様を上書きすることができます。</p>

戻り値

表 31-8 SEND_MESSAGE ファンクションの戻り値

戻り値	説明
0	成功。 パイプが存在し、パイプを作成するユーザーにそのパイプの使用が認可されている場合、Oracle は 0 を戻して成功であることを示し、パイプ内のデータはそのまま残ります。 SYSDBA または SYSOPER として接続したユーザーがパイプを再作成した場合、Oracle はステータス 0 を戻しますが、そのパイプの所有者は変更されないままです。
1	タイムアウト。 このプロシージャは、パイプでロックが取得できないか、またはパイプがいっぱいで使用できない理由でタイムアウトできます。暗黙的に作成されたパイプが空の場合、そのパイプは削除されます。
3	割込みが発生しました。 暗黙的に作成されたパイプが空の場合、そのパイプは削除されます。
ORA-23322	権限が不十分です。 同じ名前のパイプが存在し、別のユーザーがそのパイプを作成した場合、Oracle ではエラー ORA-23322 を通知して命名競合であることを示しています。

例外

表 31-9 SEND_MESSAGE ファンクションの例外

例外	説明
パイプ名が NULL	アクセス権エラー。パイプに書き込みを行うための権限が不十分です。パイプはプライベートで、別のユーザーが所有しています。

RECEIVE_MESSAGE ファンクション

このファンクションは、メッセージをローカル・メッセージ・バッファにコピーします。

パイプからメッセージを受信するには、最初に RECEIVE_MESSAGE をコールします。メッセージを受信すると、メッセージはパイプから削除されます。したがって、メッセージは 1 回しか受信できません。暗黙的に作成されたパイプは、最後のレコードがそのパイプから削除された後に削除されます。

RECEIVE_MESSAGE のコール時に指定したパイプがすでに存在しない場合、Oracle はパイプを暗黙的に作成してメッセージの受信を待ちます。メッセージが指定したタイムアウト時間内に着信しなかった場合、そのコールは戻され、パイプは削除されます。

メッセージの受信後、1 つ以上の UNPACK_MESSAGE をコールして、メッセージ内の個別の項目にアクセスする必要があります。UNPACK_MESSAGE は、DATE、NUMBER、VARCHAR2 型の項目をアンパックするためにオーバーロードされています。さらに、RAW と ROWID 項目をアンパックするために 2 つのプロシージャがあります。アンパックするデータの型が不明の場合は、NEXT_ITEM_TYPE をコールして、バッファ内にある次の項目の型を判別します。

構文

```
DBMS_PIPE.RECEIVE_MESSAGE (
    pipename      IN VARCHAR2,
    timeout       IN INTEGER      DEFAULT maxwait)
RETURN INTEGER;
```

プラグマ

```
pragma restrict_references (receive_message, WNDS, RNDS);
```

パラメータ

表 31-10 RECEIVE_MESSAGE ファンクションのパラメータ

パラメータ	説明
pipename	メッセージを受信するパイプ名。 ORA\$ で始まる名前は、オラクル社で使用するために予約されています。
timeout	メッセージを待つ時間（単位は秒）。 デフォルト値は定数 MAXWAIT で、86400000（1000 日）に定義されています。タイムアウトを 0 に指定すると、ブロックされずに読み込むことができます。

戻り値

表 31-11 RECEIVE_MESSAGE ファンクションの戻り値

戻り値	説明
0	成功。
1	タイムアウト。暗黙的に作成されたパイプが空の場合、そのパイプは削除されます。

表 31-11 RECEIVE_MESSAGE ファンクションの戻り値

戻り値	説明
2	パイプにあるレコードがバッファに対して大きすぎます（これは起こり得ない値です）。
3	割込みが発生しました。
ORA-23322	パイプから読み込むための十分な権限がユーザーにありません。

例外

表 31-12 RECEIVE_MESSAGE ファンクションの例外

例外	説明
パイプ名が NULL	アクセス権エラー。パイプからレコードを削除するための権限が不十分です。パイプは別のユーザーが所有しています。

NEXT_ITEM_TYPE ファンクション

このファンクションは、ローカル・メッセージ・バッファにある次の項目のデータ型を判別します。

RECEIVE_MESSAGE をコールして、ローカル・バッファにパイプ情報を設定した後、NEXT_ITEM_TYPE をコールします。

構文

```
DBMS_PIPE.NEXT_ITEM_TYPE
RETURN INTEGER;
```

プラグマ

```
pragma restrict_references(next_item_type,WNDS,RNDS);
```

戻り値

表 31-13 NEXT_ITEM_TYPE ファンクションの戻り値

戻り値	説明
0	次の項目がありません。
6	NUMBER
9	VARCHAR2
11	ROWID

表 31-13 NEXT_ITEM_TYPE ファンクションの戻り値

戻り値	説明
12	DATE
23	RAW

UNPACK_MESSAGE プロシージャ

このプロシージャは、バッファから項目を取り出します。

RECEIVE_MESSAGE をコールして、ローカル・バッファにパイプ情報を設定した後、UNPACK_MESSAGE をコールします。

構文

```
DBMS_PIPE.UNPACK_MESSAGE      (item OUT VARCHAR2);
DBMS_PIPE.UNPACK_MESSAGE      (item OUT NCHAR);
DBMS_PIPE.UNPACK_MESSAGE      (item OUT NUMBER);
DBMS_PIPE.UNPACK_MESSAGE      (item OUT DATE);
DBMS_PIPE.UNPACK_MESSAGE_RAW  (item OUT RAW);
DBMS_PIPE.UNPACK_MESSAGE_ROWID (item OUT ROWID);
```

注意： UNPACK_MESSAGE プロシージャは、VARCHAR2、NCHAR、NUMBER または DATE 型の項目を戻すためにオーバーロードされています。さらに、RAW と ROWID 項目をアンパックするための 2 つのプロシージャがあります。

プラグマ

```
pragma restrict_references (unpack_message,WNDS,RNDS);
pragma restrict_references (unpack_message_raw,WNDS,RNDS);
pragma restrict_references (unpack_message_rowid,WNDS,RNDS);
```

パラメータ

表 31-14 UNPACK_MESSAGE プロシージャのパラメータ

パラメータ	説明
item	アンパックされた次の項目をローカル・メッセージ・バッファから受け取るための引数。

例外

バッファに次の項目がない場合、または項目が要求された型でない場合は、ORA-06556 または 06559 が生成されます。

REMOVE_PIPE ファンクション

このファンクションは、明示的に作成されたパイプを削除します。

SEND_MESSAGE で暗黙的に作成されたパイプは、空になると自動的に削除されます。ただし、CREATE_PIPE で明示的に作成されたパイプは、REMOVE_PIPE をコールするか、またはインスタンスをシャットダウンした場合にのみ削除されます。パイプ内の未使用のレコードは、パイプが削除される前にすべて削除されます。

これは、暗黙的に作成されたパイプで PURGE をコールするのに似ています。

構文

```
DBMS_PIPE.REMOVE_PIPE (  
    pipename IN VARCHAR2)  
RETURN INTEGER;
```

プラグマ

```
pragma restrict_references(remove_pipe,WNDS,RNDS);
```

パラメータ

表 31-15 REMOVE_PIPE ファンクションのパラメータ

パラメータ	説明
pipename	削除するパイプ名

戻り値

表 31-16 REMOVE_PIPE ファンクションの戻り値

戻り値	説明
0	成功。 パイプが存在しない場合、またはパイプがすでに存在し、パイプを削除しようとするユーザーに削除が認可されている場合、Oracle は 0（ゼロ）を戻して成功であることを示し、パイプ内に残っているデータが削除されます。

表 31-16 REMOVE_PIPE ファンクションの戻り値

戻り値	説明
ORA-23322	権限が不十分です。 パイプは存在するが、ユーザーにそのパイプへのアクセス権がない場合、Oracle はエラー ORA-23322 を通知して権限が不十分であることを示します。

例外

表 31-17 REMOVE_PIPE ファンクションの例外

例外	説明
パイプ名が NULL	アクセス権エラー。パイプを削除する権限が不十分です。パイプは作成されていて、別のユーザーが所有しています。

PURGE プロシージャ

このプロシージャは、名前付きパイプの内容を空にします。

暗黙的に作成された空のパイプは、LRU アルゴリズムに従って、共有グローバル領域から削除されます。したがって、PURGE をコールすると、暗黙的に作成したパイプに関連するメモリーを空にすることができます。

PURGE プロシージャは RECEIVE_MESSAGE をコールするため、メッセージがパイプからページされるとき、ローカル・バッファはそのメッセージで上書きされる場合があります。また、不十分なアクセス権でパイプをページしようとする、ORA-23322 エラー（不十分な権限）を受け取ります。

構文

```
DBMS_PIPE.PURGE (  
    pipename IN VARCHAR2);
```

プラグマ

```
pragma restrict_references (purge, WNDS, RNDS);
```

パラメータ

表 31-18 PURGE プロシージャのパラメータ

パラメータ	説明
pipename	すべてのメッセージを削除するパイプの名前。 メッセージが廃棄されると、ローカル・バッファがそのメッセージで上書きされる場合があります。パイプ名は 128 バイト以下で指定し、大 / 小文字区別があります。

例外

パイプを別のユーザーが所有している場合は、アクセス権エラーが発生します。

RESET_BUFFER プロシージャ

このプロシージャは、PACK_MESSAGE と UNPACK_MESSAGE の位置設定標識を 0 にリセットします。

すべてのパイプが 1 つのバッファを共有しているため、新規パイプを使用する前にバッファをリセットすると効果的です。これにより、初めてメッセージをパイプに送信するとき、バッファ内に残っていた期限切れのメッセージを誤って送信することがなくなります。

構文

DBMS_PIPE.RESET_BUFFER;

パラメータ

なし

プラグマ

pragma restrict_references(reset_buffer,WNDS,RNDS);

UNIQUE_SESSION_NAME ファンクション

このファンクションは、現在データベースに接続しているすべてのセッション間での一意の名前を受け取ります。

同じセッションからこのファンクションを複数回コールしても、常に同じ値が戻されます。このファンクションは、SEND_MESSAGE と RECEIVE_MESSAGE のコールに PIPENAME パラメータを提供するのに役立ちます。

構文

```
DBMS_PIPE.UNIQUE_SESSION_NAME  
    RETURN VARCHAR2;
```

パラメータ

なし

プラグマ

```
pragma restrict_references(unique_session_name,WNDS,RNDS,WNPS);
```

戻り値

このファンクションは、一意の名前を戻します。戻される名前は、最大 30 バイトです。

例

例 1: デバッグ

この例は、デバッグ情報をパイプに設定するために、PL/SQL プログラムがコールできるプロシージャを示しています。

```
CREATE OR REPLACE PROCEDURE debug (msg VARCHAR2) AS  
    status NUMBER;  
BEGIN  
    DBMS_PIPE.PACK_MESSAGE(LENGTH(msg));  
    DBMS_PIPE.PACK_MESSAGE(msg);  
    status := DBMS_PIPE.SEND_MESSAGE('plsql_debug');  
    IF status != 0 THEN  
        raise_application_error(-20099, 'Debug error');  
    END IF;  
END debug;
```

次の例は、前述の PL/SQL 例にある PLSQL_DEBUG パイプからメッセージを受信して表示する Pro*C コードを示しています。Pro*C セッションが別のウィンドウで実行されている場合、このセッションは、別のセッションで実行中の PL/SQL プログラムからデバッグ・プロシージャに送信されるメッセージの表示に使用できます。

```
#include <stdio.h>
#include <string.h>

EXEC SQL BEGIN DECLARE SECTION;
    VARCHAR username[20];
    int      status;
    int      msg_length;
    char      retval[2000];
EXEC SQL END DECLARE SECTION;

EXEC SQL INCLUDE SQLCA;

void sql_error();

main()
{
    -- Prepare username:
    strcpy(username.arr, "SCOTT/TIGER");
    username.len = strlen(username.arr);

    EXEC SQL WHENEVER SQLERROR DO sql_error();
    EXEC SQL CONNECT :username;

    printf("connected\n");

    -- Start an endless loop to look for and print messages on the pipe:
    FOR (;;)
    {
        EXEC SQL EXECUTE
            DECLARE
                len INTEGER;
                typ INTEGER;
                sta INTEGER;
                chr VARCHAR2(2000);
            BEGIN
                chr := '';
                sta := dbms_pipe.receive_message('plsql_debug');
                IF sta = 0 THEN
                    DBMS_PIPE.UNPACK_MESSAGE(len);
                    DBMS_PIPE.UNPACK_MESSAGE(chr);
                END IF;
            
```

```

        :status := sta;
        :retval := chr;
        IF len IS NOT NULL THEN
            :msg_length := len;
        ELSE
            :msg_length := 2000;
        END IF;
    END;
END-EXEC;
IF (status == 0)
    printf("\n%.*s\n", msg_length, retval);
ELSE
    printf("abnormal status, value is %d\n", status);
}
}

void sql_error()
{
    char msg[1024];
    int rlen, len;
    len = sizeof(msg);
    sqlglm(msg, &len, &rlen);
    printf("ORACLE ERROR\n");
    printf("%.*s\n", rlen, msg);
    exit(1);
}

```

例 2: システム・コマンドの実行

この例は、PL/SQL と Pro*C コードによって、PL/SQL ストアド・プロシージャ（または無名ブロック）が PL/SQL プロシージャをコールし、コマンドをリスニングしている Pro*C プログラムにパイプを介してそのコマンドを送信する処理を示します。

Pro*C プログラムはスリープして、名前付きパイプにメッセージが着信するのを待ちます。メッセージが着信すると、C プログラムはそれを処理し、system() コールから UNIX コマンドを実行したり、埋込み SQL を使用して SQL コマンドを実行して、必要なアクションを実行します。

DAEMON.SQL は、PL/SQL パッケージ用のソース・コードです。このパッケージには、DBMS_PIPE パッケージを使用して Pro*C デーモンに対してメッセージを送受信するプロシージャが含まれています。完全なハンドシェークが使用されていることに注意してください。このデーモンは、常にメッセージをパッケージに返信します（STOP コマンドの場合は除きます）。これによって、PL/SQL プロシージャは Pro*C デーモンが動作していることを確認できるため、デーモンのこの機能は重要です。

SQL*Plus または Enterprise Manager を使用して、無名 PL/SQL ブロックから DAEMON パッケージ・プロシージャをコールできます。次に例を示します。

```
SQLPLUS> variable rv number
SQLPLUS> execute :rv := DAEMON.EXECUTE_SYSTEM('ls -la');
```

これにより、UNIX システムでは、Pro*C デーモンによってコマンド `system("ls -la")` が実行されます。

最初にデーモンを実行する必要があることに留意してください。デーモンは、バックグラウンドで実行したり、デーモンをコールした SQL*Plus または Enterprise Manager セッションの近くにある別のウィンドウで実行できます。

また、DAEMON.SQL は、DBMS_OUTPUT パッケージを使用して結果を表示します。この例を動作させるには、このパッケージに対する EXECUTE 権限が必要です。

DAEMON.SQL 例 次の例は、PL/SQL DAEMON パッケージのコード例です。

```
CREATE OR REPLACE PACKAGE daemon AS
    FUNCTION execute_sql(command VARCHAR2,
                        timeout NUMBER DEFAULT 10)
        RETURN NUMBER;

    FUNCTION execute_system(command VARCHAR2,
                        timeout NUMBER DEFAULT 10)
        RETURN NUMBER;

    PROCEDURE stop(timeout NUMBER DEFAULT 10);
END daemon;
/
CREATE OR REPLACE PACKAGE BODY daemon AS

    FUNCTION execute_system(command VARCHAR2,
                        timeout NUMBER DEFAULT 10)
        RETURN NUMBER IS

        status      NUMBER;
        result      VARCHAR2(20);
        command_code NUMBER;
        pipe_name   VARCHAR2(30);
    BEGIN
        pipe_name := DBMS_PIPE.UNIQUE_SESSION_NAME;

        DBMS_PIPE.PACK_MESSAGE('SYSTEM');
        DBMS_PIPE.PACK_MESSAGE(pipe_name);
        DBMS_PIPE.PACK_MESSAGE(command);
        status := DBMS_PIPE.SEND_MESSAGE('daemon', timeout);
        IF status <> 0 THEN
```

```
        RAISE_APPLICATION_ERROR(-20010,
        'Execute_system: Error while sending. Status = ' ||
        status);
    END IF;

    status := DBMS_PIPE.RECEIVE_MESSAGE(pipe_name, timeout);
    IF status <> 0 THEN
        RAISE_APPLICATION_ERROR(-20011,
        'Execute_system: Error while receiving.
        Status = ' || status);
    END IF;

    DBMS_PIPE.UNPACK_MESSAGE(result);
    IF result <> 'done' THEN
        RAISE_APPLICATION_ERROR(-20012,
        'Execute_system: Done not received.');
```

```
    END IF;

    DBMS_PIPE.UNPACK_MESSAGE(command_code);
    DBMS_OUTPUT.PUT_LINE('System command executed. result = ' ||
        command_code);

    RETURN command_code;
END execute_system;

FUNCTION execute_sql(command VARCHAR2,
                    timeout NUMBER DEFAULT 10)
RETURN NUMBER IS

    status      NUMBER;
    result      VARCHAR2(20);
    command_code NUMBER;
    pipe_name   VARCHAR2(30);

BEGIN
    pipe_name := DBMS_PIPE.UNIQUE_SESSION_NAME;

    DBMS_PIPE.PACK_MESSAGE('SQL');
    DBMS_PIPE.PACK_MESSAGE(pipe_name);
    DBMS_PIPE.PACK_MESSAGE(command);
    status := DBMS_PIPE.SEND_MESSAGE('daemon', timeout);
    IF status <> 0 THEN
        RAISE_APPLICATION_ERROR(-20020,
        'Execute_sql: Error while sending. Status = ' || status);
    END IF;

    status := DBMS_PIPE.RECEIVE_MESSAGE(pipe_name, timeout);
```

```
IF status <> 0 THEN
    RAISE_APPLICATION_ERROR(-20021,
        'execute_sql: Error while receiving.
        Status = ' || status);
END IF;

DBMS_PIPE.UNPACK_MESSAGE(result);
IF result <> 'done' THEN
    RAISE_APPLICATION_ERROR(-20022,
        'execute_sql: done not received. ');
END IF;

DBMS_PIPE.UNPACK_MESSAGE(command_code);
DBMS_OUTPUT.PUT_LINE
    ('SQL command executed.  sqlcode = ' || command_code);
RETURN command_code;
END execute_sql;

PROCEDURE stop(timeout NUMBER DEFAULT 10) IS
    status NUMBER;
BEGIN
    DBMS_PIPE.PACK_MESSAGE('STOP');
    status := DBMS_PIPE.SEND_MESSAGE('daemon', timeout);
    IF status <> 0 THEN
        RAISE_APPLICATION_ERROR(-20030,
            'stop: error while sending.  status = ' || status);
    END IF;
END stop;
END daemon;
```

daemon.pc 例 次の例は、Pro*C デーモンのコード例です。バージョン 1.5.x 以降の Pro*C プリコンパイラを使用して、このコードをプリコンパイルする必要があります。この例には埋込み PL/SQL コードが含まれているため、USERID と SQLCHECK オプションも指定する必要があります。

次に例を示します。

```
proc iname=daemon userid=scott/tiger sqlcheck=semantics
```


次に、通常の方法で、C コンパイルしてリンクします。

```
#include <stdio.h>
#include <string.h>

EXEC SQL INCLUDE SQLCA;

EXEC SQL BEGIN DECLARE SECTION;
    char *uid = "scott/tiger";
    int status;
    VARCHAR command[20];
    VARCHAR value[2000];
    VARCHAR return_name[30];
EXEC SQL END DECLARE SECTION;

void
connect_error()
{
    char msg_buffer[512];
    int msg_length;
    int buffer_size = 512;

    EXEC SQL WHENEVER SQLERROR CONTINUE;
    sqlgln(msg_buffer, &buffer_size, &msg_length);
    printf("Daemon error while connecting:\n");
    printf("%.s\n", msg_length, msg_buffer);
    printf("Daemon quitting.\n");
    exit(1);
}

void
sql_error()
{
    char msg_buffer[512];
    int msg_length;
    int buffer_size = 512;

    EXEC SQL WHENEVER SQLERROR CONTINUE;
    sqlgln(msg_buffer, &buffer_size, &msg_length);
    printf("Daemon error while executing:\n");
    printf("%.s\n", msg_length, msg_buffer);
    printf("Daemon continuing.\n");
}

main()
{
```

```
EXEC SQL WHENEVER SQLERROR DO connect_error();
EXEC SQL CONNECT :uid;
printf("Daemon connected.\n");

EXEC SQL WHENEVER SQLERROR DO sql_error();
printf("Daemon waiting...\n");
while (1) {
    EXEC SQL EXECUTE
        BEGIN
            :status := DBMS_PIPE.RECEIVE_MESSAGE('daemon');
            IF :status = 0 THEN
                DBMS_PIPE.UNPACK_MESSAGE(:command);
            END IF;
        END;
    END-EXEC;
    IF (status == 0)
    {
        command.arr[command.len] = '\0';
        IF (!strcmp((char *) command.arr, "STOP"))
        {
            printf("Daemon exiting.\n");
            break;
        }

        ELSE IF (!strcmp((char *) command.arr, "SYSTEM"))
        {
            EXEC SQL EXECUTE
                BEGIN
                    DBMS_PIPE.UNPACK_MESSAGE(:return_name);
                    DBMS_PIPE.UNPACK_MESSAGE(:value);
                END;
            END-EXEC;
            value.arr[value.len] = '\0';
            printf("Will execute system command '%s'\n", value.arr);

            status = system(value.arr);
            EXEC SQL EXECUTE
                BEGIN
                    DBMS_PIPE.PACK_MESSAGE('done');
                    DBMS_PIPE.PACK_MESSAGE(:status);
                    :status := DBMS_PIPE.SEND_MESSAGE(:return_name);
                END;
            END-EXEC;

            IF (status)
```

```

    {
        printf
            ("Daemon error while responding to system command.");
        printf("  status: %d\n", status);
    }
}
ELSE IF (!strcmp((char *) command.arr, "SQL")) {
    EXEC SQL EXECUTE
    BEGIN
        DBMS_PIPE.UNPACK_MESSAGE(:return_name);
        DBMS_PIPE.UNPACK_MESSAGE(:value);
    END;
    END-EXEC;
    value.arr[value.len] = '\0';
    printf("Will execute sql command '%s'\n", value.arr);

    EXEC SQL WHENEVER SQLERROR CONTINUE;
    EXEC SQL EXECUTE IMMEDIATE :value;
    status = sqlca.sqlcode;

    EXEC SQL WHENEVER SQLERROR DO sql_error();
    EXEC SQL EXECUTE
    BEGIN
        DBMS_PIPE.PACK_MESSAGE('done');
        DBMS_PIPE.PACK_MESSAGE(:status);
        :status := DBMS_PIPE.SEND_MESSAGE(:return_name);
    END;
    END-EXEC;

    IF (status)
    {
        printf("Daemon error while responding to sql command.");
        printf("  status: %d\n", status);
    }
}
ELSE
{
    printf
        ("Daemon error: invalid command '%s' received.\n",
         command.arr);
}
}
ELSE
{
    printf("Daemon error while waiting for signal.");
}

```

```

        printf("  status = %d\n", status);
    }
}
EXEC SQL COMMIT WORK RELEASE;
exit(0);

```

例 3: 外部サービス・インタフェース

ユーザー作成の 3GL コードを、OCI またはプリコンパイラ・プログラムに設定します。プログラムは、データベースに接続して PL/SQL コードを実行し、パイプから要求を読み込み、結果を計算します。次に、PL/SQL コードを実行して、パイプ上の結果をリクエストに返信します。

次に、株式サービス要求の例を示します。すべてのサービス要求についてパイプに渡す引数の順序は、次のようにお勧めします。

protocol_version	VARCHAR2	- '1', 10 bytes or less
returnpipe	VARCHAR2	- 30 bytes or less
service	VARCHAR2	- 30 bytes or less
arg1	VARCHAR2/NUMBER/DATE	
...		
argn	VARCHAR2/NUMBER/DATE	

結果を戻すための書式は、次のようにお勧めします。

success	VARCHAR2	- 'SUCCESS' if OK, otherwise error message
arg1	VARCHAR2/NUMBER/DATE	
...		
argn	VARCHAR2/NUMBER/DATE	

OCI または PRO*（疑似コードで）を使用して、株価要求サーバーを次のように設定します。

```

<loop forever>
  BEGIN dbms_stock_server.get_request(:stocksymbol); END;
  <figure out price based on stocksymbol (probably from some radio
    signal), set error if can't find such a stock>
  BEGIN dbms_stock_server.return_price(:error, :price); END;

```

クライアントは次のように設定します。

```

BEGIN :price := stock_request('YOURCOMPANY'); end;

```

前述の株価要求サーバーでコールしたストアド・プロシージャ dbms_stock_server は、次のように設定します。

```
CREATE OR REPLACE PACKAGE dbms_stock_server IS
  PROCEDURE get_request(symbol OUT VARCHAR2);
  PROCEDURE return_price(errormsg IN VARCHAR2, price IN VARCHAR2);
END;

CREATE OR REPLACE PACKAGE BODY dbms_stock_server IS
  returnpipe    VARCHAR2(30);

  PROCEDURE returnerror(reason VARCHAR2) IS
    s INTEGER;
  BEGIN
    dbms_pipe.pack_message(reason);
    s := dbms_pipe.send_message(returnpipe);
    IF s <> 0 THEN
      raise_application_error(-20000, 'Error: ' || to_char(s) ||
        ' sending on pipe');
    END IF;
  END;

  PROCEDURE get_request(symbol OUT VARCHAR2) IS
    protocol_version VARCHAR2(10);
    s                 INTEGER;
    service            VARCHAR2(30);
  BEGIN
    s := dbms_pipe.receive_message('stock_service');
    IF s <> 0 THEN
      raise_application_error(-20000, 'Error: ' || to_char(s) ||
        ' reading pipe');
    END IF;
    dbms_pipe.unpack_message(protocol_version);
    IF protocol_version <> '1' THEN
      raise_application_error(-20000, 'Bad protocol: ' ||
        protocol_version);
    END IF;
    dbms_pipe.unpack_message(returnpipe);
    dbms_pipe.unpack_message(service);
    IF service != 'getprice' THEN
      returnerror('Service ' || service || ' not supported');
    END IF;
    dbms_pipe.unpack_message(symbol);
  END;

  PROCEDURE return_price(errormsg in VARCHAR2, price in VARCHAR2) IS
    s INTEGER;
```

```
BEGIN
  IF errormsg is NULL THEN
    dbms_pipe.pack_message('SUCCESS');
    dbms_pipe.pack_message(price);
  ELSE
    dbms_pipe.pack_message(errormsg);
  END IF;
  s := dbms_pipe.send_message(returnpipe);
  IF s <> 0 THEN
    raise_application_error(-20000, 'Error: ' || to_char(s) ||
      ' sending on pipe');
  END IF;
END;
```

クライアントがコールするプロシージャは、次のように設定します。

```
CREATE OR REPLACE FUNCTION stock_request (symbol VARCHAR2)
  RETURN VARCHAR2 IS
  s      INTEGER;
  price  VARCHAR2(20);
  errormsg VARCHAR2(512);
BEGIN
  dbms_pipe.pack_message('1'); -- protocol version
  dbms_pipe.pack_message(dbms_pipe.unique_session_name); -- return pipe
  dbms_pipe.pack_message('getprice');
  dbms_pipe.pack_message(symbol);
  s := dbms_pipe.send_message('stock_service');
  IF s <> 0 THEN
    raise_application_error(-20000, 'Error: ' || to_char(s) ||
      ' sending on pipe');
  END IF;
  s := dbms_pipe.receive_message(dbms_pipe.unique_session_name);
  IF s <> 0 THEN
    raise_application_error(-20000, 'Error: ' || to_char(s) ||
      ' receiving on pipe');
  END IF;
  dbms_pipe.unpack_message(errormsg);
  IF errormsg <> 'SUCCESS' THEN
    raise_application_error(-20000, errormsg);
  END IF;
  dbms_pipe.unpack_message(price);
  RETURN price;
END;
```

一般的に、株式サービス・アプリケーション・サーバーに対してのみ `dbms_stock_service` の実行権限を付与し、このサービスを利用できるユーザーに対してのみ `stock_request` の実行権限を付与します。

関連項目： [第 2 章「DBMS_ALERT」](#)

DBMS_PROFILER

Oracle8i は、既存の PL/SQL アプリケーションをプロファイルし、パフォーマンスのボトルネックを識別するためのプロファイラ API を提供します。収集されたプロファイラ（パフォーマンス）データは、パフォーマンスを向上させるため、または PL/SQL アプリケーションのコード・カバレッジを決定するために使用できます。アプリケーション開発者は、コード・カバレッジ・データを使用して、増分テストに集中できます。

プロファイラ API は、PL/SQL パッケージの DBMS_PROFILER としてインプリメントされ、PL/SQL プロファイラ・データを収集し、継続的に格納するためのサービスを提供します。

DBMS_PROFILER の使用方法

アプリケーションのパフォーマンス向上は、繰り返し行うプロセスです。この反復プロセスには、次のステップが伴います。

1. プロファイラ・データの収集を可能にし、1 つ以上のベンチマーク・テストを使用してアプリケーションを実行します。
2. プロファイラ・データを分析し、パフォーマンス上の問題を識別します。
3. 問題を解決します。

PL/SQL プロファイラは、"実行" という概念を使用してこのプロセスをサポートします。実行には、プロファイラ・データの収集を可能にし、アプリケーションをベンチマーク・テストを介して実行することが含まれます。実行の開始と終了は、START_PROFILER と STOP_PROFILER ファンクションをコールして制御できます。

一般的な実行には、次の処理が含まれます。

- 実行でプロファイラ・データ収集を開始します。
- プロファイラ・データおよびコード・カバレッジ・データが必要な PL/SQL コードを実行します。
- プロファイラ・データ収集を停止します。その実行に対して収集したデータは、データベース表に書き込まれます。

注意： 収集したプロファイラ・データは、ユーザーの切断時に自動的に格納されません。セッションの終了時にデータを格納するためには、FLUSH_DATA または STOP_PROFILER ファンクションの明示的なコールを発行する必要があります。データ収集を停止すると、収集されたデータが格納されます。

アプリケーションの実行時、プロファイラ・データは、実行期間中存続するメモリー・データ構造に収集されます。FLUSH_DATA ファンクションを実行の途中でコールして、増分データを取得し、割り当てられているファイラ・データ構造用のメモリーを解放できます。

収集されたデータをフラッシュすると、その内容がデータベース表に格納されます。この表は、プロファイラ・ユーザーのスキーマにすでに存在する必要があります。PROFTAB.SQL スクリプトは、プロファイラ・データを継続的に格納するための表や他のデータ構造を作成します。

PROFTAB.SQL を実行すると、現行の表が削除されることに注意してください。PROFTAB.SQL スクリプトは、RDBMS/ADMIN ディレクトリにあります。PL/SQL ユニットの最初の実行など、一部の PL/SQL 操作には、実行中の PL/SQL ユニットにバイト・コードをロードするカタログ表への I/O が含まれる場合があります。また、パッケージ・プロシージャまたはファンクションが最初にコールされたとき、パッケージ初期化コードの実行に時間がかかる場合があります。

この時間的なオーバーヘッドを避けるためには、プロファイラ・データの収集前に、データベースのウォーム・アップを行います。ウォーム・アップを行うには、プロファイラ・データを収集せずにアプリケーションを 1 回実行します。

システム全体のプロファイル

システムの全ユーザーをプロファイルできます。たとえば、使用中か否かに関係なく、あるパッケージの全ユーザーをプロファイルできます。このような場合、SYSADMIN は、変更した PROFLOAD.SQL スクリプトで次の内容を実行します。

- プロファイラ表と順序を作成します。
- これらの表と順序の SELECT/INSERT/UPDATE 権限をすべてのユーザーに付与します。
- 表と順序のパブリック・シノニムを定義します。

注意： 表の実際のフィールドは変更しないでください。

関連項目： 32-9 ページの「[FLUSH_DATA ファンクション](#)」

要件

DBMS_PROFILER は、SYS としてインストールする必要があります。

PROFLOAD.SQL スクリプトを使用して、PL/SQL プロファイラ・パッケージをロードします。

収集されたデータ

プローブ・プロファイラ API を使用すると、セッションで実行されるすべての名前付きライブラリ・ユニットについて、プロファイル情報を生成できます。プロファイラは、PL/SQL 仮想マシン・レベルで情報を収集します。その情報には、各行の合計実行回数、その行の実行に要した合計時間、およびその行の特定の実行に要した最小時間と最大時間が含まれています。

注意： データが収集された PL/SQL ユニットに関するコード・カバレッジ値を推論することは可能です。

プロファイル情報は、データベース表に格納されています。このため、データに対して非定型な問合せが可能です。ユーザーは、カスタマイズ可能なレポート（サマリー・レポート、最新行、コード・カバレッジ・データなど）を作成できます。データの分析も可能です。

PROFTAB.SQL

PROFTAB.SQL スクリプトは、表 32-1、表 32-2 および表 32-3 にリストした列、データ型および定義を持つ表を作成します。

表 PLSQL_PROFILER_RUNS

表 32-1 表 PLSQL_PROFILER_RUNS の列

列	データ型	定義
runid	NUMBER	主キー。plsql_profiler_runnumber で作成される一意の実行識別子。
related_run	NUMBER	(クライアントとサーバーの相関関係に) 関連する実行の実行 ID。
run_owner	VARCHAR2 (32)	実行を開始したユーザー。
run_date	DATE	実行の開始時間。
run_comment	VARCHAR (2047)	この実行に関してユーザーが指定したコメント。
run_total_time	NUMBER	この実行の経過時間 (ナノ秒)。
run_system_info	VARCHAR2 (2047)	現在は使用されていません。
run_comment1	VARCHAR2 (2047)	追加のコメント。
spare1	VARCHAR2 (256)	未使用。

表 PLSQL_PROFILER_UNITS

表 32-2 表 PLSQL_PROFILER_UNITS の列

列	データ型	定義
runid	NUMBER	主キー。plsql_profiler_runs を参照します。
unit_number	NUMBER	主キー。内部的に生成されたライブラリ・ユニット番号。
unit_type	VARCHAR2 (32)	ライブラリ・ユニットのタイプ。
unit_owner	VARCHAR2 (32)	ライブラリ・ユニットの所有者名。
unit_name	VARCHAR2 (32)	ライブラリ・ユニットにおけるライブラリ・ユニット名のタイムスタンプ。
unit_timestamp	DATE	ユニットに複数実行の間に発生した変更を検出するために、将来使用される予定です。

表 32-2 表 PLSQL_PROFILER_UNITS の列

列	データ型	定義
total_time	NUMBER	このユニットで経過した合計時間（ナノ秒）。プロファイラはこのフィールドを設定しませんが、分析ツールで利用するために用意されています。
spare1	NUMBER	未使用。
spare2	NUMBER	未使用。

表 PLSQL_PROFILER_DATA

表 32-3 表 PLSQL_PROFILER_DATA の列

列	データ型	定義
runid	NUMBER	主キー。一意の（生成された）実行識別子。
unit_number	NUMBER	主キー。内部的に生成されたライブラリ・ユニット番号。
line#	NUMBER	主キー。ユニット内の NULL 以外の行番号。
total_occur	NUMBER	行が実行された回数。
total_time	NUMBER	行の実行に要した合計時間（ナノ秒）。
min_time	NUMBER	この行の最小実行時間（ナノ秒）。
max_time	NUMBER	この行の最大実行時間（ナノ秒）。
spare1	NUMBER	未使用。
spare2	NUMBER	未使用。
spare3	NUMBER	未使用。
spare4	NUMBER	未使用。

Oracle8 には、PL/SQL デモ用スクリプトに、文脈依存のレポート・ライター（profrep.sql）のサンプルが準備されています。

セキュリティの考慮事項

プロファイラは、ユーザーに CREATE 権限があるユニットのデータのみ収集します。EXECUTE ONLY アクセス権限が付与されているユニットを、パッケージを使用してプロファイルすることはできません。一般的に、ユニットをデバッグできるユーザーは、そのユニットをプロファイルできます。ただし、ユニットは DEBUG でコンパイルされているかどうかに関係なく、プロファイルできます。プロファイル対象のモジュールは、DEBUG でコンパイルすることをお勧めします。これによって、データベース内のユニットに関する追加情報が提供されます。

サブプログラムの要約

このセクションでは、DBMS_PROFILER のサブプログラムを要約します。

例外生成の 2 つの方法

このパッケージの各ルーチンには、エラーのレポート方法が 2 通りあります。

- 成功または失敗をステータス値として戻すファンクション。例外を呼び出すことはありません。
- 成功した場合は正常に戻り、失敗した場合は例外を呼び出すプロシージャ。

いずれの場合も、ファンクションとプロシージャのパラメータは同じです。エラーのレポート方法のみ異なります。エラーがある場合、ファンクションが戻すエラー・コードと、プロシージャが呼び出す例外は対応しています。

冗長性を避けるために、次のセクションでは、ファンクションのフォームに関する詳細のみ提供します。

例外

表 32-4 は、DBMS_PROFILER の例外を示しています。

表 32-4 DBMS_PROFILER の例外

例外	説明
version_mismatch	error_version に相当します。
profiler_error	"error_param" または "error_io" のいずれかに相当します。

エラー・コード

ファンクションからの戻り値が 0（ゼロ）の場合は、正常終了を示します。0（ゼロ）以外の戻り値は、エラー状態を示します。潜在的なエラーは、次のとおりです。

- サブプログラムが不正なパラメータとともにコールされました。

```
error_param constant binary_integer := 1;
```
- データ・フラッシュ操作に失敗しました。プロファイラ表が作成済でアクセス可能か、および十分な領域があるかどうかをチェックしてください。

```
error_io      constant binary_integer := 2;
```
- パッケージとデータベースの実装に不一致があります。不適切なバージョンの DBMS_PROFILER パッケージがインストールされ、このバージョンのプロファイラ・パッケージがそのデータベース・バージョンで動作できない場合に、このエラーが戻されます。リカバリするための唯一の方法は、適切なバージョンのパッケージをインストールすることです。

```
error_version constant binary_integer := -1;
```

表 32-5 DBMS_PROFILER パッケージのサブプログラム

サブプログラム	説明
32-8 ページの START_PROFILER ファンクション	ユーザーのセッションでプロファイラ・データ収集を開始します。
32-9 ページの STOP_PROFILER ファンクション	ユーザーのセッションでプロファイラ・データ収集を停止します。
32-9 ページの FLUSH_DATA ファンクション	ユーザーのセッションでプロファイラ・データ収集をフラッシュします。
32-9 ページの PAUSE_PROFILER ファンクション	プロファイラ・データ収集を一時停止します。
32-9 ページの RESUME_PROFILER ファンクション	プロファイラ・データ収集を再開します。
32-10 ページの GET_VERSION プロシージャ	この API のバージョンを取得します。
32-10 ページの INTERNAL_VERSION_CHECK ファンクション	このバージョンの DBMS_PROFILER パッケージが、データベース内の実装で動作可能であることを検証します。

START_PROFILER ファンクション

このファンクションは、ユーザーのセッションでプロファイラ・データ収集を開始します。

構文

START_PROFILER ファンクションのフォームは2種類、オーバーロードされています。1つはコール結果のみではなく、開始した実行の実行番号も戻します。もう1つは実行番号を戻しません。最初のフォームは、プロファイラを制御する GUI ベースのツールで使用することを目的としています。

1 番目のフォームは次のとおりです。

```
DBMS_PROFILER.START_PROFILER(run_comment IN VARCHAR2 := sysdate,
run_comment1 IN VARCHAR2 := '',
run_number OUT BINARY_INTEGER)
RETURN BINARY_INTEGER;
```

2 番目のフォームは次のとおりです。

```
DBMS_PROFILER.START_PROFILER(run_comment IN VARCHAR2 := sysdate,
run_comment1 IN VARCHAR2 := '')
RETURN BINARY_INTEGER;
```

パラメータ

表 32-6 START_PROFILER ファンクションのパラメータ

パラメータ	説明
run_comment	各プロファイラの実行にコメントを関連付けることができます。たとえば、コメントによって、データ収集で使用したベンチマーク・テストの名前とバージョンを提供できます。
run_number	実行の番号を格納します。ユーザーは実行のデータを格納しておき、後で再コールできます。
run_comment1	実行に関するわかりやすいコメントを記述できます。

STOP_PROFILER ファンクション

このファンクションは、ユーザーのセッションでプロファイラ・データ収集を停止します。

このファンクションには、セッションでそれまでに収集したデータをフラッシュする副次効果があり、これが実行の終了を示します。

構文

```
DBMS_PROFILER.STOP_PROFILER  
RETURN BINARY_INTEGER;
```

パラメータ

なし

FLUSH_DATA ファンクション

このファンクションは、ユーザーのセッションで収集したプロファイラ・データをフラッシュします。データは、以前から存在しているデータベース表にフラッシュされます。

注意： PROFTAB.SQL スクリプトを使用して、プロファイラ・データを継続的に格納するための表や他のデータ構造を作成します。

構文

```
DBMS_PROFILER.FLUSH_DATA  
RETURN BINARY_INTEGER;
```

パラメータ

なし

PAUSE_PROFILER ファンクション

このファンクションは、プロファイラ・データ収集を一時停止します。

RESUME_PROFILER ファンクション

このファンクションは、プロファイラ・データ収集を再開します。

GET_VERSION プロシージャ

このプロシージャは、この API のバージョンを取得します。

構文

```
DBMS_PROFILER.GET_VERSION (  
    major OUT BINARY_INTEGER,  
    minor OUT BINARY_INTEGER);
```

パラメータ

表 32-7 GET_VERSION プロシージャのパラメータ

パラメータ	説明
major	DBMS_PROFILER のバージョン番号
minor	DBMS_PROFILER のリリース番号

INTERNAL_VERSION_CHECK ファンクション

このファンクションは、このバージョンの DBMS_PROFILER パッケージが、データベース内の実装で動作可能であること検証します。

構文

```
DBMS_PROFILER.INTERNAL_VERSION_CHECK  
    RETURN BINARY_INTEGER;
```

パラメータ

なし

DBMS_RANDOM

DBMS_RANDOM パッケージは、組込み式の乱数ジェネレータを提供します。このパッケージは、Oracle の内部乱数ジェネレータをコールするため、PL/SQL で記述したジェネレータより高速です。

要件

DBMS_RANDOM は、乱数ジェネレータをコールする前に初期化する必要があります。このジェネレータは 8 桁の整数を生成します。初期化サブプログラムがコールされないと、パッケージでは例外が発生します。

サブプログラムの要約

表 33-1 DBMS_RANDOM パッケージのサブプログラム

サブプログラム	説明
33-2 ページの INITIALIZE プロシージャ	シード値を使用して、パッケージを初期化します。
33-3 ページの SEED プロシージャ	シードをリセットします。
33-3 ページの RANDOM ファンクション	乱数を取得します。
33-3 ページの TERMINATE プロシージャ	パッケージをクローズします。

INITIALIZE プロシージャ

このパッケージを使用するためには、最初にシードを使用して初期化サブプログラムをコールします。

構文

```
DBMS_RANDOM.INITIALIZE (  
    seed IN BINARY_INTEGER);
```

注意： 5 桁を超える十分な大きさのシードを使用してください。1 桁の値では、乱数を戻すのに不十分な場合があります。

パラメータ

表 33-2 INITIALIZE プロシージャのパラメータ

パラメータ	説明
seed	乱数の生成に使用するシード番号

SEED プロシージャ

このプロシージャは、シードをリセットします。

構文

```
DBMS_RANDOM.SEED (  
    seed IN BINARY_INTEGER);
```

パラメータ

表 33-3 INITIALIZE プロシージャのパラメータ

パラメータ	説明
seed	乱数の生成に使用するシード番号

RANDOM ファンクション

このファンクションは、乱数を取得します。

構文

```
DBMS_RANDOM.RANDOM  
    RETURN BINARY_INTEGER;
```

パラメータ

なし

例

```
my_random_number := Random;
```

TERMINATE プロシージャ

パッケージを終了するときは、TERMINATE プロシージャをコールします。

構文

```
DBMS_RANDOM.TERMINATE;
```

パラメータ

なし

DBMS_RECTIFIER_DIFF

DBMS_RECTIFIER_DIFF パッケージには、2つのレプリケート・サイト間のデータの不整合を検出して解決するための API が含まれています。

DBMS_RECTIFIER_DIFF パッケージ

サブプログラムの要約

表 34-1 DBMS_RECTIFIER_DIFF パッケージのサブプログラム

サブプログラム	説明
34-3 ページの DIFFERENCES プロシージャ	2 つの表の違いを判断します。
34-6 ページの RECTIFY プロシージャ	2 つの表の違いを解決します。

DIFFERENCES プロシージャ

このプロシージャは、2 つの表の違いを判断します。

構文

```
DBMS_RECTIFIER_DIFF.DIFFERENCES (
    sname1          IN  VARCHAR2,
    oname1          IN  VARCHAR2,
    reference_site   IN  VARCHAR2 := '',
    sname2          IN  VARCHAR2,
    oname2          IN  VARCHAR2,
    comparison_site  IN  VARCHAR2 := '',
    where_clause     IN  VARCHAR2 := '',
    { column_list    IN  VARCHAR2 := '',
    | array_columns  IN  dbms_utility.name_array, }
    missing_rows_sname IN  VARCHAR2,
    missing_rows_oname1 IN  VARCHAR2,
    missing_rows_oname2 IN  VARCHAR2,
    missing_rows_site IN  VARCHAR2 := '',
    max_missing      IN  INTEGER,
    commit_rows      IN  INTEGER := 500);
```

注意： このプロシージャはオーバーロードされています。column_list パラメータと array_columns パラメータは、両方同時には指定できません。

パラメータ

表 34-2 DIFFERENCES プロシージャのパラメータ

パラメータ	説明
sname1	REFERENCE_SITE にあるスキーマの名前。
oname1	REFERENCE_SITE にある表の名前。
reference_site	参照データベース・サイトの名前。デフォルトの NULL は、現行のサイトを示します。
sname2	COMPARISON_SITE にあるスキーマの名前。
oname2	COMPARISON_SITE にある表の名前。
comparison_site	比較データベース・サイトの名前。デフォルトの NULL は、現行のサイトを示します。

表 34-2 DIFFERENCES プロシージャのパラメータ

パラメータ	説明
where_clause	この制約を満たす行のみが比較のために選択されます。デフォルトの NULL は、すべての行を比較することを示します。
column_list	2 つの表について比較される 1 つ以上の列名のカンマで区切られたリスト。カンマの前後に空白を入れなくてください。デフォルトの NULL は、すべての列を比較することを示します。
array_columns	2 つの表について比較される列名の PL/SQL 表。索引は 1 から始まり、配列の最後の要素は NULL である必要があります。位置 1 が NULL の場合は、すべての列が使用されます。
missing_rows_sname	欠落行の情報がある表を含んでいるスキーマの名前。
missing_rows_onsame1	MISSING_ROWS_SITE にある表の名前で、REFERENCE サイトの表にあって COMPARISON サイトの表にはない行に関する情報、および COMPARISON サイトの表にあって REFERENCE サイトの表にはない行に関する情報が格納されています。
missing_rows_onsame2	欠落行に関する情報が格納されている MISSING_ROWS_SITE にある表の名前。この表には、次の 3 つの列があります。MISSING_ROWS_ONAME1 表にある行の ROWID、行が存在するサイトの名前、および行が欠落しているサイトの名前。
missing_rows_site	MISSING_ROWS_ONAME1 表と MISSING_ROWS_ONAME2 表が配置されているサイトの名前。デフォルトの NULL は、これらの表が現行のサイトに配置されていることを示します。
max_missing	missing_rows_onsame 表に挿入する必要がある最大行数を示す整数。max_missing の数値を超える行が欠落している場合は、その行数が missing_rows_onsame に挿入され、ルーチンは、さらに行が欠落しているかどうかを判断することなく正常に戻ります。この引数は、断片部分が違いすぎるために欠落行の表の項目が多くなり、継続する必要がなくなった場合に役に立ちます。max_missing が 1 未満または NULL の場合は、例外の badnumber が呼び出されます。
commit_rows	COMMIT が発生する前に参照表または比較表に挿入される、またはこれらの表から削除される最大行数。デフォルトでは、500 行挿入されるか、または 500 行削除されると COMMIT が発生します。空の文字列 (') または NULL は、1 つの表のすべての行が挿入または削除された後でのみ、COMMIT が発行されることを示します。

例外

表 34-3 DIFFERENCES プロシージャの例外

例外	説明
nosuchsite	データベース・サイトが見つかりません。
badnumber	COMMIT_ROWS パラメータが 1 未満です。
missingprimarykey	列のリストには、主キー（または、SET_COLUMNS と等価のもの）を含める必要があります。
badname	表またはスキーマ名が NULL または空の文字列です。
cannotbenull	パラメータに NULL を指定できません。
notshapeequivalent	比較される表の形態が同じではありません。形態とは、列数、表の列名および列のデータ型を指します。
unknowncolumn	列が存在しません。
unsupportedtype	サポートされていない型です。
dbms_repcat. commfailure	リモート・サイトにアクセスできません。
dbms_repcat. missingobject	表が存在しません。

制限事項

MISSING_ROWS_DATA 表に一意キー制約または主キー制約がある場合は、エラー ORA-00001（一意制約の違反）が発行されます。

RECTIFY プロシージャ

このプロシージャは、2つの表の違いを解決します。

構文

```
DBMS_RECTIFIER_DIFF.RECTIFY (
    sname1          IN  VARCHAR2,
    oname1          IN  VARCHAR2,
    reference_site   IN  VARCHAR2 := '',
    sname2          IN  VARCHAR2,
    oname2          IN  VARCHAR2,
    comparison_site  IN  VARCHAR2 := '',
    { column_list    IN  VARCHAR2 := '',
      | array_columns IN  dbms_utility.name_array, }
    missing_rows_sname IN  VARCHAR2,
    missing_rows_oname1 IN  VARCHAR2,
    missing_rows_oname2 IN  VARCHAR2,
    missing_rows_site  IN  VARCHAR2 := '',
    commit_rows       IN  INTEGER := 500);
```

注意： このプロシージャはオーバーロードされています。column_list パラメータと array_columns パラメータは、両方同時には指定できません。

パラメータ

表 34-4 RECTIFY プロシージャのパラメータ

パラメータ	説明
sname1	REFERENCE_SITE にあるスキーマの名前。
oname1	REFERENCE_SITE にある表の名前。
reference_site	参照データベース・サイトの名前。デフォルトの NULL は、現行のサイトを示します。
sname2	COMPARISON_SITE にあるスキーマの名前。
oname2	COMPARISON_SITE にある表の名前。
comparison_site	比較データベース・サイトの名前。デフォルトの NULL は、現行のサイトを示します。
column_list	2つの表について比較される 1 つ以上の列名のカンマで区切られたリスト。カンマの前後に空白を入れないでください。デフォルトの NULL は、すべての列を比較することを示します。

表 34-4 RECTIFY プロシージャのパラメータ

パラメータ	説明
array_columns	2つの表について比較される列名の PL/SQL 表。索引は 1 から始まり、配列の最後の要素は NULL である必要があります。位置 1 が NULL の場合は、すべての列が使用されます。
missing_rows_sname	欠落行の情報がある表を含んでいるスキーマの名前。
missing_rows_onsame1	MISSING_ROWS_SITE にある表の名前で、REFERENCE サイトの表にあって COMPARISON サイトの表にはない行に関する情報、および COMPARISON サイトの表にあって REFERENCE サイトの表にはない行に関する情報が格納されています。
missing_rows_onsame2	欠落行に関する情報が格納されている MISSING_ROWS_SITE にある表の名前。この表には、次の 3 つの列があります。MISSING_ROWS_ONAME1 表にある行の ROWID、行が存在するサイトの名前、および行が欠落しているサイトの名前。
missing_rows_site	MISSING_ROWS_ONAME1 表と MISSING_ROWS_ONAME2 表が配置されているサイトの名前。デフォルトの NULL は、これらの表が現行のサイトに配置されていることを示します。
commit_rows	COMMIT が発生する前に参照表または比較表に挿入される、またはこれらの表から削除される最大行数。デフォルトでは、500 行挿入されるか、または 500 行削除されると COMMIT が発生します。空の文字列 ('') または NULL は、1 つの表のすべての行が挿入または削除された後でのみ、COMMIT が発行されることを示します。

例外

表 34-5 RECTIFY プロシージャの例外

例外	説明
nosuchsite	データベース・サイトが見つかりません。
badnumber	COMMIT_ROWS パラメータが 1 未満です。
badname	表またはスキーマ名が NULL または空の文字列です。
dbms_repcat. commfailure	リモート・サイトにアクセスできません。
dbms_repcat. missingobject	表が存在しません。

DBMS_REFRESH

DBMS_REFRESH によって、トランザクション的に一貫性を保つ時点にまとめてリフレッシュできるスナップショットのグループを作成できます。

DBMS_REFRESH パッケージ

サブプログラムの要約

表 35-1 DBMS_REFRESH パッケージのサブプログラム

サブプログラム	説明
35-3 ページの ADD プロシージャ	リフレッシュ・グループにスナップショットを追加します。
35-4 ページの CHANGE プロシージャ	リフレッシュ・グループのリフレッシュ間隔を変更します。
35-6 ページの DESTROY プロシージャ	リフレッシュ・グループからすべてのスナップショットを削除して、そのリフレッシュ・グループを削除します。
35-7 ページの MAKE プロシージャ	リフレッシュ・グループのメンバー、およびグループのメンバーをリフレッシュする時間間隔を指定します。
35-9 ページの REFRESH プロシージャ	リフレッシュ・グループを手動でリフレッシュします。
35-10 ページの SUBTRACT プロシージャ	リフレッシュ・グループからスナップショットを削除します。

ADD プロシージャ

このプロシージャは、リフレッシュ・グループにスナップショットを追加します。

関連項目： 詳細は、『Oracle8i レプリケーション・マネージメント API リファレンス』の第 5 章「スナップショット・グループの作成」、および『Oracle8i レプリケーション・ガイド』の第 3 章「スナップショットの概念およびアーキテクチャ」を参照してください。

構文

```
DBMS_REFRESH.ADD (
    name      IN VARCHAR2,
    { list     IN VARCHAR2,
      | tab    IN DBMS_UTILITY.UNCL_ARRAY, }
    lax       IN BOOLEAN := FALSE);
```

注意： このプロシージャはオーバーロードされています。list パラメータと tab パラメータは、両方同時には指定できません。

パラメータ

表 35-2 ADD プロシージャのパラメータ

パラメータ	説明
name	メンバーを追加するリフレッシュ・グループの名前。
list	リフレッシュ・グループに追加するスナップショットのカンマで区切られたリスト（シノニムはサポートされていません）。
tab	カンマで区切られたリストのかわりに、DBMS_UTILITY.UNCL_ARRAY 型の PL/SQL 表を提供できます。このとき、各要素がスナップショット名です。最初のスナップショットを位置 1 に設定し、最後の位置には NULL を設定する必要があります。
lax	1 つのスナップショットは、一度に 1 つのリフレッシュ・グループにのみ所属できます。スナップショットを 1 つのグループから別のグループに移動する場合は、lax フラグを TRUE に設定する必要があります。Oracle はスナップショットを元のリフレッシュ・グループから自動的に削除し、そのリフレッシュ間隔を新規グループのリフレッシュ間隔に更新します。そうでない場合は、ADD へのコールでエラー・メッセージが生成されます。

CHANGE プロシージャ

このプロシージャは、リフレッシュ・グループのリフレッシュ間隔を変更します。

関連項目： 詳細は、『Oracle8i レプリケーション・ガイド』の第3章「スナップショットの概念およびアーキテクチャ」を参照してください。

構文

```
DBMS_REFRESH.CHANGE (
    name                IN VARCHAR2,
    next_date           IN DATE           := NULL,
    interval            IN VARCHAR2      := NULL,
    implicit_destroy    IN BOOLEAN       := NULL,
    rollback_seg        IN VARCHAR2      := NULL,
    push_deferred_rpc   IN BOOLEAN       := NULL,
    refresh_after_errors IN BOOLEAN      := NULL,
    purge_option        IN BINARY_INTEGER := NULL,
    parallelism         IN BINARY_INTEGER := NULL,
    heap_size           IN BINARY_INTEGER := NULL);
```

パラメータ

表 35-3 CHANGE プロシージャのパラメータ

パラメータ	説明
name	リフレッシュ間隔を変更するリフレッシュ・グループの名前。
next_date	次にリフレッシュを実行する日付。デフォルトでは、この日付は変更されずにそのまま残ります。
interval	リフレッシュ・グループにあるスナップショットの次回のリフレッシュ時期を計算するためのファンクション。この間隔は、リフレッシュの直前に計算されます。このため、リフレッシュの実行時間より長い間隔を選択する必要があります。デフォルトでは、この間隔は変更されずにそのまま残ります。
implicit_destroy	implicit_destroy フラグの値をリセットします。このフラグを設定すると、グループにメンバーが含まれていない場合、Oracle は自動的にそのグループを削除します。デフォルトでは、このフラグは変更されずにそのまま残ります。
rollback_seg	使用するロールバック・セグメントを変更できます。デフォルトでは、ロールバック・セグメントは変更されずにそのまま残ります。このパラメータをリセットしてデフォルトのロールバック・セグメントを使用するためには、引用符も含めて NULL を指定します。引用符を付けずに NULL を指定すると、現在使用しているロールバック・セグメントを変更しないことを示します。

表 35-3 CHANGE プロシージャのパラメータ

パラメータ	説明
push_deferred_rpc	更新可能なスナップショットでのみ使用します。スナップショットをリフレッシュする前に、スナップショットから関連マスターに変更を送信する場合は、このパラメータを TRUE に設定します。そうでない場合は、変更が一時的に失われたように表示される場合があります。デフォルトでは、このフラグは変更されずにそのまま残ります。
refresh_after_errors	更新可能なスナップショットでのみ使用します。スナップショットのマスターの DEFERROR ビューに未解決の競合が記録されていても、リフレッシュを続行する場合は、このパラメータを TRUE に設定します。デフォルトでは、このフラグは変更されずにそのまま残ります。
purge_option	<p>パラレル伝播メカニズムを使用する場合（つまり、parallelism に 1 以上を設定する場合）は、次のように指定します。</p> <ul style="list-style-type: none"> ■ 0 = パージなし ■ 1 = レイジー・パージ（デフォルト） ■ 2 = aggressive パージ <p>ほとんどの場合、レイジー・パージが最適な設定です。複数のマスター・レプリケーション・グループが別々のターゲット・サイトに送信され、1 つ以上のレプリケーション・グループへの更新やその送信がまれな場合、aggressive パージに設定してキューを減らします。すべてのレプリケーション・グループへの更新と送信がまれな場合は、「パージなし」に設定し、キューを減らすために時々「aggressive パージ」に設定して PUSH を実行してください。</p>
parallelism	0 はシリアル伝播を、 $n > 0$ は n 個のパラレル・サーバー・プロセスを使用するパラレル伝播を、1 は 1 つのパラレル・サーバー・プロセスのみ使用するパラレル伝播をそれぞれ指定します。
heap_size	パラレル伝播スケジューリングで同時に検査されるトランザクションの最大数。最適なパフォーマンスのためのデフォルト設定は Oracle が自動的に計算します。オラクル社カスタマ・サポート・センターから指示がない限り、このパラメータは設定しないでください。

DESTROY プロシージャ

このプロシージャは、リフレッシュ・グループからすべてのスナップショットを削除して、そのリフレッシュ・グループを削除します。

関連項目： 詳細は、『Oracle8i レプリケーション・ガイド』の第3章「スナップショットの概念およびアーキテクチャ」を参照してください。

構文

```
DBMS_REFRESH.DESTROY (  
    name      IN    VARCHAR2);
```

パラメータ

表 35-4 DESTROY プロシージャのパラメータ

パラメータ	説明
name	破棄するリフレッシュ・グループの名前

MAKE プロシージャ

このプロシージャは、リフレッシュ・グループのメンバー、およびグループのメンバーをリフレッシュする時間間隔を指定します。

関連項目： 詳細は、『Oracle8i レプリケーション・マネージメント API リファレンス』の第 5 章「スナップショット・グループの作成」、および『Oracle8i レプリケーション・ガイド』の第 3 章「スナップショットの概念およびアーキテクチャ」を参照してください。

構文

```
DBMS_REFRESH.MAKE (
    name                IN      VARCHAR2
  { list                IN      VARCHAR2,
    | tab               IN      DBMS_UTILITY.UNCL_ARRAY, }
  next_date            IN      DATE,
  interval              IN      VARCHAR2,
  implicit_destroy     IN      BOOLEAN          := FALSE,
  lax                  IN      BOOLEAN          := FALSE,
  job                  IN      BINARY_INTEGER  := 0,
  rollback_seg         IN      VARCHAR2        := NULL,
  push_deferred_rpc    IN      BOOLEAN          := TRUE,
  refresh_after_errors IN      BOOLEAN          := FALSE)
  purge_option          IN      BINARY_INTEGER := NULL,
  parallelism          IN      BINARY_INTEGER := NULL,
  heap_size            IN      BINARY_INTEGER := NULL);
```

注意： このプロシージャはオーバーロードされています。list パラメータと tab パラメータは、両方同時には指定できません。

表 35-5 MAKE プロシージャのパラメータ

パラメータ	説明
name	リフレッシュ・グループの識別に使用する一意の名前。リフレッシュ・グループは、表と同じ命名規則に従っている必要があります。
list	リフレッシュするスナップショットのカンマで区切られたリスト。(シノニムはサポートされていません)。スナップショットは、異なるスキーマに配置したり、異なるマスター表を持つことができますが、リスト内のすべてのスナップショットは、現行のデータベースにある必要があります。
tab	カンマで区切られたリストのかわりに、データ型 DBMS_UTILITY.UNCL_ARRAY を使用して、リフレッシュするスナップショット名の PL/SQL 表を提供できます。表に <i>n</i> 個のスナップショット名が含まれている場合、最初のスナップショットを位置 1 に設定し、 <i>n</i> + 1 の位置には NULL を設定する必要があります。
next_date	次にリフレッシュを実行する日付。
interval	グループにあるスナップショットの次回のリフレッシュ時期を計算するためのファンクション。このフィールドは、NEXT_DATE 値とともに使用されます。 たとえば、自分が使用する間隔として NEXT_DAY (SYSDATE+1, "MONDAY") を指定した場合、NEXT_DATE の値が月曜日になると、Oracle はスナップショットを月曜日ごとにリフレッシュします。この間隔は、リフレッシュの直前に計算されます。このため、リフレッシュの実行時間より長い間隔を選択する必要があります。
implicit_destroy	リフレッシュ・グループにメンバーがないときにそのグループを自動的に削除する場合は、このパラメータを TRUE に設定します。Oracle は、ユーザーが SUBTRACT プロシージャをコールしたときのみ、このフラグをチェックします。つまり、このフラグを設定しても、空のリフレッシュ・グループを作成することができます。
lax	1 つのスナップショットは、一度に 1 つのリフレッシュ・グループにのみ所属できます。スナップショットを既存のグループから新規のリフレッシュ・グループに移動する場合は、このパラメータを TRUE に設定する必要があります。Oracle はスナップショットを元のリフレッシュ・グループから自動的に削除し、そのリフレッシュ間隔を新規グループのリフレッシュ間隔に更新します。そうでない場合は、MAKE へのコールでエラー・メッセージが生成されます。
job	インポート・ユーティリティで必要です。デフォルト値の 0 (ゼロ) を使用してください。
rollback_seg	スナップショットのリフレッシュ中に使用するロールバック・セグメントの名前。デフォルトの NULL では、デフォルト・ロールバック・セグメントが使用されます。

表 35-5 MAKE プロシージャのパラメータ

パラメータ	説明
push_deferred_rpc	更新可能なスナップショットでのみ使用します。スナップショットをリフレッシュする前に、スナップショットから関連マスターに変更を送信する場合は、デフォルト値の TRUE を使用します。そうでない場合は、変更が一時的に失われたように表示される場合があります。
refresh_after_errors	更新可能なスナップショットでのみ使用します。スナップショットのマスターの DEFERROR ビューに未解決の競合が記録されていても、リフレッシュを続行する場合は、このパラメータを 0 (ゼロ) に設定します。
purge_option	パラレル伝播メカニズムを使用する場合 (つまり、並列性に 1 以上を設定) は、次のように指定します。0 = パージなし、1 = レイジー・パージ (デフォルト)、2 = aggressive パージ。ほとんどの場合、レイジー・パージが最適な設定です。 複数のマスター・レプリケーション・グループが別々のターゲット・サイトに送信され、1 つ以上のレプリケーション・グループへの更新やその送信がまれない場合、「aggressive パージ」に設定してキューを減らします。すべてのレプリケーション・グループへの更新と送信がまれない場合は、「パージなし」に設定し、キューを減らすために時々「aggressive パージ」に設定して PUSH を実行してください。
parallelism	0 はシリアル伝播を、 $n > 0$ は n 個のパラレル・サーバー・プロセスを使用するパラレル伝播を、1 は 1 つのパラレル・サーバー・プロセスのみ使用するパラレル伝播をそれぞれ指定します。
heap_size	パラレル伝播スケジューリングで同時に検査されるトランザクションの最大数。最適なパフォーマンスのためのデフォルト設定は Oracle が自動的に計算します。オラクル社カスタマ・サポート・センターから指示がない限り、このパラメータは設定しないでください。

REFRESH プロシージャ

このプロシージャでは、リフレッシュ・グループを手動でリフレッシュします。

関連項目：『Oracle8i レプリケーション・ガイド』の第 3 章「スナップショットの概念およびアーキテクチャ」を参照してください。

構文

```
DBMS_REFRESH.REFRESH (  
    name      IN      VARCHAR2);
```

表 35-6 REFRESH プロシージャのパラメータ

パラメータ	説明
name	手動でリフレッシュするリフレッシュ・グループの名前

SUBTRACT プロシージャ

このプロシージャは、リフレッシュ・グループからスナップショットを削除します。

関連項目：『Oracle8i レプリケーション・ガイド』の第3章「スナップショットの概念およびアーキテクチャ」を参照してください。

構文

```
DBMS_REFRESH.SUBTRACT (  
  name      IN    VARCHAR2,  
  { list    IN    VARCHAR2,  
    | tab    IN    DBMS_UTILITY.UNCL_ARRAY, }  
  lax       IN    BOOLEAN := FALSE);
```

注意： このプロシージャはオーバーロードされています。list パラメータと tab パラメータは、両方同時には指定できません。

パラメータ

表 35-7 SUBTRACT プロシージャのパラメータ

パラメータ	説明
name	メンバーを削除するリフレッシュ・グループの名前。
list	リフレッシュ・グループから削除するスナップショットのカンマで区切られたリスト（シノニムはサポートされていません）。スナップショットは、異なるスキーマに配置したり、異なるマスター表を持つことができます。ただし、リスト内のすべてのスナップショットは、現行のデータベースに存在している必要があります。
tab	カンマで区切られたリストのかわりに、データ型 DBMS_UTILITY.UNCL_ARRAY を使用して、リフレッシュするスナップショット名の PL/SQL 表を提供できます。表に <i>n</i> 個のスナップショット名が含まれている場合、最初のスナップショットを位置 1 に設定し、 <i>n</i> + 1 の位置には NULL を設定する必要があります。
lax	削除するスナップショットがリフレッシュ・グループのメンバーでないときに、Oracle がエラー・メッセージを生成するようにする場合、このパラメータを FALSE に設定します。

DBMS_REPAIR

DBMS_REPAIR には、データ破損修復プロシージャが含まれており、ユーザーは表と索引にある破損ブロックを検出して修復できます。可能な場合は破損をつき止め、再構築中または修復中にオブジェクトの使用を続行することができます。

注意： DBMS_REPAIR パッケージは、データベース管理者のみの使用を目的としています。アプリケーション開発者を対象にした機能ではありません。

関連項目： DBMS_REPAIR パッケージの使用方法は、『Oracle8i 管理者ガイド』を参照してください。

セキュリティ

このパッケージの所有者は sys です。他のユーザーに実行権限は付与されていません。

列挙型

DBMS_REPAIR パッケージは、パラメータ値の指定に使用するいくつかの列挙定数を定義します。列挙定数にはパッケージ名を接頭辞として付加する必要があります。たとえば、DBMS_REPAIR.TABLE_OBJECT と記述します。

表 36-1 は、パラメータと列挙定数の一覧です。

表 36-1 DBMS_REPAIR の列挙型

パラメータ	定数
object_type	TABLE_OBJECT, INDEX_OBJECT, CLUSTER_OBJECT
action	CREATE_ACTION, DROP_ACTION, PURGE_ACTION
table_type	REPAIR_TABLE, ORPHAN_TABLE
flags	SKIP_FLAG, NOSKIP_FLAG

注意： デフォルトの table_name は、table_type が REPAIR_TABLE のときは REPAIR_TABLE で、table_type が ORPHAN_TABLE のときは ORPHAN_KEY_TABLE です。

例外

表 36-2 DBMS_REPAIR の例外

例外	説明	アクション
942	指定した表が存在していないと、DROP_ACTION 時に DBMS_REPAIR.ADMIN_TABLES によって戻されます。	
955	指定した表がすでに存在していると、DBMS_REPAIR.CREATE_ACTION によって戻されます。	
24120	指定した DBMS_REPAIR プロシージャに無効なパラメータが渡されました。	有効なパラメータ値を指定するか、またはパラメータのデフォルトを使用してください。

表 36-2 DBMS_REPAIR の例外

例外	説明	アクション
24122	ブロック範囲の指定に誤りがあります。	BLOCK_START と BLOCK_END パラメータに正しい値を指定してください。
24123	指定した機能を使用しようとしたが、その機能はまだインプリメントされていません。	この機能は使用しないでください。
24124	ACTION パラメータに無効な値が指定されました。	ACTION パラメータには、CREATE_ACTION、PURGE_ACTION または DROP_ACTION のいずれかを指定してください。
24125	DBMS_REPAIR.CHECK_OBJECT の実行後に削除または切り捨てられたオブジェクトの破損ブロックを修正しようとした。	DBMS_REPAIR.ADMIN_TABLES で修復表をバージし、DBMS_REPAIR.CHECK_OBJECT を実行して、修正対象の破損ブロックがあるかどうかを確認してください。
24127	CREATE_ACTION 以外の ACTION で TABLESPACE パラメータが指定されました。	CREATE_ACTION 以外のアクションの実行時には、TABLESPACE は指定しないでください。
24128	パーティション化されていないオブジェクトに対して、パーティション名が指定されました。	パーティション名は、オブジェクトがパーティション化されているときのみ指定してください。
24129	接頭辞を指定しないで、table_name パラメータを渡そうとした。	有効な table_name パラメータを渡してください。
24130	存在しない修復表または親なし表を指定しようとした。	table_name パラメータに有効な値を指定してください。
24131	誤った定義内容の修復表または親なし表を指定しようとした。	正しく作成された表を参照する表名を指定してください。
24132	30 文字を超える表名を指定しようとした。	table_name パラメータに有効な値を指定してください。

サブプログラムの要約

表 36-3 DBMS_REPAIR パッケージのサブプログラム

サブプログラム	説明
36-4 ページの ADMIN_TABLES プロシージャ	DBMS_REPAIR パッケージの修復表と孤立したキー表に対して、作成、バージおよび削除処理を実行する管理ファンクションを提供します。
36-5 ページの CHECK_OBJECT プロシージャ	表または索引の破損を検出し、レポートします。
36-7 ページの DUMP_ORPHAN_KEYS プロシージャ	破損データ・ブロック内の行を指す索引エントリをレポートします。
36-9 ページの FIX_CORRUPT_BLOCKS プロシージャ	CHECK_OBJECT によって破損が検出されたブロックにソフトウェア破損のマークを付けます。
36-10 ページの REBUILD_FREELISTS プロシージャ	オブジェクトの空きリストを再作成します。
36-11 ページの SKIP_CORRUPT_BLOCKS プロシージャ	表と索引のスキャン時に破損マークのあるブロックを無視するか、または破損マークのあるブロックが検出された場合に ORA-1578 をレポートするかを設定します。

ADMIN_TABLES プロシージャ

このプロシージャは、DBMS_REPAIR パッケージの修復表と孤立したキー表に対する管理ファンクションを提供します。

構文

```
DBMS_REPAIR.ADMIN_TABLES (  
    table_name IN    VARCHAR2,  
    table_type IN    BINARY_INTEGER,  
    action      IN    BINARY_INTEGER,  
    tablespace IN    VARCHAR2      DEFAULT NULL);
```

パラメータ

表 36-4 ADMIN_TABLES プロシージャのパラメータ

パラメータ	説明
table_name	処理する表の名前。指定した table_type に基づいて、ORPHAN_KEY_TABLE または REPAIR_TABLE をデフォルト設定します。指定するときは、表名には適切な接頭辞（ORPHAN_ または REPAIR_）が必要です。
table_type	表の型。ORPHAN_TABLE または REPAIR_TABLE のいずれかです。 36-2 ページの「 列挙型 」を参照してください。
action	実行する管理アクションを示します。 CREATE_ACTION、PURGE_ACTION または DROP_ACTION のいずれかです。CREATE_ACTION の指定時に、表がすでに存在しているとエラーが戻されます。PURGE_ACTION を指定すると、存在しないオブジェクトに関連付けられている表内の行はすべて削除されます。DROP_ACTION の指定時に、表が存在していないとエラーが戻されます。 CREATE_ACTION と DROP_ACTION を指定すると、DBA_<table_name> という名前の関連ビューが、それぞれ作成または削除されます。このビューは、存在しないオブジェクトに関連付けられている行を排除するように定義されています。 SYS スキーマ内に作成されます。 36-2 ページの「 列挙型 」を参照してください。
tablespace	表の作成時に使用する表領域を示します。 デフォルトでは、SYS のデフォルト表領域が使用されます。CREATE_ACTION 以外のときに表領域を指定するとエラーが戻されます。

CHECK_OBJECT プロシージャ

このプロシージャは、指定したオブジェクトをチェックし、破損と修復指示に関する情報を修復表に移入します。

妥当性チェックでは、オブジェクト内のすべてのブロックがチェックされます。オブジェクトの一部をチェック対象とする場合は、オプションで、DBA 範囲、パーティション名またはサブパーティション名を指定することもできます。

構文

```
DBMS_REPAIR.CHECK_OBJECT (
  schema_name      IN  VARCHAR2,
  object_name       IN  VARCHAR2,
  partition_name    IN  VARCHAR2      DEFAULT NULL,
  object_type       IN  BINARY_INTEGER DEFAULT TABLE_OBJECT,
  repair_table_name IN  VARCHAR2      DEFAULT 'REPAIR_TABLE',
  flags             IN  BINARY_INTEGER DEFAULT NULL,
  relative_fno      IN  BINARY_INTEGER DEFAULT NULL,
  block_start       IN  BINARY_INTEGER DEFAULT NULL,
  block_end         IN  BINARY_INTEGER DEFAULT NULL,
  corrupt_count     OUT BINARY_INTEGER);
```

パラメータ

表 36-5 CHECK_OBJECT プロシージャのパラメータ

パラメータ	説明
schema_name	チェックするオブジェクトのスキーマ名。
object_name	チェックする表または索引の名前。
partition_name	チェックするパーティションまたはサブパーティションの名前。 パーティション・オブジェクトで、partition_name が指定されていない場合は、すべてのパーティションとサブパーティションがチェックされます。パーティション・オブジェクトで、指定したパーティションにサブパーティションが含まれている場合は、すべてのサブパーティションがチェックされます。
object_type	処理するオブジェクトの型。TABLE_OBJECT（デフォルト）または INDEX_OBJECT のいずれかです。 36-2 ページの「 列挙型 」を参照してください。
repair_table_name	情報を移入する修復表の名前。 この表は、SYS スキーマに存在している必要があります。修復表を作成するには、admin_tables プロシージャを使用します。デフォルト名は REPAIR_TABLE です。
flags	将来使用のために確保。
relative_fno	相対ファイル番号。ブロック範囲の指定時に使用します。
block_start	ブロック範囲を指定する場合に、最初に処理するブロックを指定します。オブジェクトが単一表、パーティションまたはサブパーティションの場合のみ指定できます。

表 36-5 CHECK_OBJECT プロシージャのパラメータ

パラメータ	説明
block_end	ブロック範囲を指定する場合に、最後に処理するブロックを指定します。オブジェクトが単一表、パーティションまたはサブパーティションの場合のみ指定できます。block_start または block_end のいずれか一方のみ指定した場合、他方の値は、ファイル内の第 1 ブロックまたは最終ブロックにそれぞれデフォルト設定されます。
corrupt_count	レポートされた破損数。

DUMP_ORPHAN_KEYS プロシージャ

このプロシージャは、破損データ・ブロック内の行を指す索引エントリをレポートします。検出された該当索引エントリごとに、指定した親なし表に行が挿入されます。

修復表が指定されている場合は、ソフトウェア破損のマークがあるすべてのデータ・ブロックの他に、ベース表に関連付けられている破損ブロックが処理されます。修復表が指定されていない場合は、破損マークのあるブロックのみ処理されます。

この情報は、表内で失われた行を再構築する場合や診断の目的に使用されます。

構文

```
DBMS_REPAIR.DUMP_ORPHAN_KEYS (
    schema_name      IN  VARCHAR2,
    object_name      IN  VARCHAR2,
    partition_name   IN  VARCHAR2      DEFAULT NULL,
    object_type      IN  BINARY_INTEGER DEFAULT INDEX_OBJECT,
    repair_table_name IN  VARCHAR2      DEFAULT 'REPAIR_TABLE',
    orphan_table_name IN  VARCHAR2      DEFAULT 'ORPHAN_KEYS_TABLE',
    flags            IN  BINARY_INTEGER DEFAULT NULL,
    key_count        OUT BINARY_INTEGER);
```

パラメータ

表 36-6 DUMP_ORPHAN_KEYS プロシージャのパラメータ

パラメータ	説明
schema_name	スキーマ名
object_name	オブジェクト名

表 36-6 DUMP_ORPHAN_KEYS プロシージャのパラメータ

パラメータ	説明
partition_name	処理するパーティションまたはサブパーティションの名前。 パーティション・オブジェクトで、partition_name を指定しない場合は、すべてのパーティションとサブパーティションが処理されます。パーティション・オブジェクトで、指定したパーティションにサブパーティションが含まれている場合は、すべてのサブパーティションが処理されます。
object_type	処理するオブジェクトの型。デフォルトは INDEX_OBJECT です。 36-2 ページの「 列挙型 」を参照してください。
repair_table_name	ベース表の破損ブロックに関する情報を含んだ修復表の名前。 指定した表は、SYS スキーマに存在する必要があります。表を作成するには、admin_tables プロシージャを使用します。
orphan_table_name	破損データ・ブロック内の行を参照する各索引エントリに関する情報を移入する孤立したキー表の名前。 指定した表は、SYS スキーマに存在する必要があります。表を作成するには、admin_tables プロシージャを使用します。
flags	将来使用のために確保。
key_count	処理された索引エントリ数。

FIX_CORRUPT_BLOCKS プロシージャ

このプロシージャは、check_object プロシージャによって事前に生成された修復表の情報に基づいて、指定したオブジェクト内の破損ブロックを修正します。

ブロックに変更を加える前に、そのブロックがまだ破損状態であることを確認するチェックが行われます。破損ブロックは、そのブロックにソフトウェア破損のマークを付けることによって修復されます。修復が有効になると、修復表内の関連行が修正タイムスタンプで更新されます。

構文

```
DBMS_REPAIR.FIX_CORRUPT_BLOCKS (  
    schema_name      IN  VARCHAR2,  
    object_name       IN  VARCHAR2,  
    partition_name    IN  VARCHAR2      DEFAULT NULL,  
    object_type        IN  BINARY_INTEGER DEFAULT TABLE_OBJECT,  
    repair_table_name  IN  VARCHAR2      DEFAULT 'REPAIR_TABLE',  
    flags              IN  BINARY_INTEGER DEFAULT NULL,  
    fix_count          OUT BINARY_INTEGER);
```

パラメータ

表 36-7 FIX_CORRUPT_BLOCKS プロシージャのパラメータ

パラメータ	説明
schema_name	スキーマ名。
object_name	修正対象の破損ブロックがあるオブジェクトの名前。
partition_name	処理するパーティションまたはサブパーティションの名前。 パーティション・オブジェクトで、partition_name を指定しない場合は、すべてのパーティションとサブパーティションが処理されます。パーティション・オブジェクトで、指定したパーティションにサブパーティションが含まれている場合は、すべてのサブパーティションが処理されます。
object_type	処理するオブジェクトの型。TABLE_OBJECT（デフォルト）または INDEX_OBJECT のいずれかです。 36-2 ページの「 列挙型 」を参照してください。
repair_table_name	修復指示を含んだ修復表の名前。 SYS スキーマに存在している必要があります。
flags	将来使用のために確保。
fix_count	修正されたブロック数。

REBUILD_FREELISTS プロシージャ

このプロシージャは、指定したオブジェクトの空きリストを再作成します。すべての空きブロックは、マスター空きリストに格納されます。その他の空きリストはすべてゼロになります。

オブジェクトに複数の空きリスト・グループがある場合、空きブロックは、ラウンドロビン方式で異なるグループに割り当てられ、すべての空きリスト間に配分されます。

構文

```
DBMS_REPAIR.REBUILD_FREELISTS (  
    schema_name      IN VARCHAR2,  
    partition_name   IN VARCHAR2      DEFAULT NULL,  
    object_type      IN BINARY_INTEGER DEFAULT TABLE_OBJECT);
```

パラメータ

表 36-8 REBUILD_FREELISTS プロシージャのパラメータ

パラメータ	説明
schema_name	スキーマ名。
object_name	空きリストを再作成するオブジェクトの名前。
partition_name	空きリストを再作成するパーティションまたはサブパーティションの名前。 パーティション・オブジェクトの場合に partition_name を指定しない場合は、すべてのパーティションとサブパーティションが処理されます。パーティション・オブジェクトので、指定したパーティションにサブパーティションが含まれている場合は、すべてのサブパーティションが処理されます。
object_type	処理するオブジェクトの型。TABLE_OBJECT（デフォルト）または INDEX_OBJECT のいずれかです。 36-2 ページの「 列挙型 」を参照してください。

SKIP_CORRUPT_BLOCKS プロシージャ

このプロシージャは、指定したオブジェクトの索引と表のスキャン時に、破損ブロックのスキップを使用可能または使用禁止にします。

オブジェクトが表のときは、スキップが表とその索引に適用されます。オブジェクトがクラスタのときは、クラスタ内のすべての表とその各索引に適用されます。

注意： ペース表に対して DBMS_REPAIR.SKIP_CORRUPT_BLOCKS を設定した後、破損している索引で索引範囲スキャンを実行すると、破損ブランチ・ブロックと破損ルート・ブロックはスキップされません。ルート以外の破損しているリーフ・ブロックのみがスキップされます。

構文

```
DBMS_REPAIR.SKIP_CORRUPT_BLOCKS (  
    schema_name IN VARCHAR2,  
    object_name IN VARCHAR2,  
    object_type IN BINARY_INTEGER DEFAULT TABLE_OBJECT,  
    flags        IN BINARY_INTEGER DEFAULT SKIP_FLAG);
```

パラメータ

表 36-9 SKIP_CORRUPT_BLOCKS プロシージャのパラメータ

パラメータ	説明
schema_name	処理するオブジェクトのスキーマ名。
object_name	オブジェクト名。
partition_name (optional)	処理するパーティションまたはサブパーティションの名前。 パーティション・オブジェクトで、partition_name を指定しない場合は、すべてのパーティションとサブパーティションが処理されます。パーティション・オブジェクトで、指定したパーティションにサブパーティションが含まれている場合は、すべてのサブパーティションが処理されます。
object_type	処理するオブジェクトの型。TABLE_OBJECT（デフォルト）または CLUSTER_OBJECT のいずれかです。 36-2 ページの「 列挙型 」を参照してください。

表 36-9 SKIP_CORRUPT_BLOCKS プロシージャのパラメータ

パラメータ	説明
flags	SKIP_FLAG を指定すると、索引と表のスキャン時に、そのオブジェクトのソフトウェア破損ブロックのスキップがオンになります。NOSKIP_FLAG を指定すると、ソフトウェア破損ブロックが検出されたときに、ORA-1578 エラーが戻されます。 36-2 ページの「 列挙型 」を参照してください。

DBMS_REPCAT

DBMS_REPCAT は、レプリケーション・カタログとレプリケーション環境を管理および更新するためのルーチンを提供します。

DBMS_REPCAT パッケージ

サブプログラムの要約

表 37-1 DBMS_REPCAT パッケージのサブプログラム

サブプログラム	説明
37-6 ページの ADD_GROUPED_COLUMN プロシージャ	既存の列グループにメンバーを追加します。
37-7 ページの ADD_MASTER_DATABASE プロシージャ	レプリケート環境に別のマスター・サイトを追加します。
37-9 ページの ADD_PRIORITY_datatype プロシージャ	優先グループにメンバーを追加します。
37-10 ページの ADD_SITE_PRIORITY_SITE プロシージャ	サイト優先グループに新規サイトを追加します。
37-12 ページの ADD_conflicttype_RESOLUTION プロシージャ	更新、削除または一意性の競合の解消方法を指定します。
37-16 ページの ALTER_MASTER_PROPAGATION プロシージャ	指定したマスター・サイトで指定したオブジェクト・グループの伝播方法を変更します。
37-17 ページの ALTER_MASTER_REPOBJECT プロシージャ	レプリケート環境内のオブジェクトを変更します。
37-19 ページの ALTER_PRIORITY プロシージャ	指定した優先グループ・メンバーに関連付けられている優先順位レベルを変更します。
37-20 ページの ALTER_PRIORITY_datatype プロシージャ	優先グループ内のメンバーの値を変更します。
37-22 ページの ALTER_SITE_PRIORITY プロシージャ	指定したサイトに関連付けられている優先順位レベルを変更します。
37-23 ページの ALTER_SITE_PRIORITY_SITE プロシージャ	指定した優先順位レベルに関連付けられているサイトを変更します。
37-24 ページの ALTER_SNAPSHOT_PROPAGATION プロシージャ	現行のスナップショット・サイトで指定したオブジェクト・グループの伝播方法を変更します。
37-25 ページの CANCEL_STATISTICS プロシージャ	表の更新競合、一意性競合および削除競合の正常な解消に関する統計の収集を停止します。
37-26 ページの COMMENT_ON_COLUMN_GROUP プロシージャ	列グループの ALL_REPCOLUMN_GROUP ビューのコメント・フィールドを更新します。

表 37-1 DBMS_REPCAT パッケージのサブプログラム

サブプログラム	説明
37-27 ページの COMMENT_ON_PRIORITY_GROUP/COMMENT_ON_SITE_PRIORITY プロシージャ	(サイト) 優先グループの ALL_REPPRIORITY_GROUP ビューのコメント・フィールドを更新します。
37-28 ページの COMMENT_ON_REPGROUP プロシージャ	レプリケート・マスター・グループの ALL_REPGROUP ビューのコメント・フィールドを更新します。
37-29 ページの COMMENT_ON_REPOBJECT プロシージャ	レプリケート・オブジェクトの ALL_REPOBJECT ビューのコメント・フィールドを更新します。
37-31 ページの COMMENT_ON_REPSITES プロシージャ	レプリケート・サイトの ALL_REPSITE ビューのコメント・フィールドを更新します。
37-32 ページの COMMENT_ON_SNAPSHOT_REPSITES プロシージャ	スナップショット・サイトの ALL_REPGROUP ビューの SCHEMA_COMMENT フィールドを更新します。
37-33 ページの COMMENT_ON_conflicttype_RESOLUTION プロシージャ	競合解消ルーチンの ALL_REPRESOLUTION ビューのコメント・フィールドを更新します。
37-35 ページの COMPARE_OLD_VALUES プロシージャ	更新と削除のために、レプリケート表の各非キー列に対して、各マスター・サイトの元の列値を比較します。
37-37 ページの CREATE_MASTER_REPGROUP プロシージャ	新規に、空で停止中のマスター・レプリケーション・オブジェクト・グループを作成します。
37-38 ページの CREATE_MASTER_REPOBJECT プロシージャ	オブジェクトをレプリケート・オブジェクトとして定義します。
37-41 ページの CREATE_SNAPSHOT_REPGROUP プロシージャ	ローカル・データベース内に、新規で空のスナップショット・レプリケーション・オブジェクト・グループを作成します。
37-42 ページの CREATE_SNAPSHOT_REPOBJECT プロシージャ	スナップショット・サイトにレプリケート・オブジェクトを追加します。
37-44 ページの DEFINE_COLUMN_GROUP プロシージャ	空の列グループを作成します。
37-45 ページの DEFINE_PRIORITY_GROUP プロシージャ	レプリケート・マスター・グループに新規の優先グループを作成します。
37-47 ページの DEFINE_SITE_PRIORITY プロシージャ	レプリケート・マスター・グループに新規のサイト優先グループを作成します。

表 37-1 DBMS_REPCAT パッケージのサブプログラム

サブプログラム	説明
37-48 ページの DO_DEFERRED_REPCAT_ADMIN プロシージャ	現行のマスター・サイトで指定したレプリケート・マスター・グループまたはすべてのマスター・サイトに対して未処理のローカル遅延管理手順を実行します。
37-49 ページの DROP_COLUMN_GROUP プロシージャ	列グループを削除します。
37-50 ページの DROP_GROUPED_COLUMN プロシージャ	列グループからメンバーを削除します。
37-51 ページの DROP_MASTER_REPGROUP プロシージャ	現行のサイトからレプリケート・マスター・グループを削除します。
37-52 ページの DROP_MASTER_REPOBJECT プロシージャ	レプリケート・マスター・グループからレプリケート・オブジェクトを削除します。
37-53 ページの DROP_PRIORITY プロシージャ	優先順位レベルに従って優先グループのメンバーを削除します。
37-54 ページの DROP_PRIORITY_GROUP プロシージャ	指定したレプリケート・マスター・グループの優先グループを削除します。
37-55 ページの DROP_PRIORITY_datatype プロシージャ	値による優先グループのメンバーを削除します。
37-57 ページの DROP_SITE_PRIORITY プロシージャ	指定したレプリケート・マスター・グループのサイト優先グループを削除します。
37-58 ページの DROP_SITE_PRIORITY_SITE プロシージャ	サイト優先グループから、名前を指定してサイトを削除します。
37-59 ページの DROP_SNAPSHOT_REPGROUP プロシージャ	レプリケート環境からスナップショット・サイトを削除します。
37-60 ページの DROP_SNAPSHOT_REPOBJECT プロシージャ	スナップショット・サイトからレプリケート・オブジェクトを削除します。
37-61 ページの DROP_conflicttype_RESOLUTION プロシージャ	更新、削除または一意性競合解消ルーチンを削除します。
37-63 ページの EXECUTE_DDL プロシージャ	各マスター・サイトで実行する DDL を提供します。
37-64 ページの GENERATE_REPLICATION_SUPPORT プロシージャ	レプリケーションのサポートに必要なトリガー、パッケージおよびプロシージャを生成します。

表 37-1 DBMS_REPCAT パッケージのサブプログラム

サブプログラム	説明
37-66 ページの GENERATE_SNAPSHOT_SUPPORT プロシージャ	トリガーを起動し、更新可能スナップショットのレプリケーションまたはプロシージャ型レプリケーションのサポートに必要なパッケージを生成します。
37-68 ページの MAKE_COLUMN_GROUP プロシージャ	1 つ以上のメンバーを含んだ新しい列グループを作成します。
37-69 ページの PURGE_MASTER_LOG プロシージャ	指定した識別番号、ソースまたはレプリケート・マスター・グループに関連している RepCatLog 内のローカル・メッセージを削除します。
37-70 ページの PURGE_STATISTICS プロシージャ	ALL_REPRESOLUTION_STATISTICS ビューから情報を削除します。
37-71 ページの REFRESH_SNAPSHOT_REPGROUP プロシージャ	スナップショット・サイトのオブジェクト・グループを、関連するマスター・サイトからの最新データでリフレッシュします。
37-72 ページの REGISTER_SNAPSHOT_REPGROUP プロシージャ	DBA_REGISTERED_SNAPSHOT_GROUPS への挿入、変更または削除を行うことによって、各マスター・サイトのスナップショット管理を容易にします。
37-74 ページの REGISTER_STATISTICS プロシージャ	表の更新競合、削除競合および一意性競合の正常な解消に関する情報を収集します。
37-75 ページの RELOCATE_MASTERDEF プロシージャ	マスター定義サイトをレプリケート環境内の別のマスター・サイトに変更します。
37-76 ページの REMOVE_MASTER_DATABASES プロシージャ	レプリケート環境から 1 つ以上のマスター・データベースを削除します。
37-77 ページの REPCAT_IMPORT_CHECK プロシージャ	レプリケート・オブジェクトまたはアドバンスド・レプリケーション機能で使用されたオブジェクトのエクスポートまたはインポートの実行後、レプリケート・マスター・グループ内のオブジェクトの識別子と状態値が適切かを確認します。
37-78 ページの RESUME_MASTER_ACTIVITY プロシージャ	レプリケート環境の休止後、通常のレプリケーション・アクティビティを再開します。
37-79 ページの SEND_OLD_VALUES プロシージャ	更新と削除のために、レプリケート表の各非キー列の元の値を送信するかどうかを指定します。
37-81 ページの SET_COLUMNS プロシージャ	行レベル・レプリケーションの使用時に、主キーのかわりに使用する代替列または列グループを指定し、表のどの列を比較するかを判断します。
37-83 ページの SUSPEND_MASTER_ACTIVITY プロシージャ	オブジェクト・グループのレプリケーション・アクティビティを中断します。

表 37-1 DBMS_REPCAT パッケージのサブプログラム

サブプログラム	説明
37-83 ページの SWITCH_SNAPSHOT_MASTER プロシージャ	スナップショット・レプリケート・マスター・グループのマスター・データベースを別のマスター・サイトに変更します。
37-84 ページの UNREGISTER_SNAPSHOT_REPGROUP プロシージャ	repcat\$_repsite から挿入、変更または削除を行うことによって、各マスター・サイトのスナップショット管理を容易にします。
37-85 ページの VALIDATE ファンクション	複数マスター・レプリケーション環境のキー状態が正しいかどうかを検証します。
37-88 ページの WAIT_MASTER_LOG プロシージャ	マスター・サイトに非同期で伝播された変更内容が適用されたかどうかを判断します。

ADD_GROUPED_COLUMN プロシージャ

このプロシージャは、既存の列グループにメンバーを追加します。このプロシージャは、マスター定義サイトからコールする必要があります。

構文

```
DBMS_REPCAT.ADD_GROUPED_COLUMN (  
    sname                IN    VARCHAR2,  
    oname                IN    VARCHAR2,  
    column_group         IN    VARCHAR2,  
    list_of_column_names IN    VARCHAR2 | DBMS_REPCAT.VARCHAR2s);
```

パラメータ

表 37-2 ADD_GROUPED_COLUMN プロシージャのパラメータ

パラメータ	説明
sname	レプリケート表が置かれているスキーマ。
oname	列グループが関連付けられているレプリケート表の名前。
column_group	メンバーを追加する列グループの名前。
list_of_column_names	指定した列グループに追加する列の名前。列名は、名前のカンマで区切られたリストまたは PL/SQL 表のいずれかで示します。PL/SQL 表は、DBMS_REPCAT.VARCHAR2 型にしてください。表内のすべての列を含んだ列グループを作成するには、単一の値 '*' を使用します。

表 37-3 ADD_GROUPED_COLUMN プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
missingobject	指定した表が存在しません。
missinggroup	指定した列グループが存在しません。
missingcolumn	指定した列が、指定した表内に存在していません。
duplicatecolumn	指定した列は、すでに別の列グループのメンバーです。
missingschema	指定のスキーマが存在しません。
notquiesced	指定した表が属しているオブジェクト・グループが停止中ではありません。

ADD_MASTER_DATABASE プロシージャ

このプロシージャは、レプリケート環境に別のマスター・サイトを追加します。また、すべてのトリガーと関連するパッケージを既存のマスター・サイトで再生成します。このプロシージャは、マスター定義サイトからコールする必要があります。

構文

```
DBMS_REPCAT.ADD_MASTER_DATABASE (
    gname          IN   VARCHAR2,
    master         IN   VARCHAR2,
    use_existing_objects IN  BOOLEAN := TRUE,
    copy_rows      IN   BOOLEAN := TRUE,
    comment        IN   VARCHAR2 := '',
    propagation_mode IN  VARCHAR2 := 'ASYNCHRONOUS',
    fname         IN   VARCHAR2 := NULL);
```

パラメータ

表 37-4 ADD_MASTER_DATABASE プロシージャのパラメータ

パラメータ	説明
gname	レプリケートするオブジェクト・グループの名前。このオブジェクト・グループは、マスター定義サイトにすでに存在している必要があります。
master	新規マスター・データベースの完全修飾データベース名。
use_existing_objects	スキーマ内にすでに存在するオブジェクトと同じ型、同じ形式のオブジェクトを、新規マスター・サイトで再使用する場合は、TRUE を指定します。これらの変更内容の適用方法の詳細は、『Oracle8i レプリケーション・ガイド』の第 2 章「基本概念およびアーキテクチャ」を参照してください。
copy_rows	新規マスター・サイトの表の初期の内容を、マスター定義サイトの表の内容と一致させる場合は TRUE を指定します。
comment	このパラメータは DBA_REPSITES ビューの MASTER_COMMENT フィールドに追加されます。
propagation_mode	新規マスター・データベースとの間の変更内容の送受信方法を示します。指定できる値は SYNCHRONOUS と ASYNCHRONOUS です。
fname	内部使用のためのパラメータ。オラクル社カスタム・サポート・センターから指示がない限り、このパラメータは設定しないでください。

例外

表 37-5 ADD_MASTER_DATABASE プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
notquiesced	レプリケート・マスター・グループが中断されていません。
missingrepgroup	オブジェクト・グループが、指定したデータベース・サイトに存在していません。
commfailure	新規マスターにアクセスできません。
typefailure	伝播モードの指定に誤りがあります。
notcompat	互換モードは 7.3.0.0 以上であることが必要です。
duplrepgrp	マスター・サイトがすでに存在しています。

ADD_PRIORITY_datatype プロシージャ

このプロシージャは、優先グループにメンバーを追加します。このプロシージャは、マスター定義サイトからコールする必要があります。コールする必要があるプロシージャは、priority 列のデータ型によって決まります。このプロシージャは、priority 列の有効な値ごとに 1 回ずつコールしてください。

関連項目： 競合の解消方法の詳細は、『Oracle8i レプリケーション・ガイド』を参照してください。

構文

```
DBMS_REPCAT.ADD_PRIORITY_datatype (
    gname          IN   VARCHAR2,
    pgroup         IN   VARCHAR2,
    value          IN   datatype,
    priority       IN   NUMBER);
```

datatype には、次のいずれかを指定します。

```
{ NUMBER
| VARCHAR2
| CHAR
| DATE
| RAW
| NCHAR
| NVARCHAR2 }
```

パラメータ

表 37-6 ADD_PRIORITY_datatype プロシージャのパラメータ

パラメータ	説明
gname	優先グループを作成するレプリケート・マスター・グループ。
pgroup	優先グループの名前。
value	優先グループ・メンバーの値。この値は、この優先グループを使用している表の priority 列に関連する候補値の 1 つです。
priority	この値の優先順位。数値が大きいくほど優先順位は高くなります。

例外

表 37-7 ADD_PRIORITY_datatype プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
duplicatevalue	指定した値が、優先グループ内にすでに存在しています。
duplicatepriority	指定した優先順位が優先グループ内にすでに存在しています。
missingrepgroup	指定したレプリケート・マスター・グループが存在しません。
missingprioritygroup	指定した優先グループが存在しません。
typefailure	優先グループに対して指定した値のデータ型が正しくありません。
notquiesced	指定したレプリケート・マスター・グループが停止中ではありません。

ADD_SITE_PRIORITY_SITE プロシージャ

このプロシージャは、サイト優先グループに新規サイトを追加します。このプロシージャは、マスター定義サイトからコールする必要があります。

関連項目： 競合の解消方法の詳細は、『Oracle8i レプリケーション・ガイド』を参照してください。

構文

```
DBMS_REPCAT.ADD_SITE_PRIORITY_SITE (  
  gname          IN   VARCHAR2,  
  name           IN   VARCHAR2  
  site           IN   VARCHAR2,  
  priority       IN   NUMBER);
```

パラメータ

表 37-8 ADD_SITE_PRIORITY_SITE プロシージャのパラメータ

パラメータ	説明
gname	サイトをグループに追加するレプリケート・マスター・グループ。
name	メンバーを追加するサイト優先グループの名前。
site	追加するサイトのグローバル・データベース名。
priority	追加するサイトの優先順位レベル。数値が大きいほど優先順位レベルは高くなります。

例外

表 37-9 ADD_SITE_PRIORITY_SITE プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
missingrepgroup	指定したレプリケート・マスター・グループが存在しません。
missingpriority	指定したサイト優先グループが存在しません。
duplicatepriority	指定した優先順位レベルが、グループ内の別のサイト用にすでに存在しています。
duplicatevalue	指定したサイトが、サイト優先グループ内にすでに存在しています。
notquiesced	レプリケート・マスター・グループが停止中ではありません。

ADD_conflicttype_RESOLUTION プロシージャ

このプロシージャは、更新、削除または一意性の競合の解消方法を指定します。このプロシージャは、マスター定義サイトからコールする必要があります。コールする必要があるプロシージャは、ルーチンが解消する競合の型によって決まります。

表 37-10 ADD_conflicttype_RESOLUTION プロシージャ

競合の型	プロシージャ名
update	ADD_UPDATE_RESOLUTION
uniqueness	ADD_UNIQUE_RESOLUTION
delete	ADD_DELETE_RESOLUTION

関連項目： 更新競合の解消方法の指定、一意性競合の解消方法の選択、および削除競合の解消方法の割当ては、『Oracle8i レプリケーション・ガイド』を参照してください。

構文

```
DBMS_REPCAT.ADD_UPDATE_RESOLUTION (
    sname          IN   VARCHAR2,
    oname          IN   VARCHAR2,
    column_group   IN   VARCHAR2,
    sequence_no    IN   NUMBER,
    method         IN   VARCHAR2,
    parameter_column_name IN VARCHAR2 | DBMS_REPCAT.VARCHAR2s,
    priority_group  IN   VARCHAR2      := NULL,
    function_name   IN   VARCHAR2      := NULL,
    comment        IN   VARCHAR2      := NULL);
```

```
DBMS_REPCAT.ADD_DELETE_RESOLUTION (
    sname          IN   VARCHAR2,
    oname          IN   VARCHAR2,
    sequence_no    IN   NUMBER,
    parameter_column_name IN VARCHAR2 | DBMS_REPCAT.VARCHAR2s,
    function_name   IN   VARCHAR2,
    comment        IN   VARCHAR2      := NULL
    method         IN   VARCHAR2      := 'USER FUNCTION');
```



```
DBMS_REPCAT.ADD_UNIQUE_RESOLUTION(  
    sname          IN    VARCHAR2,  
    oname          IN    VARCHAR2,  
    constraint_name IN    VARCHAR2,  
    sequence_no    IN    NUMBER,  
    method         IN    VARCHAR2,  
    parameter_column_name IN VARCHAR2 | DBMS_REPCAT.VARCHAR2s,  
    function_name   IN    VARCHAR2      := NULL,  
    comment        IN    VARCHAR2      := NULL);
```

パラメータ

表 37-11 ADD_conflicttype_RESOLUTION プロシージャのパラメータ

パラメータ	説明
sname	レプリケートする表が含まれているスキーマの名前。
oname	競合解消ルーチンを追加する表の名前。
column_group	競合解消ルーチンを追加する列グループの名前。更新競合解消ルーチンのみが列グループを必要とします。
constraint_name	競合解消ルーチンを追加する一意制約または一意索引の名前。一意索引の名前が関連する一意制約の名前と異なる場合は、一意索引の名前を使用します。一意性競合解消ルーチンのみ制約名を必要とします。
sequence_no	指定した競合解消方法を適用する順序。
method	作成する競合解消ルーチンの型。アドバンスト・レプリケーションで提供されている標準ルーチンのいずれかの名前を指定できます。または、独自のルーチンを作成している場合は、USER FUNCTION を選択し、FUNCTION_NAME 引数にそのルーチン名を指定してください。このリリースでサポートされている方法は、MINIMUM、MAXIMUM、LATEST TIMESTAMP、EARLIEST TIMESTAMP、ADDITIVE、AVERAGE、PRIORITY GROUP、SITE PRIORITY、OVERWRITE および DISCARD（更新競合の場合）、および APPEND SITE NAME、APPEND SEQUENCE および DISCARD（一意性競合の場合）です。削除競合の標準ルーチンは提供されていません。

表 37-11 ADD_conflicttype_RESOLUTION プロシージャのパラメータ

パラメータ	説明
parameter_column_name	<p>競合の解消に使用する列の名前。標準の方法では、単一の列で操作が行われます。たとえば、列グループに対して LATEST_TIMESTAMP 方法を使用している場合は、この引数としてタイムスタンプを含んだ列の名前を渡す必要があります。USER_FUNCTION を使用している場合は、任意の数の列を使用して競合を解消できます。</p> <p>この引数は、列名のカンマで区切られたリスト、または型 DBMS_REPCAT.VARCHAR2 の PL/SQL 表のいずれかを受け入れます。値 '*' は、競合の解消に表内のすべての列（または更新競合の場合は列グループ）を使用することを意味します。'*' を指定すると、列はユーザー・ファンクションにアルファベット順で渡されます。</p>
priority_group	<p>PRIORITY_GROUP または SITE_PRIORITY の更新競合解消方法を使用している場合は、自分で作成した優先グループの名前を指定する必要があります。</p> <p>『Oracle8i レプリケーション・ガイド』の「競合の解消」を参照してください。他の方法を使用している場合は、この引数のデフォルト値 NULL を使用できます。この引数は更新競合の場合のみ適用できます。</p>
function_name	<p>USER_FUNCTION 方法を選択した場合、または削除競合解消ルーチンを追加する場合は、自分で作成した競合解消ルーチンの名前を指定する必要があります。標準方法の 1 つを使用している場合は、この引数のデフォルト値 NULL を使用できます。</p>
comment	<p>DBA_REPRESOLUTION ビューに追加されるユーザー・コメント。</p>

例外

表 37-12 ADD_conflicttype_RESOLUTION プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
missingobject	指定したオブジェクトが、行レベル・レプリケーションを使用する表として指定のスキーマ内に存在していません。
missingschema	指定したスキーマが存在しません。
missingcolumn	PARAMETER_COLUMN_NAME 引数の一部としてユーザーが指定した列が存在しません。
missinggroup	指定した列グループが存在しません。
missingprioritygroup	ユーザーが指定した優先グループがその表に存在していません。
invalidmethod	ユーザーが指定した解消方法は認識されていません。
invalidparameter	PARAMETER_COLUMN_NAME 引数に指定した列の数が正しくありません。(標準ルーチンは列名を 1 つしか使用しません。)
missingfunction	ユーザーが指定したユーザー・ファンクションが存在しません。
missingconstraint	一意性競合に対してユーザーが指定した制約が存在しません。
notquiesced	指定した表が属しているオブジェクト・グループが停止中ではありません。
duplicateresolution	指定した競合解消方法は、すでに登録されています。
duplicatesequene	指定オブジェクトには指定した順序番号がすでに存在しています。
invalidprioritygroup	指定した優先グループが存在しません。
paramtype	型が、優先グループに割り当てられた型と異なります。

ALTER_MASTER_PROPAGATION プロシージャ

このプロシージャは、指定したマスター・サイトで指定したオブジェクト・グループの伝播方法を変更します。オブジェクト・グループは停止中にしておいてください。このプロシージャは、マスター定義サイトからコールする必要があります。マスターが dblink_list または dblink_table に含まれている場合、そのデータベース・リンクは ALTER_MASTER_PROPAGATION によって無視されます。マスターから同じマスター自身への伝播モードは変更できません。

構文

```
DBMS_REPCAT.ALTER_MASTER_PROPAGATION (  
    gname                IN   VARCHAR2,  
    master               IN   VARCHAR2,  
    { dblink_list        IN   VARCHAR2,  
      | dblink_table     IN   dbms_utility.dblink_array,}  
    propagation_mode     IN   VARCHAR2 := 'asynchronous',  
    comment              IN   VARCHAR2 := '' );
```

注意： このプロシージャはオーバーロードされています。dblink_list パラメータと dblink_table パラメータは、両方同時には指定できません。

パラメータ

表 37-13 ALTER_MASTER_PROPAGATION プロシージャのパラメータ

パラメータ	説明
gname	伝播モードを変更するオブジェクト・グループの名前。
master	伝播モードを変更するマスター・サイトの名前。
dblink_list	伝播を変更するデータベース・リンクのカンマで区切られたリスト。NULL の場合は、変更対象のマスター・サイト以外のすべてのマスターがデフォルトとして使用されます。
dblink_table	伝播を変更するデータベース・リンクの PL/SQL 表（位置 1 から索引付け）。
propagation_mode	指定したマスター・サイトの変更内容を、データベース・リンクのリストで指定したサイトに伝播する方法を指定します。指定できる値は SYNCHRONOUS と ASYNCHRONOUS です。
comment	DBA_REPPROP ビューに追加されるコメント。

例外

表 37-14 ALTER_MASTER_PROPAGATION プロシージャの例外

例外	説明
nonmasterdef	ローカル・サイトがマスター定義サイトではありません。
notquiesced	ローカル・サイトが停止中ではありません。
typefailure	指定した伝播モードは認識されませんでした。
nonmaster	データベース・リンクのリストに、マスター・サイト以外のサイトが含まれています。

ALTER_MASTER_REPOBJECT プロシージャ

このプロシージャは、レプリケート環境内のオブジェクトを変更します。このプロシージャは、マスター定義サイトからコールする必要があります。

構文

```
DBMS_REPCAT.ALTER_MASTER_REPOBJECT (  
  sname      IN   VARCHAR2,  
  oname      IN   VARCHAR2,  
  type       IN   VARCHAR2,  
  ddl_text   IN   VARCHAR2,  
  comment    IN   VARCHAR2      := '',  
  retry      IN   BOOLEAN       := FALSE);
```

パラメータ

表 37-15 ALTER_MASTER_REPOBJECT プロシージャのパラメータ

パラメータ	説明
sname	変更するオブジェクトを含んだスキーマ。
oname	変更するオブジェクトの名前。
type	変更するオブジェクトの型。サポートされている型は、TABLE、INDEX、SYNONYM、TRIGGER、VIEW、PROCEDURE、FUNCTION、PACKAGE および PACKAGE BODY です。
ddl_text	オブジェクトの変更に使用する DDL テキスト。この DDL は、適用前には解析されません。したがって、変更するオブジェクトに対する適切なスキーマ名とオブジェクト名を DDL テキストで使用していることを確認してください。 スキーマを指定しないで DDL を発行すると、レプリケーション管理者のスキーマがデフォルトのスキーマとなります。レプリケーション管理者のスキーマを使用しない場合は、スキーマを必ず指定してください。
comment	NULL 以外の場合、このコメントは DBA_REPOBJECT ビューの COMMENT フィールドに追加されます。
retry	retry が TRUE の場合は、オブジェクトの状態が VALID 以外のマスターにおいてのみ、オブジェクトが ALTER_MASTER_REPOBJECT によって変更されます。

例外

表 37-16 ALTER_MASTER_REPOBJECT プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
notquiesced	関連するオブジェクト・グループが中断されていません。
missingobject	SNAME と ONAME に該当するオブジェクトが存在しません。
typefailure	指定した型パラメータはサポートされていません。
ddlfailure	マスター定義サイトで DDL が成功しませんでした。
commfailure	アクセスできないマスター・サイトが少なくとも 1 つあります。

ALTER_PRIORITY プロシージャ

このプロシージャは、指定した優先グループ・メンバーに関連付けられている優先順位レベルを変更します。このプロシージャは、マスター定義サイトからコールする必要があります。

関連項目： 競合の解消方法の詳細は、『Oracle8i レプリケーション・ガイド』を参照してください。

構文

```
DBMS_REPCAT.ALTER_PRIORITY (  
    gname          IN   VARCHAR2,  
    pgroup         IN   VARCHAR2,  
    old_priority    IN   NUMBER,  
    new_priority    IN   NUMBER);
```

パラメータ

表 37-17 ALTER_PRIORITY プロシージャのパラメータ

パラメータ	説明
gname	優先グループが関連付けられているレプリケート・マスター・グループ
pgroup	変更する優先順位を含んだ優先グループの名前
old_priority	優先グループ・メンバーの現行の優先順位レベル
new_priority	優先グループ・メンバーに割り当てる新しい優先順位レベル

例外

表 37-18 ALTER_PRIORITY プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
duplicatepriority	新しい優先順位レベルが、優先グループ内にすでに存在しています。
missingrepgroup	指定したレプリケート・マスター・グループが存在しません。
missingvalue	値が、DBMS_REPCAT.ADD_PRIORITY_datatype のコールで登録されていません。
missingprioritygroup	指定した優先グループが存在しません。
notquiesced	指定したレプリケート・マスター・グループが停止中ではありません。

ALTER_PRIORITY_datatype プロシージャ

このプロシージャは、優先グループ内のメンバーの値を変更します。このプロシージャは、マスター定義サイトからコールする必要があります。コールする必要があるプロシージャは、priority 列のデータ型によって決まります。

関連項目： 競合の解消方法の詳細は、『Oracle8i レプリケーション・ガイド』を参照してください。

構文

```
DBMS_REPCAT.ALTER_PRIORITY_datatype (  
  gname          IN   VARCHAR2,  
  pgroup         IN   VARCHAR2,  
  old_value      IN   datatype,  
  new_value      IN   datatype);
```

datatype には、次のいずれかを指定します。

```
{ NUMBER  
| VARCHAR2  
| CHAR  
| DATE  
| RAW  
| NCHAR  
| NVARCHAR2 }
```


パラメータ

表 37-19 ALTER_PRIORITY_datatype プロシージャのパラメータ

パラメータ	説明
gname	優先グループが関連付けられているレプリケート・マスター・グループ
pgroup	変更する値を含んだ優先グループの名前
old_value	優先グループ・メンバーの現行の値
new_value	優先グループ・メンバーに割り当てる新規の値

例外

表 37-20 ALTER_PRIORITY_datatype プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
duplicatevalue	新しい値が、優先グループ内にすでに存在しています。
missingrepgroup	指定したレプリケート・マスター・グループが存在しません。
missingprioritygroup	指定した優先グループが存在しません。
missingvalue	元の値が存在しません。
paramtype	優先グループに対する新しい値のデータ型が正しくありません。
typefailure	優先グループに対して指定した値のデータ型が正しくありません。
notquiesced	指定したレプリケート・マスター・グループが停止中ではありません。

ALTER_SITE_PRIORITY プロシージャ

このプロシージャは、指定したサイトに関連付けられている優先順位レベルを変更します。
このプロシージャは、マスター定義サイトからコールする必要があります。

関連項目： 競合の解消方法の詳細は、『Oracle8i レプリケーション・ガイド』を参照してください。

構文

```
DBMS_REPCAT.ALTER_SITE_PRIORITY (  
    gname          IN   VARCHAR2,  
    name           IN   VARCHAR2,  
    old_priority   IN   NUMBER,  
    new_priority   IN   NUMBER);
```

パラメータ

表 37-21 ALTER_SITE_PRIORITY プロシージャのパラメータ

パラメータ	説明
gname	サイト優先グループが関連付けられているレプリケート・マスター・グループ。
name	メンバーを変更するサイト優先グループの名前。
old_priority	優先順位レベルを変更するサイトの現行の優先順位レベル。
new_priority	サイトの新しい優先順位レベル。数値が大きいかほど優先順位レベルは高くなります。

例外

表 37-22 ALTER_SITE_PRIORITY プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
missingrepgroup	指定したレプリケート・マスター・グループが存在しません。
missingpriority	元の優先順位レベルが、どのグループ・メンバーにも関連付けられていません。
duplicatepriority	新しい優先順位レベルが、グループ内の別のサイト用にすでに存在しています。
missingvalue	元の値が存在しません。
paramtype	優先グループに対する新しい値のデータ型が正しくありません。
notquiesced	レプリケート・マスター・グループが停止中ではありません。

ALTER_SITE_PRIORITY_SITE プロシージャ

このプロシージャは、指定した優先順位レベルに関連付けられているサイトを変更します。
このプロシージャは、マスター定義サイトからコールする必要があります。

関連項目： 競合の解消方法の詳細は、『Oracle8i レプリケーション・ガイド』を参照してください。

構文

```
DBMS_REPCAT.ALTER_SITE_PRIORITY_SITE (  
  gname      IN   VARCHAR2,  
  name       IN   VARCHAR2,  
  old_site   IN   VARCHAR2,  
  new_site   IN   VARCHAR2);
```

パラメータ

表 37-23 ALTER_SITE_PRIORITY_SITE プロシージャのパラメータ

パラメータ	説明
gname	サイト優先グループが関連付けられているレプリケート・マスター・グループ
name	メンバーを変更するサイト優先グループの名前
old_site	優先順位レベルから分離するサイトの現行のグローバル・データベース名
new_site	現行の優先順位レベルと関連付ける新しいグローバル・データベース名

例外

表 37-24 ALTER_SITE_PRIORITY_SITE プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
missingrepgroup	指定したレプリケート・マスター・グループが存在しません。
missingpriority	指定したサイト優先グループが存在しません。
missingvalue	旧サイトがグループ・メンバーではありません。
notquiesced	レプリケート・マスター・グループが停止中ではありません。

ALTER_SNAPSHOT_PROPAGATION プロシージャ

このプロシージャは、現行のスナップショット・サイトで指定したオブジェクト・グループの伝播方法を変更します。また、スナップショット・サイトで遅延トランザクション・キューを送信し、スナップショットのベース表をロックして、トリガーと関連するパッケージを再生成します。このプロシージャは、スナップショット・サイトからコールする必要があります。

構文

```
DBMS_REPCAT.ALTER_SNAPSHOT_PROPAGATION (  
  gname          IN  VARCHAR2,  
  propagation_mode IN  VARCHAR2,  
  comment        IN  VARCHAR2  := ''  
  gowner         IN  VARCHAR2  := 'PUBLIC');
```

パラメータ

表 37-25 ALTER_SNAPSHOT_PROPAGATION プロシージャのパラメータ

パラメータ	説明
gname	伝播モードを変更するオブジェクト・グループの名前。
propagation_mode	現行スナップショット・サイトの変更内容を、関連するマスター・サイトに伝播する方法。指定できる値は SYNCHRONOUS と ASYNCHRONOUS です。
comment	DBA_REPPROP ビューに追加されるコメント。
gowner	スナップショット・グループの所有者。

例外

表 37-26 ALTER_SNAPSHOT_PROPAGATION プロシージャの例外

例外	説明
missingrepgroup	指定したレプリケート・マスター・グループが存在しません。
typefailure	伝播モードの指定に誤りがあります。
nonsnapshot	現行のサイトが、指定したオブジェクト・グループのスナップショット・サイトではありません。
commfailure	マスターに接続できません。
notcompat	互換モードは 7.3.0.0 以上である必要があります。
failaltersnapprop	スナップショット・グループの伝播は、スナップショット・サイトを共有している同じマスターを持つ他のスナップショット・グループがない場合のみ変更できます。

CANCEL_STATISTICS プロシージャ

このプロシージャは、表の更新競合、一意性競合および削除競合の正常な解消に関する統計の収集を停止します。

構文

```
DBMS_REPCAT.CANCEL_STATISTICS (
    sname      IN   VARCHAR2,
    oname      IN   VARCHAR2);
```

パラメータ

表 37-27 CANCEL_STATISTICS プロシージャのパラメータ

パラメータ	説明
sname	表が置かれているスキーマの名前
oname	競合解消統計の収集を停止する表の名前

例外

表 37-28 CANCEL_STATISTICS プロシージャの例外

例外	説明
missingschema	指定のスキーマが存在しません。
missingobject	指定した表が存在しません。
statnotreg	現在、指定した表は統計収集用に登録されていません。

COMMENT_ON_COLUMN_GROUP プロシージャ

このプロシージャは、列グループの DBA_REPCOLUMN_GROUP ビューのコメント・フィールドを更新します。このコメントは、次回の DBMS_REPCAT.GENERATE_REPLICATION_SUPPORT コール後に、すべてのマスター・サイトで追加されます。

構文

```
DBMS_REPCAT.COMMENT_ON_COLUMN_GROUP (  
  sname          IN   VARCHAR2,  
  oname          IN   VARCHAR2,  
  column_group   IN   VARCHAR2,  
  comment        IN   VARCHAR2);
```

パラメータ

表 37-29 COMMENT_ON_COLUMN_GROUP プロシージャのパラメータ

パラメータ	説明
sname	オブジェクトが置かれているスキーマの名前
oname	列グループが関連付けられているレプリケート表の名前
column_group	列グループの名前
comment	DBA_REPCOLUMN_GROUP ビューの GROUP_COMMENT フィールドに含める更新したコメントのテキスト

例外

表 37-30 COMMENT_ON_COLUMN_GROUP プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
missinggroup	指定した列グループが存在しません。
missingobj	オブジェクトが見つかりません。

COMMENT_ON_PRIORITY_GROUP/COMMENT_ON_SITE_PRIORITY プロシージャ

COMMENT_ON_PRIORITY_GROUP は、優先グループの DBA_REPPRIORITY_GROUP ビューのコメント・フィールドを更新します。このコメントは、次回の GENERATE_REPLICATION_SUPPORT コール後に、すべてのマスター・サイトに追加されます。

COMMENT_ON_SITE_PRIORITY は、サイト優先グループの DBA_REPPRIORITY_GROUP ビューのコメント・フィールドを更新します。このプロシージャは、COMMENT_ON_COLUMN_GROUP プロシージャに対するラッパーで、便宜上の目的で提供されています。このプロシージャは、マスター定義サイトで発行する必要があります。

構文

```
DBMS_REPCAT.COMMENT_ON_PRIORITY_GROUP (
    gname      IN   VARCHAR2,
    pgroup     IN   VARCHAR2,
    comment    IN   VARCHAR2);
```

```
DBMS_REPCAT.COMMENT_ON_SITE_PRIORITY (
    gname      IN   VARCHAR2,
    name       IN   VARCHAR2,
```

```
comment IN VARCHAR2);
```

パラメータ

表 37-31 COMMENT_ON_PRIORITY_GROUP および COMMENT_ON_SITE_PRIORITY プロシージャのパラメータ

パラメータ	説明
gname	レプリケート・マスター・グループの名前
pgroup/name	優先グループまたはサイト優先グループの名前
comment	DBA_REPPRIORITY_GROUP ビューの PRIORITY_COMMENT フィールドに含める更新したコメントのテキスト

例外

表 37-32 COMMENT_ON_PRIORITY_GROUP および COMMENT_ON_SITE_PRIORITY プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
missingrepgroup	指定したレプリケート・マスター・グループが存在しません。
missingprioritygroup	指定した優先グループが存在しません。

COMMENT_ON_REPGROUP プロシージャ

このプロシージャは、レプリケート・マスター・グループの DBA_REPGROUP ビューのコメント・フィールドを更新します。このプロシージャは、マスター定義サイトで発行する必要があります。

構文

```
DBMS_REPCAT.COMMENT_ON_REPGROUP (  
  gname IN VARCHAR2,  
  comment IN VARCHAR2);
```


パラメータ

表 37-33 COMMENT_ON_REPGROUP プロシージャのパラメータ

パラメータ	説明
gname	コメントを記述するオブジェクト・グループの名前
comment	DBA_REPGROUP ビューの SCHEMA_COMMENT フィールドに含める更新したコメント

例外

表 37-34 COMMENT_ON_REPGROUP プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
commfailure	アクセスできないマスター・サイトが少なくとも 1 つあります。

COMMENT_ON_REPOBJECT プロシージャ

このプロシージャは、マスター・グループにあるレプリケート・オブジェクトの DBA_REPOBJECT ビューのコメント・フィールドを更新します。このプロシージャは、マスター定義サイトで発行する必要があります。

構文

```
DBMS_REPCAT.COMMENT_ON_REPOBJECT (  
    sname      IN   VARCHAR2,  
    oname      IN   VARCHAR2,  
    type       IN   VARCHAR2,  
    comment    IN   VARCHAR2);
```

パラメータ

表 37-35 COMMENT_ON_REOBJECT プロシージャのパラメータ

パラメータ	説明
sname	オブジェクトが置かれているスキーマの名前。
oname	コメントを記述するオブジェクトの名前。
type	オブジェクトの型。サポートされている型は、TABLE、INDEX、SYNONYM、TRIGGER、VIEW、PROCEDURE、FUNCTION、PACKAGE および PACKAGE BODY です。
comment	DBA_REOBJECT ビューの OBJECT_COMMENT フィールドに含める更新したコメントのテキスト。

例外

表 37-36 COMMENT_ON_REOBJECT プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
missingobject	指定したオブジェクトが存在しません。
typefailure	指定した型パラメータはサポートされていません。
commfailure	アクセスできないマスター・サイトが少なくとも 1 つあります。

COMMENT_ON_REPSITES プロシージャ

オブジェクト・グループがマスター・グループの場合、このプロシージャはマスター・サイトの DBA_REPSITES ビューの MASTER_COMMENT フィールドを更新します。オブジェクト・グループがスナップショット・グループの場合、このプロシージャはスナップショット・サイトの DBA_REPGROUP ビューの SCHEMA_COMMENT フィールドを更新します。

このプロシージャは、マスター・サイトまたはスナップショット・サイトで実行できます。スナップショット・サイトでこのプロシージャを実行する場合、スナップショット・グループ所有者は PUBLIC である必要があります。

関連項目： スナップショット・グループ所有者が PUBLIC でない場合に、スナップショット・サイトの DBA_REPGROUP ビューの SCHEMA_COMMENT フィールドにコメントを記入する方法は、37-32 ページの「[COMMENT_ON_SNAPSHOT_REPSITES プロシージャ](#)」を参照してください。

構文

```
DBMS_REPCAT.COMMENT_ON_REPSITES (  
    gname          IN    VARCHAR2,  
    [ master       IN    VARCHAR2,]  
    comment        IN    VARCHAR2);
```

パラメータ

表 37-37 COMMENT_ON_REPSITES プロシージャのパラメータ

パラメータ	説明
gname	オブジェクト・グループの名前。データベースが複数のレプリケート環境のマスター・サイトである場合は、このパラメータによって混乱を避けることができます。
master	(オプション) コメントを記述するマスター・サイトの完全修飾データベース名。マスター・サイトでこのプロシージャを実行するときは、このパラメータが必要です。スナップショット・サイトのコメントを更新するときは、このパラメータを省略します。
comment	適切なディクショナリ・ビューのコメント・フィールドに含める更新したコメントのテキスト。サイトがマスター・サイトの場合、このプロシージャは DBA_REPSITES ビューの MASTER_COMMENT フィールドを更新します。サイトがスナップショット・サイトの場合、このプロシージャは DBA_REPGROUP ビューの SCHEMA_COMMENT フィールドを更新します。

例外

表 37-38 COMMENT_ON_REPSITES プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
nonmaster	起動サイトがマスター・サイトではありません。
commfailure	アクセスできないマスター・サイトが少なくとも 1 つあります。
missingrepgroup	オブジェクト・グループが存在しません。
commfailure	1 つ以上のマスター・サイトがアクセス不可です。
corrupt	レプリケーション・カタログ・ビューに一貫性がありません。

COMMENT_ON_SNAPSHOT_REPSITES プロシージャ

このプロシージャは、指定したスナップショット・グループの DBA_REPGROUP データ・ディクショナリ・ビューの SCHEMA_COMMENT フィールドを更新します。グループ名は、レプリケート・スナップショット・グループにローカルで登録してください。このプロシージャは、スナップショット・サイトで実行する必要があります。

構文

```
DBMS_REPCAT.COMMENT_ON_SNAPSHOT_REPSITES (  
  gowner      IN  VARCHAR2,  
  gname       IN  VARCHAR2,  
  comment     IN  VARCHAR2);
```

パラメータ

表 37-39 COMMENT_ON_SNAPSHOT_REPSITES プロシージャのパラメータ

パラメータ	説明
gowner	スナップショット・オブジェクト・グループの所有者
gname	スナップショット・オブジェクト・グループの名前
comment	DBA_REPGROUP ビューの SCHEMA_COMMENT フィールドに含める更新したコメント

表 37-40 COMMENT_ON_SNAPSHOT_REPSITES プロシージャの例外

パラメータ	説明
missingrepgroup	スナップショット・オブジェクト・グループが存在しません。
nonsnapshot	接続サイトがスナップショット・サイトではありません。

COMMENT_ON_conflicttype_RESOLUTION プロシージャ

このプロシージャは、競合解消ルーチンの DBA_REPRESOLUTION ビューのコメント・フィールドを更新します。コールする必要があるプロシージャは、ルーチンが解消する競合の型によって決まります。このプロシージャは、マスター定義サイトで発行する必要があります。

表 37-41 COMMENT_ON_conflicttype_RESOLUTION プロシージャ

競合の型	プロシージャ名
update	COMMENT_ON_UPDATE_RESOLUTION
uniqueness	COMMENT_ON_UNIQUE_RESOLUTION
delete	COMMENT_ON_DELETE_RESOLUTION

コメントは、次の GENERATE_REPLICATION_SUPPORT コール後に、すべてのマスター・サイトに追加されます。

構文

```
DBMS_REPCAT.COMMENT_ON_UPDATE_RESOLUTION (
    sname          IN   VARCHAR2,
    oname          IN   VARCHAR2,
    column_group   IN   VARCHAR2,
    sequence_no    IN   NUMBER,
    comment        IN   VARCHAR2);
```

```
DBMS_REPCAT.COMMENT_ON_UNIQUE_RESOLUTION (
    sname          IN   VARCHAR2,
    oname          IN   VARCHAR2,
    constraint_name IN   VARCHAR2,
    sequence_no    IN   NUMBER,
    comment        IN   VARCHAR2);
```

```
DBMS_REPCAT.COMMENT_ON_DELETE_RESOLUTION (
  sname          IN   VARCHAR2,
  oname          IN   VARCHAR2,
  sequence_no    IN   NUMBER,
  comment        IN   VARCHAR2);
```

パラメータ

表 37-42 COMMENT_ON_conflicttype_RESOLUTION プロシージャのパラメータ

パラメータ	説明
sname	スキーマの名前
oname	競合解消ルーチンが関連付けられているレプリケート表の名前
column_group	更新競合解消ルーチンが関連付けられている列グループの名前
constraint_name	一意性競合解消ルーチンが関連付けられている一意制約の名前
sequence_no	競合解消プロシージャの順序番号
comment	DBA_REPRESOLUTION ビューの RESOLUTION_COMMENT フィールドに含める更新したコメントのテキスト

例外

表 37-43 COMMENT_ON_conflicttype_RESOLUTION プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
missingobject	指定したオブジェクトが存在しません。
missingresolution	指定した競合解消ルーチンが登録されていません。

COMPARE_OLD_VALUES プロシージャ

このプロシージャによって、更新と削除のために、レプリケート表の非キー列に対して、各マスター・サイトの元の値を比較できます。デフォルトでは、すべての列の元の値が比較されます。DBMS_REPCAT.COMPARE_OLD_VALUES をマスター定義サイトで起動すると、すべてのマスター・サイトとスナップショット・サイトでこの動作を変更できます。

構文

```
DBMS_REPCAT.COMPARE_OLD_VALUES (
    sname          IN  VARCHAR2,
    oname          IN  VARCHAR2,
    { column_list  IN  VARCHAR2,
      | column_table IN DBMS_REPCAT.VARCHAR2s, }
    operation      IN  VARCHAR2 := 'UPDATE',
    compare        IN  BOOLEAN := TRUE );
```

注意： このプロシージャはオーバーロードされています。column_list パラメータと column_table パラメータは、両方同時には指定できません。

パラメータ

表 37-44 COMPARE_OLD_VALUES プロシージャのパラメータ

パラメータ	説明
sname	表が置かれているスキーマ。
oname	レプリケート表の名前。
column_list	表内の列のカンマで区切られたリスト。エントリ間に空白を挿入しないでください。
column_table	リストのかわりに、列名を含んだ DBMS_REPCAT.VARCHAR2 型の PL/SQL 表を使用できます。最初の列名は位置 1、2 番目は位置 2、以下同様に設定されている必要があります。
operation	有効な値は、UPDATE、DELETE またはアスタリスクのワイルドカード '*'（更新と削除を意味します）です。

表 37-44 COMPARE_OLD_VALUES プロシージャのパラメータ

パラメータ	説明
compare	TRUE の場合は、指定した列の元の値が送信時に比較されます。 FALSE の場合、指定した列の元の値は送信時に比較されません。指定外の列と指定外の操作には影響しません。min_communication が表に対して TRUE になると、マスター定義サイトでは指定した変更がすぐに有効となります。変更内容がマスター・サイトまたはスナップショット・サイトで有効になるのは、次の回に、そのサイトで min_communication が TRUE の状態でレプリケーション・サポートが生成されたときです。

注意： operation パラメータを使用すると、行の削除時または非キー列の更新時に、非キー列の元の値を送信するかどうかを決定できます。元の値を送信しないと、元の値のかわりに NULL が送信され、更新または削除の適用時に送信先の現行の列値と元の値が等しいとみなされます。

Oracle のデフォルト動作を変更する前に、データ伝播の最小化について『Oracle8i レプリケーション・ガイド』を参照してください。

例外

表 37-45 COMPARE_OLD_VALUES プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
missingobject	指定したオブジェクトが、行レベル・レプリケーション情報を待機している表として指定のスキーマ内に存在していません。
missingcolumn	表に存在しない列が少なくとも 1 つあります。
notquiesced	レプリケート・マスター・グループが停止中ではありません。
typefailure	無効な操作が指定されています。

CREATE_MASTER_REPGROUP プロシージャ

このプロシージャは、新規で、空で、停止中のマスター・レプリケーション・オブジェクト・グループを作成します。

構文

```
DBMS_REPCAT.CREATE_MASTER_REPGROUP (
    gname          IN    VARCHAR2,
    group_comment   IN    VARCHAR2    := '',
    master_comment  IN    VARCHAR2    := '',
    qualifier       IN    VARCHAR2    := '');
```

パラメータ

表 37-46 CREATE_MASTER_REPGROUP プロシージャのパラメータ

パラメータ	説明
gname	作成するオブジェクト・グループの名前。
group_comment	DBA_REPGROUP ビューに追加されるコメント。
master_comment	DBA_REPSITES ビューに追加されるコメント。
qualifier	オブジェクト・グループの接続修飾子。必ず @ 符号を使用してください。接続修飾子は、『Oracle8i レプリケーション・ガイド』を参照してください。

例外

表 37-47 CREATE_MASTER_REPGROUP プロシージャの例外

例外	説明
duplicaterepgroup	オブジェクト・グループはすでに存在しています。
norepopt	アドバンスト・レプリケーション・オプションがインストールされていません。
missingrepgroup	オブジェクト・グループ名が指定されていません。
qualifiertoolong	接続修飾子が長すぎます。

CREATE_MASTER_REPOBJECT プロシージャ

このプロシージャは、オブジェクトをレプリケート・オブジェクトにします。

クラスタ化表のレプリケーションはサポートされていますが、`use_existing_object` パラメータをクラスタ化表に `FALSE` で設定することはできません。つまり、クラスタ化表はマスター・グループに関係している全マスター・サイトで事前に作成されている必要があります。ただし、事前に作成する表に表データは必要ありません。このようにして、クラスタ化表に対して `copy_rows` パラメータを `TRUE` に設定できます。

構文

```
DBMS_REPCAT.CREATE_MASTER_REPOBJECT (  
    sname                IN    VARCHAR2,  
    oname                IN    VARCHAR2,  
    type                 IN    VARCHAR2,  
    use_existing_object  IN    BOOLEAN    := TRUE,  
    ddl_text             IN    VARCHAR2    := NULL,  
    comment              IN    VARCHAR2    := '',  
    retry                IN    BOOLEAN    := FALSE,  
    copy_rows            IN    BOOLEAN    := TRUE,  
    gname                IN    VARCHAR2    := '');
```

パラメータ

このプロシージャのパラメータは、次のとおりです。

表 37-48 CREATE_MASTER_REPOBJECT プロシージャのパラメータ

パラメータ	説明
sname	レプリケートするオブジェクトが置かれているスキーマの名前。
oname	レプリケートするオブジェクトの名前。DDL_TEXT が NULL の場合は、指定したスキーマ内にこのオブジェクトがすでに存在している必要があります。一意性を確保するために、表名は 27 バイト以内、パッケージ名は 24 バイト以内で指定してください。
type	レプリケートするオブジェクトの型。サポートされている型は、TABLE、INDEX、SYNONYM、TRIGGER、VIEW、PROCEDURE、FUNCTION、PACKAGE および PACKAGE BODY です。
use_existing_object	<p>現行マスター・サイトの同じ型、同じ形式のオブジェクトを再使用する場合は、TRUE を指定します。詳細は、表 37-50 を参照してください。</p> <p>注意： クラスタ化表に対してこのパラメータを TRUE に設定する必要があります。</p>
ddl_text	<p>オブジェクトがマスター定義サイトにまだ存在していない場合は、このオブジェクトの作成に必要な DDL テキストを指定します。PL/SQL パッケージ、パッケージ本体、プロシージャおよびファクションの後には、セミコロンを付ける必要があります。SQL 文には、セミコロンを付ける必要はありません。この DDL は、適用前には解析されません。したがって、作成するオブジェクトに対応する適切なスキーマ名とオブジェクト名を DDL テキストで使用していることを確認してください。</p> <p>スキーマ (sname パラメータ) を指定しないで DDL を発行すると、レプリケーション管理者のスキーマがデフォルトのスキーマとして使用されます。レプリケーション管理者のスキーマを使用しない場合は、スキーマを必ず指定してください。</p>
comment	DBA_REPOBJECT ビューの OBJECT_COMMENT フィールドに追加されるコメント。
retry	以前にオブジェクトを作成できなかった場合に、そのオブジェクトの作成を再試行する場合は、TRUE を指定します。このパラメータは、エラーが一時的な場合やすでに修正されている場合 (リソースが不十分などのエラーの場合) に使用します。TRUE の場合、オブジェクトは、オブジェクトの状態が VALID 以外のマスター・サイトでのみ作成されます。
copy_rows	新規レプリケート・オブジェクトの初期の内容を、マスター定義サイトのオブジェクトの内容と一致させる場合は TRUE を指定します。詳細は、 表 37-50 を参照してください。
gname	レプリケート・オブジェクトを作成するオブジェクト・グループの名前。指定しないと、デフォルトのオブジェクト・グループ名としてスキーマ名が使用されます。

表 37-49 CREATE_MASTER_REPOBJECT プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
notquiesced	レプリケート・マスター・グループが中断されていません。
duplicateobject	指定したオブジェクトがレプリケート・マスター・グループにすでに存在しており、retry が FALSE であるか、または名前の競合が発生しています。
missingobject	sname と oname に該当するオブジェクトが存在せず、適切な DDL も指定されていません。
typefailure	指定した型のオブジェクトはレプリケートできません。
ddlfailure	マスター定義サイトで DDL が成功しませんでした。
commfailure	アクセスできないマスター・サイトが少なくとも 1 つあります。
notcompat	最低でも 7.3 互換モードではないリモート・マスターがあります。

オブジェクト作成

表 37-50 マスター・サイトにおけるオブジェクト作成

オブジェクトはすでに存在？	COPY_ROWS	USE_EXISTING_OBJECTS	結果
はい	TRUE	TRUE	オブジェクトが一致しない場合は duplicatedobject メッセージが戻されます。表の場合は、マスター定義サイトのデータが使用されます。
はい	FALSE	TRUE	オブジェクトが一致しない場合は duplicatedobject メッセージが戻されます。表の場合、DBA は内容が同じであることを確認する必要があります。
はい	TRUE/FALSE	FALSE	duplicatedobject メッセージが戻されます。
いいえ	TRUE	TRUE/FALSE	オブジェクトが作成されます。表には、マスター定義サイトのデータが移入されます。
いいえ	FALSE	TRUE/FALSE	オブジェクトが作成されます。DBA は表にデータを移入し、すべてのサイトで表の内容が一致していることを確認する必要があります。

CREATE_SNAPSHOT_REPGROUP プロシージャ

このプロシージャは、空のスナップショット・グループをローカル・データベース内に新規に作成します。CREATE_SNAPSHOT_REPGROUP は、REGISTER_SNAPSHOT_REPGROUP を自動的にコールします。ただし、登録中に発生したエラーは無視します。

構文

```
DBMS_REPCAT.CREATE_SNAPSHOT_REPGROUP (
    gname          IN   VARCHAR2,
    master         IN   VARCHAR2,
    comment        IN   VARCHAR2      := '',
    propagation_mode IN VARCHAR2      := 'ASYNCHRONOUS',
    fname         IN   VARCHAR2      := NULL
    gowner        IN   VARCHAR2      := 'PUBLIC');
```

パラメータ

表 37-51 CREATE_SNAPSHOT_REPGROUP プロシージャのパラメータ

パラメータ	説明
gname	レプリケート・マスター・グループの名前。このオブジェクト・グループは、指定したマスター・サイトに存在している必要があります。
master	レプリケート環境でマスターとして使用するデータベースの完全修飾データベース名。必要に応じて接続修飾子を指定できます。接続修飾子の使用法は、『Oracle8i レプリケーション・ガイド』および『Oracle8i 分散システム』を参照してください。
comment	DBA_REPGROUP ビューに追加されるコメント。
propagation_mode	オブジェクト・グループ内のすべての更新可能スナップショットに使用する伝播方法。指定できる値は SYNCHRONOUS と ASYNCHRONOUS です。
fname	内部使用のためのパラメータ。オラクル社カスタマ・サポート・センターから指示がない限り、このパラメータは設定しないでください。
gowner	スナップショット・グループの所有者。

例外

表 37-52 CREATE_SNAPSHOT_REPGROUP プロシージャの例外

例外	説明
duplicaterepgroup	オブジェクト・グループが、起動サイトにすでに存在しています。
nonmaster	指定したデータベースはマスター・サイトではありません。
commfailure	指定したデータベースにアクセスできません。
norepopt	アドバンスド・レプリケーション・オプションがインストールされていません。
typefailure	伝播モードの指定に誤りがあります。
missingrepgroup	レプリケート・マスター・グループがマスター・サイトに存在していません。
invalidqualifier	マスターに指定した接続修飾子がオブジェクト・グループに対して無効です。
alreadymastered	ローカル・サイトに、同じグループ名を持つ別のスナップショット・グループがありますが、マスター・サイトが異なります。

CREATE_SNAPSHOT_REPOBJECT プロシージャ

このプロシージャは、スナップショット・サイトにレプリケート・オブジェクトを追加します。

構文

```
DBMS_REPCAT.CREATE_SNAPSHOT_REPOBJECT (
  sname          IN   VARCHAR2,
  oname          IN   VARCHAR2,
  type           IN   VARCHAR2,
  ddl_text       IN   VARCHAR2 := '',
  comment        IN   VARCHAR2 := '',
  gname          IN   VARCHAR2 := '',
  gen_objs_owner IN   VARCHAR2 := '',
  min_communication IN  BOOLEAN := TRUE ,
  generate_80_compatible IN  BOOLEAN := TRUE
  gowner         IN   VARCHAR2 := 'PUBLIC');
```

パラメータ

表 37-53 CREATE_SNAPSHOT_REPOBJECT プロシージャのパラメータ

パラメータ	説明
sname	オブジェクトが置かれているスキーマの名前。
oname	レプリケート・スナップショット・オブジェクト・グループに追加するオブジェクトの名前。ONAME が関連するマスター・サイトに存在している必要があります。
type	レプリケートするオブジェクトの型。スナップショット・サイトでサポートされる型は、PACKAGE、PACKAGE BODY、PROCEDURE、FUNCTION、SNAPSHOT、SYNONYM、TRIGGER および VIEW です。
ddl_text	<p>SNAPSHOT 型のオブジェクトの場合は、DDL がオブジェクトの作成に必要です。その他の型の場合は、デフォルトの '' (空文字列) を使用します。</p> <p>同じ名前のスナップショットがすでに存在している場合、DDL は無視され、既存のスナップショットがレプリケート・オブジェクトとして登録されます。スナップショットのマスター表が、このスキーマに対して指定したマスター・サイトのレプリケート・マスター・グループ内に存在しない場合は、missingobject エラーが発生します。</p> <p>スキーマを指定せずに DDL を発行すると、レプリケーション管理者のスキーマがデフォルトのスキーマとして使用されます。レプリケーション管理者のスキーマを使用しない場合は、スキーマを必ず指定してください。</p>
comment	DBA_REPOBJECT ビューの OBJECT_COMMENT フィールドに追加されるコメント。
gname	オブジェクトを追加するレプリケート・マスター・グループの名前。指定しないと、デフォルトのグループ名としてスキーマ名が使用されます。
gen_objs_owner	トランザクションの所有者として割り当てるユーザーの名前。
min_communication	マスター・サイトのいずれかが Oracle7 リリース 7.3 を実行している場合は、FALSE を設定します。新旧の値の伝播を最小化する場合は、TRUE を設定します。デフォルトは TRUE です。競合の解消方法の詳細は、『Oracle8i レプリケーション・ガイド』を参照してください。
generate_80_compatible	マスター・サイトのいずれかが Oracle8i リリース 8.1.5 以前のバージョンの Oracle Server を実行している場合は、TRUE を設定します。レプリケート環境が純粋な Oracle8i リリース 8.1.5 以上の環境の場合は、FALSE を設定します。
gowner	スナップショット・グループの所有者。

例外

表 37-54 CREATE_SNAPSHOT_REPOBJECT プロシージャの例外

例外	説明
nonsnapshot	起動サイトがスナップショット・サイトではありません。
nonmaster	このマスターは、マスター・サイトではありません。
missingobject	指定したオブジェクトがマスターのレプリケート・マスター・グループに存在していません。
duplicateobject	指定したオブジェクトは、すでに別の形式で存在しています。
typefailure	この型は許可されていません。
ddlfailure	DDL は成功しませんでした。
commfailure	マスター・サイトにアクセスできません。
missingschema	スキーマが、データベース・スキーマとして存在していません。
badsnapddl	DDL は実行されましたが、スナップショットが存在しません。
onlyonesnap	マスター表のスナップショットは、1 つしか作成できません。
badsnapname	スナップショットのベース表がマスター表と異なります。
missingrepgroup	レプリケート・マスター・グループが存在しません。

DEFINE_COLUMN_GROUP プロシージャ

このプロシージャは、空の列グループを作成します。このプロシージャは、マスター定義サイトからコールする必要があります。

関連項目： 競合の解消方法の詳細は、『Oracle8i レプリケーション・ガイド』を参照してください。

構文

```
DBMS_REPCAT.DEFINE_COLUMN_GROUP (  
  sname          IN   VARCHAR2,  
  oname          IN   VARCHAR2,  
  column_group   IN   VARCHAR2,  
  comment        IN   VARCHAR2 := NULL);
```


パラメータ

表 37-55 DEFINE_COLUMN_GROUP プロシージャのパラメータ

パラメータ	説明
sname	レプリケート表が置かれているスキーマ
oname	列グループを作成するレプリケート表の名前
column_group	作成する列グループの名前
comment	DBA_REPCOLUMN_GROUP ビューに表示されるユーザー・テキスト

例外

表 37-56 DEFINE_COLUMN_GROUP プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
missingobject	指定した表が存在しません。
duplicategroup	指定した列グループが、表にすでに存在しています。
notquiesced	指定した表が属しているオブジェクト・グループが停止中ではありません。

DEFINE_PRIORITY_GROUP プロシージャ

このプロシージャは、レプリケート・マスター・グループに新規の優先グループを作成します。このプロシージャは、マスター定義サイトからコールする必要があります。

関連項目： 競合の解消方法の詳細は、『Oracle8i レプリケーション・ガイド』を参照してください。

構文

```
DBMS_REPCAT.DEFINE_PRIORITY_GROUP (
  gname          IN   VARCHAR2,
  pgroup         IN   VARCHAR2,
  datatype       IN   VARCHAR2,
  fixed_length   IN   INTEGER := NULL,
  comment        IN   VARCHAR2 := NULL);
```

パラメータ

表 37-57 DEFINE_PRIORITY_GROUP プロシージャのパラメータ

パラメータ	説明
gname	優先グループを作成するレプリケート・マスター・グループ。
pgroup	作成する優先グループの名前。
datatype	優先グループ・メンバーのデータ型。サポートされているデータ型は、CHAR、VARCHAR2、NUMBER、DATE、RAW、NCHAR および NVARCHAR2 です。
fixed_length	CHAR データ型の場合は、列の長さを指定してください。その他のすべての型では、デフォルトの NULL を使用できます。
comment	DBA_REPPRIORITY ビューに追加されるユーザー・コメント。

例外

表 37-58 DEFINE_PRIORITY_GROUP プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
missingrepgroup	指定したレプリケート・マスター・グループが存在しません。
duplicateprioritygroup	指定した優先グループは、レプリケート・マスター・グループ内にすでに存在しています。
typefailure	指定したデータ型はサポートされていません。
notquiesced	レプリケート・マスター・グループが停止中ではありません。

DEFINE_SITE_PRIORITY プロシージャ

このプロシージャは、レプリケート・マスター・グループに新規のサイト優先グループを作成します。このプロシージャは、マスター定義サイトからコールする必要があります。

関連項目： 競合の解消方法の詳細は、『Oracle8i レプリケーション・ガイド』を参照してください。

構文

```
DBMS_REPCAT.DEFINE_SITE_PRIORITY (  
    gname          IN   VARCHAR2,  
    name           IN   VARCHAR2,  
    comment        IN   VARCHAR2 := NULL);
```

パラメータ

表 37-59 DEFINE_SITE_PRIORITY プロシージャのパラメータ

パラメータ	説明
gname	サイト優先グループを作成するレプリケート・マスター・グループ
name	作成するサイト優先グループの名前
comment	DBA_REPPRIORITY ビューに追加されるユーザー・コメント

例外

表 37-60 DEFINE_SITE_PRIORITY プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
missingrepgroup	指定したレプリケート・マスター・グループが存在しません。
duplicate prioritygroup	指定したサイト優先グループは、レプリケート・マスター・グループ内にすでに存在しています。
notquiesced	レプリケート・マスター・グループが停止中ではありません。

DO_DEFERRED_REPCAT_ADMIN プロシージャ

このプロシージャは、現行のマスター・サイトで指定したレプリケート・マスター・グループまたはすべてのマスター・サイト（ジョブ・キューを利用）に対して未処理のローカル遅延管理手順を実行します。

DO_DEFERRED_REPCAT_ADMIN は、DO_DEFERRED_REPCAT_ADMIN をコールした接続ユーザーが送信した管理要求のみ実行します。その他のユーザーが送信した要求は無視されます。

構文

```
DBMS_REPCAT.DO_DEFERRED_REPCAT_ADMIN (  
    gname           IN   VARCHAR2,  
    all_sites       IN   BOOLEAN := FALSE);
```

パラメータ

表 37-61 DO_DEFERRED_REPCAT_ADMIN プロシージャのパラメータ

パラメータ	説明
gname	レプリケート・マスター・グループの名前。
all_sites	TRUE の場合は、ジョブを使用して各マスター・サイトごとにローカル管理手順を実行します。

例外

表 37-62 DO_DEFERRED_REPCAT_ADMIN プロシージャの例外

例外	説明
nonmaster	起動サイトがマスター・サイトではありません。
commfailure	all_sites が TRUE ですが、アクセスできないマスター・サイトが少なくとも 1 つあります。

DROP_COLUMN_GROUP プロシージャ

このプロシージャは列グループを削除します。このプロシージャは、マスター定義サイトからコールする必要があります。

関連項目： 競合の解消方法の詳細は、『Oracle8i レプリケーション・ガイド』を参照してください。

構文

```
DBMS_REPCAT.DROP_COLUMN_GROUP (
    sname          IN   VARCHAR2,
    oname          IN   VARCHAR2,
    column_group IN   VARCHAR2);
```

パラメータ

表 37-63 DROP_COLUMN_GROUP プロシージャのパラメータ

パラメータ	説明
sname	レプリケート表が置かれているスキーマ
oname	列グループを削除するレプリケート表の名前
column_group	削除する列グループの名前

例外

表 37-64 DROP_COLUMN_GROUP プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
referenced	指定した列グループは、競合の検出と解消で使用されています。
missingobject	指定した表が存在しません。
missinggroup	指定した列グループが存在しません。
notquiesced	表が属しているレプリケート・マスター・グループが停止中ではありません。

DROP_GROUPED_COLUMN プロシージャ

このプロシージャは、列グループからメンバーを削除します。このプロシージャは、マスター定義サイトからコールする必要があります。

関連項目： 競合の解消方法の詳細は、『Oracle8i レプリケーション・ガイド』を参照してください。

構文

```
DBMS_REPCAT.DROP_GROUPED_COLUMN (  
    sname                IN    VARCHAR2,  
    oname                IN    VARCHAR2,  
    column_group         IN    VARCHAR2,  
    list_of_column_names IN    VARCHAR2 | DBMS_REPCAT.VARCHAR2s);
```

パラメータ

表 37-65 DROP_GROUPED_COLUMN プロシージャのパラメータ

パラメータ	説明
sname	レプリケート表が置かれているスキーマ。
oname	列グループが置かれているレプリケート表の名前。
column_group	メンバーを削除する列グループの名前。
list_of_column_names	指定した列グループから削除する列の名前。列名は、名前のカンマで区切られたリストまたは PL/SQL 表のいずれかで示します。PL/SQL 表は、DBMS_REPCAT.VARCHAR2 型にしてください。

例外

表 37-66 DROP_GROUPED_COLUMN プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
missingobject	指定した表が存在しません。
notquiesced	表が属しているレプリケート・マスター・グループが停止中ではありません。

DROP_MASTER_REPGROUP プロシージャ

このプロシージャは、現行のサイトからレプリケート・マスター・グループを削除します。マスター定義サイトも含め、すべてのマスター・サイトからレプリケート・マスター・グループを削除するには、最後の引数に TRUE を設定して、このプロシージャをマスター定義サイトでコールします。

構文

```
DBMS_REPCAT.DROP_MASTER_REPGROUP (  
    gname           IN VARCHAR2,  
    drop_contents   IN BOOLEAN    := FALSE,  
    all_sites       IN BOOLEAN    := FALSE);
```

パラメータ

表 37-67 DROP_MASTER_REPGROUP プロシージャのパラメータ

パラメータ	説明
gname	現行のマスター・サイトから削除するレプリケート・マスター・グループの名前。
drop_contents	デフォルトでは、マスター・サイトのオブジェクト・グループを削除したとき、すべてのオブジェクトがデータベースに残ります。このオブジェクトは、この後レプリケートされなくなります。つまり、オブジェクト・グループ内のレプリケート・オブジェクトと他のマスター・サイトとの間で、変更内容の送受信が行われなくなります。このパラメータに TRUE を設定すると、レプリケート・マスター・グループ内のすべてのレプリケート・オブジェクトが、関連するスキーマから削除されます。
all_sites	このパラメータが TRUE で起動サイトがマスター定義サイトの場合、このプロシージャは、要求をすべてのマスターに同期式でマルチキャストします。この場合、マスター定義サイトでは要求がすぐに実行され、その他のマスター・サイトでは遅れて実行される可能性があります。

例外

表 37-68 DROP_MASTER_REPGROUP プロシージャの例外

例外	説明
nonmaster	起動サイトがマスター・サイトではありません。
nonmasterdef	all_sites が TRUE ですが、起動サイトがマスター定義サイトではありません。
commfailure	all_sites が TRUE ですが、アクセスできないマスター・サイトが少なくとも1つあります。
fullqueue	遅延 RPC キューに、レプリケート・マスター・グループに対するエントリがあります。
masternotremoved	all_sites が TRUE ですが、マスターでマスタ定義サイトが認識されていません。

DROP_MASTER_REPOBJECT プロシージャ

このプロシージャは、レプリケート・マスター・グループからレプリケート・オブジェクトを削除します。このプロシージャは、マスター定義サイトからコールする必要があります。

構文

```
DBMS_REPCAT.DROP_MASTER_REPOBJECT (  
  sname          IN   VARCHAR2,  
  oname          IN   VARCHAR2,  
  type           IN   VARCHAR2,  
  drop_objects   IN   BOOLEAN      := FALSE);
```


パラメータ

表 37-69 DROP_MASTER_REPOBJECT プロシージャのパラメータ

パラメータ	説明
sname	オブジェクトが置かれているスキーマの名前。
oname	レプリケート・マスター・グループから削除するオブジェクトの名前。
type	削除するオブジェクトの型。
drop_objects	デフォルトでは、オブジェクトはスキーマ内に残りますが、レプリケート・マスター・グループからは削除されます。つまり、このオブジェクトに対する変更は、他のマスター・サイトとスナップショット・サイトにレプリケートされなくなります。レプリケート環境からオブジェクトを完全に削除するには、このパラメータに TRUE を設定します。パラメータを TRUE に設定すると、各マスター・サイトでデータベースからオブジェクトが削除されます。

例外

表 37-70 DROP_MASTER_REPOBJECT プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
missingobject	指定したオブジェクトが存在しません。
typefailure	指定した型パラメータはサポートされていません。
commfailure	アクセスできないマスター・サイトが少なくとも 1 つあります。

DROP_PRIORITY プロシージャ

このプロシージャは、優先順位レベルに従って優先グループのメンバーを削除します。このプロシージャは、マスター定義サイトからコールする必要があります。

関連項目： 競合の解消方法の詳細は、『Oracle8i レプリケーション・ガイド』を参照してください。

構文

```
DBMS_REPCAT.DROP_PRIORITY(
    gname          IN   VARCHAR2,
    pgroup         IN   VARCHAR2,
    priority_num    IN   NUMBER);
```

パラメータ

表 37-71 DROP_PRIORITY プロシージャのパラメータ

パラメータ	説明
gname	優先グループが関連付けられているレプリケート・マスター・グループ
pgroup	削除するメンバーを含んだ優先グループの名前
priority_num	グループから削除する優先グループ・メンバーの優先順位レベル

例外

表 37-72 DROP_PRIORITY プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
missingrepgroup	指定したレプリケート・マスター・グループが存在しません。
missingprioritygroup	指定した優先グループが存在しません。
notquiesced	レプリケート・マスター・グループが停止中ではありません。

DROP_PRIORITY_GROUP プロシージャ

このプロシージャは、指定したレプリケート・マスター・グループの優先グループを削除します。このプロシージャは、マスター定義サイトからコールする必要があります。

関連項目： 競合の解消方法の詳細は、『Oracle8i レプリケーション・ガイド』を参照してください。

構文

```
DBMS_REPCAT.DROP_PRIORITY_GROUP (  
    gname      IN    VARCHAR2,  
    pgroup     IN    VARCHAR2);
```

パラメータ

表 37-73 DROP_PRIORITY_GROUP プロシージャのパラメータ

パラメータ	説明
gname	優先グループが関連付けられているレプリケート・マスター・グループ
pgroup	削除する優先グループの名前

例外

表 37-74 DROP_PRIORITY_GROUP プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
missingrepgroup	指定したレプリケート・マスター・グループが存在しません。
referenced	指定した優先グループは、競合の解消で使用されています。
notquiesced	指定したレプリケート・マスター・グループが停止中ではありません。

DROP_PRIORITY_datatype プロシージャ

このプロシージャは、値による優先グループのメンバーを削除します。このプロシージャは、マスター定義サイトからコールする必要があります。コールする必要があるプロシージャは、priority 列のデータ型によって決まります。

関連項目： 競合の解消方法の詳細は、『Oracle8i レプリケーション・ガイド』を参照してください。

構文

```
DBMS_REPCAT.DROP_PRIORITY_datatype (  
    gname      IN   VARCHAR2,  
    pgroup     IN   VARCHAR2,  
    value      IN   datatype);
```

datatype には、次のいずれかを指定します。

```
{ NUMBER  
| VARCHAR2  
| CHAR  
| DATE  
| RAW  
| NCHAR  
| NVARCHAR2 }
```

パラメータ

表 37-75 DROP_PRIORITY_datatype プロシージャのパラメータ

パラメータ	説明
gname	優先グループが関連付けられているレプリケート・マスター・グループ
pgroup	削除するメンバーを含んだ優先グループの名前
value	グループから削除する優先グループ・メンバーの値

例外

表 37-76 DROP_PRIORITY_datatype プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
missingrepgroup	指定したレプリケート・マスター・グループが存在しません。
missingprioritygroup	指定した優先グループが存在しません。
paramtype, typefailure	優先グループに対する値のデータ型が正しくありません。
notquiesced	指定したレプリケート・マスター・グループが停止中ではありません。

DROP_SITE_PRIORITY プロシージャ

このプロシージャは、指定したレプリケート・マスター・グループのサイト優先グループを削除します。このプロシージャは、マスター定義サイトからコールする必要があります。

関連項目： 競合の解消方法の詳細は、『Oracle8i レプリケーション・ガイド』を参照してください。

構文

```
DBMS_REPCAT.DROP_SITE_PRIORITY (
    gname      IN   VARCHAR2,
    name       IN   VARCHAR2);
```

パラメータ

表 37-77 DROP_SITE_PRIORITY プロシージャのパラメータ

パラメータ	説明
gname	サイト優先グループが関連付けられているレプリケート・マスター・グループ
name	削除するサイト優先グループの名前

例外

表 37-78 DROP_SITE_PRIORITY プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
missingrepgroup	指定したレプリケート・マスター・グループが存在しません。
referenced	指定したサイト優先グループは、競合の解消で使用されています。
notquiesced	指定したレプリケート・マスター・グループが停止中ではありません。

DROP_SITE_PRIORITY_SITE プロシージャ

このプロシージャは、サイト優先グループから、名前で指定したサイトを削除します。このプロシージャは、マスター定義サイトからコールする必要があります。

関連項目： 競合の解消方法の詳細は、『Oracle8i レプリケーション・ガイド』を参照してください。

構文

```
DBMS_REPCAT.DROP_SITE_PRIORITY_SITE (  
    gname      IN   VARCHAR2,  
    name       IN   VARCHAR2,  
    site       IN   VARCHAR2);
```

パラメータ

表 37-79 DROP_SITE_PRIORITY_SITE プロシージャのパラメータ

パラメータ	説明
gname	サイト優先グループが関連付けられているレプリケート・マスター・グループ
name	メンバーを削除するサイト優先グループの名前
site	グループから削除するサイトのグローバル・データベース名

例外

表 37-80 DROP_SITE_PRIORITY_SITE プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
missingrepgroup	指定したレプリケート・マスター・グループが存在しません。
missingpriority	指定したサイト優先グループが存在しません。
notquiesced	指定したレプリケート・マスター・グループが停止中ではありません。

DROP_SNAPSHOT_REPGROUP プロシージャ

このプロシージャは、レプリケート環境からスナップショット・サイトを削除します。
 DROP_SNAPSHOT_REPGROUP は、UNREGISTER_SNAPSHOT_REPGROUP を自動的にコールしてスナップショットの登録を解除します。ただし、登録解除中にエラーが発生しても無視します。

構文

```
DBMS_REPCAT.DROP_SNAPSHOT_REPGROUP (
    gname                IN   VARCHAR2,
    drop_contents        IN   BOOLEAN   := FALSE
    gowner               IN   VARCHAR2   := 'PUBLIC');
```

パラメータ

表 37-81 DROP_SNAPSHOT_REPGROUP プロシージャのパラメータ

パラメータ	説明
gname	現行のスナップショット・サイトから削除するレプリケート・マスター・グループの名前。トリガーやパッケージなど、レプリケーションをサポートするために生成されたすべてのオブジェクトが削除されます。
drop_contents	デフォルトでは、スナップショット・サイトのレプリケート・マスター・グループを削除すると、すべてのオブジェクトが関連するスキーマに残ります。このオブジェクトは、この後レプリケートされなくなります。このパラメータに TRUE を設定すると、レプリケート・マスター・グループ内のすべてのレプリケート・オブジェクトが、スキーマから削除されます。
gowner	スナップショット・グループの所有者。

例外

表 37-82 DROP_SNAPSHOT_REPGROUP プロシージャの例外

例外	説明
nonsnapshot	起動サイトがスナップショット・サイトではありません。
missingrepgroup	指定したオブジェクト・グループが存在しません。

DROP_SNAPSHOT_REPOBJECT プロシージャ

このプロシージャは、スナップショット・サイトからレプリケート・オブジェクトを削除します。

構文

```
DBMS_REPCAT.DROP_SNAPSHOT_REPOBJECT (  
    sname          IN   VARCHAR2,  
    oname          IN   VARCHAR2,  
    type           IN   VARCHAR2,  
    drop_objects   IN   BOOLEAN := FALSE);
```

パラメータ

表 37-83 DROP_SNAPSHOT_REPOBJECT プロシージャのパラメータ

パラメータ	説明
sname	オブジェクトが置かれているスキーマの名前。
oname	レプリケート・マスター・グループから削除するオブジェクトの名前。
type	削除するオブジェクトの型。
drop_objects	デフォルトでは、オブジェクトは関連スキーマ内に残りますが、その関連オブジェクト・グループからは削除されます。現行スナップショット・サイトのスキーマからオブジェクトを完全に削除するには、この引数に TRUE を設定します。パラメータを TRUE に設定すると、スナップショット・サイトでデータベースからオブジェクトが削除されます。

例外

表 37-84 DROP_SNAPSHOT_REPOBJECT プロシージャの例外

例外	説明
nonsnapshot	起動サイトがスナップショット・サイトではありません。
missingobject	指定したオブジェクトが存在しません。
typefailure	指定した型パラメータはサポートされていません。

DROP_conflictype_RESOLUTION プロシージャ

このプロシージャは、更新、削除または一意性の競合解消ルーチンを削除します。このプロシージャは、マスター定義サイトからコールする必要があります。コールする必要があるプロシージャは、ルーチンが解消する競合の型によって決まります。

競合の型

表 37-85 DROP_conflictype_RESOLUTION プロシージャの競合の型

競合の型	説明
update	DROP_UPDATE_RESOLUTION
uniqueness	DROP_UNIQUE_RESOLUTION
delete	DROP_DELETE_RESOLUTION

構文

```
DBMS_REPCAT.DROP_UPDATE_RESOLUTION (  
    sname          IN   VARCHAR2,  
    oname          IN   VARCHAR2,  
    column_group   IN   VARCHAR2,  
    sequence_no    IN   NUMBER);
```

```
DBMS_REPCAT.DROP_DELETE_RESOLUTION (  
    sname          IN   VARCHAR2,  
    oname          IN   VARCHAR2,  
    sequence_no    IN   NUMBER);
```

```
DBMS_REPCAT.DROP_UNIQUE_RESOLUTION (  
    sname          IN   VARCHAR2,  
    oname          IN   VARCHAR2,  
    constraint_name IN   VARCHAR2,  
    sequence_no    IN   NUMBER);
```

パラメータ

表 37-86 DROP_conflicttype_RESOLUTION プロシージャのパラメータ

パラメータ	説明
sname	表が置かれているスキーマ。
oname	競合解消ルーチンを削除する表の名前。
column_group	更新競合解消ルーチンを削除する列グループの名前。
constraint_name	一意性競合解消ルーチンを削除する一意性制約の名前。
sequence_no	削除する競合解消方法に割り当てられている順序番号。この番号でルーチンが一意に識別されます。

例外

表 37-87 DROP_conflicttype_RESOLUTION プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
missingobject	指定したオブジェクトが、指定のスキーマ内の表として存在していないか、または指定した順序番号の競合解消ルーチンが登録されていません。
notquiesced	レプリケート・マスター・グループが停止中ではありません。

EXECUTE_DDL プロシージャ

このプロシージャは、一部またはすべてのマスター・サイトで実行される DDL を提供します。このプロシージャは、マスター定義サイトからコールする必要があります。

構文

```
DBMS_REPCAT.EXECUTE_DDL (
  gname          IN   VARCHAR2,
  { master_list  IN   VARCHAR2      := NULL,
    | master_table IN   DBMS_UTILITY.DBLINK_ARRAY, }
  ddl_text       IN   VARCHAR2);
```

注意： このプロシージャはオーバーロードされています。master_list パラメータと master_table パラメータは、両方同時には指定できません。

パラメータ

表 37-88 EXECUTE_DDL プロシージャのパラメータ

パラメータ	説明
gname	レプリケート・マスター・グループの名前。
master_list	提供した DDL を実行するマスター・サイトの名前のカンマで区切られたリスト。サイト名の間に空白を挿入しないでください。デフォルト値 NULL は、マスター定義サイトも含め、すべてのサイトで DDL が実行されることを示します。
master_table	提供した DDL を実行するマスター・サイトの表。最初のマスターは位置 1、2 番目は位置 2、以下同様に設定されている必要があります。
ddl_text	指定した各マスター・サイトで実行する DDL。スキーマを指定せずに DDL を発行すると、レプリケーション管理者のスキーマがデフォルトのスキーマとして使用されます。レプリケーション管理者のスキーマを使用しない場合は、スキーマを必ず指定してください。

例外

表 37-89 EXECUTE_DDL プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
nonmaster	マスター・サイト以外のサイトが少なくとも 1 つあります。
ddlfailure	マスター定義サイトで DDL が成功しませんでした。
commfailure	アクセスできないマスター・サイトが少なくとも 1 つあります。

GENERATE_REPLICATION_SUPPORT プロシージャ

このプロシージャは、レプリケーションのサポートに必要なトリガーとパッケージを生成します。このプロシージャは、マスター定義サイトからコールする必要があります。

構文

```
DBMS_REPCAT.GENERATE_REPLICATION_SUPPORT (
  sname          IN   VARCHAR2,
  oname          IN   VARCHAR2,
  type           IN   VARCHAR2,
  package_prefix IN   VARCHAR2  := NULL,
  procedure_prefix IN  VARCHAR2  := NULL,
  distributed    IN   BOOLEAN   := TRUE,
  gen_objs_owner IN   VARCHAR2  := NULL,
  min_communication IN  BOOLEAN  := TRUE,
  generate_80_compatible IN  BOOLEAN  := TRUE);
```

パラメータ

表 37-90 GENERATE_REPLICATION_SUPPORT プロシージャのパラメータ

パラメータ	説明
sname	オブジェクトが置かれているスキーマ。
oname	レプリケーション・サポートを生成するオブジェクトの名前。
type	オブジェクトの型。サポートされている型は、TABLE、PACKAGE および PACKAGE BODY です。
package_prefix	型が PACKAGE または PACKAGE BODY のオブジェクトの場合は、生成されたラッパー・パッケージ名の前にこのパラメータの値が付加されます。デフォルトは DEFER_ です。
procedure_prefix	型が PACKAGE または PACKAGE BODY のオブジェクトの場合は、生成されたラッパー・プロシージャ名の前にこのパラメータの値が付加されます。デフォルトでは、接頭辞は割り当てられません。
distributed	この値は TRUE に設定してください。
gen_objs_owner	型が PACKAGE または PACKAGE BODY のオブジェクトの場合は、生成されたオブジェクトが作成されるスキーマを指定します。NULL の場合、オブジェクトは sname に作成されます。
min_communication	マスター・サイトのいずれかが Oracle7 リリース 7.3 を実行している場合は、FALSE を設定します。新旧の値の伝播を最小化する場合は、TRUE を設定します。デフォルトは TRUE です。詳細は、『Oracle8i レプリケーション・ガイド』を参照してください。
generate_80_compatible	マスター・サイトのいずれかが Oracle8i リリース 8.1.5 以前のバージョンの Oracle Server を実行している場合は、TRUE を設定します。レプリケート環境が純粋な Oracle8i リリース 8.1.5 以上の環境の場合は、FALSE を設定します。

例外

表 37-91 GENERATE_REPLICATION_SUPPORT プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
missingobject	指定したオブジェクトが、行レベル・レプリケーション情報を待機している表として指定のスキーマ内に存在していないか、またはラッパー生成を待機しているパッケージ（本体）として存在していません。
typefailure	指定した型パラメータはサポートされていません。
notquiesced	レプリケート・マスター・グループが停止中ではありません。
commfailure	アクセスできないマスター・サイトが少なくとも 1 つあります。
missingschema	スキーマが存在しません。
dbnotcompatible	マスターの 1 つがリリース 7.3.0.0 と互換性がありません。
notcompat	マスターの 1 つがリリース 7.3.0.0 と互換性がありません。 (dbnotcompatible と同じです。)
duplicateobject	オブジェクトがすでに存在しています。

GENERATE_SNAPSHOT_SUPPORT プロシージャ

このプロシージャは、トリガーを起動し、更新可能スナップショットのレプリケーションまたはプロシージャ型レプリケーションのサポートに必要なパッケージを生成します。このプロシージャは、スナップショット・サイトからコールする必要があります。

注意： CREATE_SNAPSHOT_REPOBJECT は、更新可能スナップショットのスナップショット・サポートを自動的に生成します。

構文

```
DBMS_REPCAT.GENERATE_SNAPSHOT_SUPPORT (
  sname          IN VARCHAR2,
  oname          IN VARCHAR2,
  type           IN VARCHAR2,
  gen_objs_owner IN VARCHAR2 := '',
  min_communication IN BOOLEAN := TRUE,
  generate_80_compatible IN BOOLEAN := TRUE);
```

パラメータ

表 37-92 GENERATE_SNAPSHOT_SUPPORT プロシージャのパラメータ

パラメータ	説明
sname	オブジェクトが置かれているスキーマ。
oname	サポートを生成するオブジェクトの名前。
type	オブジェクトの型。サポートされている型は、SNAPSHOT、PACKAGE および PACKAGE BODY です。
gen_objs_owner	型が PACKAGE または PACKAGE BODY のオブジェクトの場合は、生成されたオブジェクトが作成されるスキーマを指定します。NULL の場合、オブジェクトは SNAME に作成されます。
min_communication	TRUE の場合は、更新文で列を変更する場合に限り、更新トリガーによって列の新しい値が送信されます。その列がキー列または変更された列グループ内の列の場合、更新トリガーはその列の元の値のみ送信します。
generate_80_compatible	マスター・サイトのいずれかが Oracle8i リリース 8.1.5 以前のバージョンの Oracle Server を実行している場合は、TRUE を設定します。レプリケート環境が純粋な Oracle8i リリース 8.1.5 以上の環境の場合は、FALSE を設定します。

例外

表 37-93 GENERATE_SNAPSHOT_SUPPORT プロシージャの例外

例外	説明
nonsnapshot	起動サイトがスナップショット・サイトではありません。
missingobject	指定したオブジェクトが、行または列レベルのレプリケーション情報を待機しているスナップショットとしてレプリケート・スキーマ内に存在していないか、またはラッパー生成を待機しているパッケージ（本体）として存在していません。
typfailure	指定した型パラメータはサポートされていません。
missingschema	生成オブジェクトの指定の所有者が存在しません。
missingremoteobject	マスター・オブジェクトが、レプリケーション・サポートをまだ生成していません。
commfailure	マスターにアクセスできません。

MAKE_COLUMN_GROUP プロシージャ

このプロシージャは、1 つ以上のメンバーを含んだ新しい列グループを作成します。このプロシージャは、マスター定義サイトからコールする必要があります。

関連項目： 競合の解消方法の詳細は、『Oracle8i レプリケーション・ガイド』を参照してください。

構文

```
DBMS_REPCAT.MAKE_COLUMN_GROUP (  
    sname                IN   VARCHAR2,  
    oname                IN   VARCHAR2,  
    column_group         IN   VARCHAR2,  
    list_of_column_names IN   VARCHAR2 | DBMS_REPCAT.VARCHAR2S);
```

パラメータ

表 37-94 MAKE_COLUMN_GROUP プロシージャのパラメータ

パラメータ	説明
sname	レプリケート表が置かれているスキーマ。
oname	新しい列グループを作成するレプリケート表の名前。
column_group	作成する列グループに割り当てる名前。
list_of_column_names	グループ化する列の名前。列名は、名前のカンマで区切られたリストまたは PL/SQL 表のいずれかで示します。PL/SQL 表は、DBMS_REPCAT.VARCHAR2S 型にしてください。表内のすべての列を含んだ列グループを作成するには、単一の値 '*' を使用します。

例外

表 37-95 MAKE_COLUMN_GROUP プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
duplicategroup	指定した列グループが、表にすでに存在しています。
missingobject	指定した表が存在しません。
missingcolumn	指定した列が、指定した表内に存在していません。
duplicatecolumn	指定した列は、すでに別の列グループのメンバーです。
notquiesced	レプリケート・マスター・グループが停止中ではありません。

PURGE_MASTER_LOG プロシージャ

このプロシージャは、指定した識別番号、ソースまたはレプリケート・マスター・グループに関連している DBA_REPCATLOG ビュー内のローカル・メッセージを削除します。

構文

```
DBMS_REPCAT.PURGE_MASTER_LOG (  
    id      IN   BINARY_INTEGER,  
    source  IN   VARCHAR2,  
    gname   IN   VARCHAR2);
```

パラメータ

表 37-96 PURGE_MASTER_LOG プロシージャのパラメータ

パラメータ	説明
id	要求の識別番号。DBA_REPCATLOG ビューに表示される番号です。
source	要求発信元のマスター・サイト。
gname	要求の作成対象となったレプリケート・マスター・グループの名前。

例外

表 37-97 PURGE_MASTER_LOG プロシージャの例外

例外	説明
nonmaster	gname が NULL でなく、起動サイトがマスター・サイトではありません。

PURGE_STATISTICS プロシージャ

このプロシージャは、DBA_REPRESOLUTION_STATISTICS ビューから情報を削除します。

構文

```
DBMS_REPCAT.PURGE_STATISTICS (  
    sname      IN   VARCHAR2,  
    oname      IN   VARCHAR2,  
    start_date IN   DATE,  
    end_date   IN   DATE);
```

パラメータ

表 37-98 PURGE_STATISTICS プロシージャのパラメータ

パラメータ	説明
sname	レプリケート表が置かれているスキーマの名前。
oname	競合解消統計を削除する表の名前。
start_date/end_date	統計を削除する日付範囲。start_date が NULL の場合は、end_date までの統計がすべて削除されます。end_date が NULL の場合は、start_date 以降の統計がすべて削除されます。

例外

表 37-99 PURGE_STATISTICS プロシージャの例外

例外	説明
missingschema	指定のスキーマが存在しません。
missingobject	指定した表が存在しません。
statnotreg	この表は統計収集をするために登録されていません。

REFRESH_SNAPSHOT_REPGROUP プロシージャ

このプロシージャは、スナップショット・サイトのオブジェクト・グループを、関連するマスター・サイトからの最新データでリフレッシュします。

構文

```
DBMS_REPCAT.REFRESH_SNAPSHOT_REPGROUP (  
    gname                IN    VARCHAR2,  
    drop_missing_contents IN    BOOLEAN    := FALSE,  
    refresh_snapshots    IN    BOOLEAN    := FALSE,  
    refresh_other_objects IN    BOOLEAN    := FALSE,  
    gowner               IN    VARCHAR2    := 'PUBLIC');
```

パラメータ

表 37-100 REFRESH_SNAPSHOT_REPGROUP プロシージャのパラメータ

パラメータ	説明
gname	レプリケート・マスター・グループの名前。
drop_missing_contents	オブジェクトがレプリケート・マスター・グループから削除されている場合、そのオブジェクトはスナップショット・サイトのスキーマからは自動的に削除されません。ただし、そのオブジェクトはレプリケートされなくなります。つまり、このオブジェクトに対する変更は、関連するマスター・サイトに送信されなくなります。スナップショットは、引き続き関連するマスター表からリフレッシュされます。ただし、更新可能スナップショットに対する変更は失われます。この引数に TRUE を設定すると、オブジェクト・グループからオブジェクトが削除されるときに、そのオブジェクトをスキーマから完全に削除できます。
refresh_snapshots	これを TRUE に設定すると、レプリケート・マスター・グループ内のスナップショットの内容がリフレッシュされます。
refresh_other_objects	これを TRUE に設定すると、レプリケート・マスター・グループ内の非スナップショット・オブジェクトの内容がリフレッシュされます。
gowner	スナップショット・グループの所有者。

例外

表 37-101 REFRESH_SNAPSHOT_REPGROUP プロシージャの例外

例外	説明
nonsnapshot	起動サイトがスナップショット・サイトではありません。
nonmaster	このマスターは、マスター・サイトではありません。
commfailure	マスターにアクセスできません。
missingrepgroup	オブジェクト・グループ名が指定されていません。

REGISTER_SNAPSHOT_REPGROUP プロシージャ

このプロシージャは、DBA_REGISTERED_SNAPSHOT_GROUPS にスナップショット・グループを挿入または変更することによって、各マスター・サイトのスナップショット管理を容易にします。

構文

```
DBMS_REPCAT.REGISTER_SNAPSHOT_REPGROUP (  
  gname          IN  VARCHAR2,  
  snapsite       IN  VARCHAR2,  
  comment        IN  VARCHAR2  := NULL,  
  rep_type       IN  NUMBER     := reg_unknown,  
  fname         IN  VARCHAR2  := NULL,  
  gowner        IN  VARCHAR2  := 'PUBLIC');
```

パラメータ

表 37-102 REGISTER_SNAPSHOT_REPGROUP プロシージャのパラメータ

パラメータ	説明
gname	登録するスナップショット・オブジェクト・グループの名前。
snapsite	スナップショット・サイトのグローバル名。
comment	スナップショット・サイトに対するコメント、または既存のコメントに対する更新。
rep_type	スナップショット・グループのバージョン。代入できる有効な定数は、reg_unknown (デフォルト)、reg_v7_group、reg_v8_group および reg_repapi_group です。
fname	内部使用のためのパラメータ。オラクル社カスタマ・サポート・センターから指示がない限り、このパラメータは設定しないでください。
gowner	スナップショット・グループの所有者。

例外

表 37-103 REGISTER_SNAPSHOT_REPGROUP プロシージャの例外

例外	説明
failregsnaprepgroup	スナップショット・グループの登録に失敗しました。
missingrepgroup	オブジェクト・グループ名が指定されていません。
nullsitename	スナップショット・サイトが指定されていません。
nonmaster	このプロシージャは、スナップショットのマスター・サイトで実行する必要があります。
duplicaterepgroup	オブジェクトがすでに存在しています。

REGISTER_STATISTICS プロシージャ

このプロシージャは、表の更新競合、削除競合および一意性競合の正常な解消に関する情報を収集します。

構文

```
DBMS_REPCAT.REGISTER_STATISTICS (  
    sname IN    VARCHAR2,  
    oname IN    VARCHAR2);
```

パラメータ

表 37-104 REGISTER_STATISTICS プロシージャのパラメータ

パラメータ	説明
sname	表が置かれているスキーマの名前
oname	競合解消統計を収集する表の名前

例外

表 37-105 REGISTER_STATISTICS プロシージャの例外

例外	説明
missingschema	指定のスキーマが存在しません。
missingobject	指定した表が存在しません。

RELOCATE_MASTERDEF プロシージャ

このプロシージャは、マスター定義サイトをレプリケート環境内の別のマスター・サイトに変更します。

RELOCATE_MASTERDEF のコール時に、元のマスター定義サイトと新しいマスター定義サイトのいずれも使用可能である必要はありません。計画的に再構成する場合は、notify_masters に TRUE および include_old_masterdef に TRUE を設定して RELOCATE_MASTERDEF を起動してください。

構文

```
DBMS_REPCAT.RELOCATE_MASTERDEF (
    gname                IN   VARCHAR2,
    old_masterdef         IN   VARCHAR2,
    new_masterdef        IN   VARCHAR2,
    notify_masters        IN   BOOLEAN    := TRUE,
    include_old_masterdef IN   BOOLEAN    := TRUE,
    require_flavor_change IN   BOOLEAN    := FALSE);
```

パラメータ

表 37-106 RELOCATE_MASTERDEF プロシージャのパラメータ

パラメータ	説明
gname	マスター定義を再配置するオブジェクト・グループの名前。
old_masterdef	現行のマスター定義サイトの完全修飾データベース名。
new_masterdef	新しいマスター定義サイトにする既存のマスター・サイトの完全修飾データベース名。
notify_masters	TRUE の場合、このプロシージャは、変更内容をすべてのマスターに同期式でマルチキャストします (include_old_masterdef が TRUE の場合のみ old_masterdef も含まれます)。変更を適用できないマスターがある場合は、すべてのマスターで変更がロールバックされます。 マスター定義サイトでのみ障害が発生した場合は、notify_masters に TRUE および include_old_masterdef に FALSE を設定して RELOCATE_MASTERDEF を起動してください。複数のマスター・サイトとマスター定義サイトで障害が発生した場合、管理者は、notify_masters に FALSE を設定し、各マスターで RELOCATE_MASTERDEF を起動する必要があります。
include_old_masterdef	notify_masters が TRUE で、include_old_masterdef も TRUE の場合は、元のマスター定義サイトにも変更内容が通知されます。

表 37-106 RELOCATE_MASTERDEF プロシージャのパラメータ

パラメータ	説明
require_flavor_change	内部使用のためのパラメータ。オラクル社カスタマ・サポート・センターから指示がない限り、このパラメータは設定しないでください。

例外

表 37-107 RELOCATE_MASTERDEF プロシージャの例外

例外	説明
nonmaster	NEW_MASTERDEF がマスター・サイトではないか、または起動サイトがマスター・サイトではありません。
nonmasterdef	old_masterdef がマスター定義サイトではありません。
commfailure	notify_masters が TRUE ですが、アクセスできないマスター・サイトが少なくとも 1 つあります。

REMOVE_MASTER_DATABASES プロシージャ

このプロシージャは、レプリケート環境から 1 つ以上のマスター・データベースを削除します。また、トリガーと関連するパッケージを、残りのマスター・サイトで再生成します。このプロシージャは、マスター定義サイトからコールする必要があります。

構文

```
DBMS_REPCAT.REMOVE_MASTER_DATABASES (  
  gname          IN   VARCHAR2,  
  master_list     IN   VARCHAR2 |  
  master_table    IN   DBMS_UTILITY.DBLINK_ARRAY);
```

注意： このプロシージャはオーバーロードされています。 master_list パラメータと master_table パラメータは、両方同時には指定できません。

パラメータ

表 37-108 REMOVE_MASTER_DATABASES プロシージャのパラメータ

パラメータ	説明
gname	レプリケート環境に関連付けられているオブジェクト・グループの名前。マスター・データベースが複数のレプリケート環境で使用されている場合に、このパラメータによって混乱を避けることができます。
master_list	レプリケート環境から削除する完全修飾マスター・データベース名のカンマで区切られたリスト。リストの名前の間に空白を挿入しないでください。
master_table	リストのかわりに、DBMS_UTILITY.DBLINK_ARRAY 型の PL/SQL 表でデータベース名を指定できます。

例外

表 37-109 REMOVE_MASTER_DATABASES プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
nonmaster	指定したデータベースの中に、マスター・サイト以外のデータベースが少なくとも 1 つあります。
reconfigerror	指定したデータベースの 1 つが、マスター定義サイトです。
commfailure	残りのマスター・サイトの中に、アクセスできないサイトが少なくとも 1 つあります。

REPCAT_IMPORT_CHECK プロシージャ

このプロシージャは、レプリケート・オブジェクトまたは Oracle レプリケーションが使用しているオブジェクトのエクスポートまたはインポートを実行した後で、レプリケート・マスター・グループ内のオブジェクトの識別子と状態値が適切であることを確認します。

構文

```
DBMS_REPCAT.REPCAT_IMPORT_CHECK (  
    gname      IN   VARCHAR2 ,  
    master     IN   BOOLEAN ,  
    gowner     IN   VARCHAR2  := 'PUBLIC') ;
```

パラメータ

表 37-110 REPCAT_IMPORT_CHECK プロシージャのパラメータ

パラメータ	説明
gname	レプリケート・マスター・グループの名前。パラメータを両方とも省略すると、現行サイトのすべてのレプリケート・マスター・グループがチェックされます。
master	マスター・サイトをチェックする場合は TRUE、スナップショット・サイトをチェックする場合は FALSE を設定します。
gowner	マスター・グループの所有者。

例外

表 37-111 REPCAT_IMPORT_CHECK プロシージャの例外

例外	説明
nonmaster	master が TRUE ですが、データベースがそのオブジェクト・グループのマスター・サイトではないか、または該当のデータベースではありません。
nonsnapshot	master が FALSE ですが、データベースがそのオブジェクト・グループのスナップショット・サイトではありません。
missingobject	オブジェクト・グループ内に有効なレプリケート・オブジェクトが存在しません。
missingrepgroup	指定したレプリケート・オブジェクト・グループが存在しません。
missingschema	指定したレプリケート・オブジェクト・スキーマが存在しません。

RESUME_MASTER_ACTIVITY プロシージャ

このプロシージャは、レプリケート環境の休止後、通常のレプリケーション・アクティビティを再開します。

構文

```
DBMS_REPCAT.RESUME_MASTER_ACTIVITY (  
  gname      IN  VARCHAR2,  
  override   IN  BOOLEAN := FALSE);
```

パラメータ

表 37-112 RESUME_MASTER_ACTIVITY プロシージャのパラメータ

パラメータ	説明
gname	レプリケート・マスター・グループの名前。
override	TRUE の場合、保留中の RepCat 管理要求は無視され、通常のレプリケーション・アクティビティが各マスターで可能な限り迅速に再開されます。これは、緊急の状況でのみ指定してください。 FALSE の場合は、各マスターの gname に対する保留中の RepCat 管理要求がない場合に限り、そのマスターで通常のレプリケーション・アクティビティが再開されます。

例外

表 37-113 RESUME_MASTER_ACTIVITY プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
notquiesced	レプリケート・マスター・グループが休止中または停止中ではありません。
commfailure	アクセスできないマスター・サイトが少なくとも 1 つあります。
notallgenerated	レプリケーション・アクティビティを再開する前に、レプリケーション・サポートを生成してください。

SEND_OLD_VALUES プロシージャ

更新と削除のために、レプリケート表の各非キー列の元の値を送信するオプションがあります。デフォルトでは、すべての列の元の値が送信されます。DBMS_REPCAT.SEND_OLD_VALUES をマスター定義サイトで起動すると、すべてのマスター・サイトとスナップショット・サイトでこの動作を変更できます。

構文

```
DBMS_REPCAT.SEND_OLD_VALUES (
    sname          IN  VARCHAR2,
    oname          IN  VARCHAR2,
    { column_list  IN  VARCHAR2,
    | column_table IN  DBMS_REPCAT.VARCHAR2S, }
    operation      IN  VARCHAR2 := 'UPDATE',
    send           IN  BOOLEAN := TRUE );
```

注意： このプロシージャはオーバーロードされています。column_list パラメータと column_table パラメータは、両方同時には指定できません。

パラメータ

表 37-114 SEND_OLD_VALUES プロシージャのパラメータ

パラメータ	説明
sname	表が置かれているスキーマ。
oname	レプリケート表の名前。
column_list	表内の列のカンマで区切られたリスト。エントリ間に空白を挿入しないでください。
column_table	リストのかわりに、列名を含んだ DBMS_REPCAT.VARCHAR2S 型の PL/SQL 表を使用できます。最初の列名は位置 1、2 番目は位置 2、以下同様に設定されている必要があります。
operation	有効な値は、UPDATE、DELETE、またはアスタリスクのワイルドカード '*'（更新と削除を意味します）です。
send	TRUE の場合は、指定した列の元の値が送信されます。FALSE の場合は、指定した列の元の値が送信されません。指定外の列と指定外の操作には影響しません。 マスター定義サイトでは、表の min_communication が TRUE になると、指定した変更がすぐに有効となります。変更内容がマスター・サイトまたはスナップショット・サイトで有効になるのは、min_communication に TRUE を設定して、次回そのサイトでレプリケーション・サポートを生成したときです。

注意： operation パラメータを使用すると、行の削除時または非キー列の更新時に、非キー列の元の値を送信するかどうかを決定できます。元の値を送信しないと、元の値のかわりに NULL が送信され、更新または削除の適用時に送信先の現行の列値と元の値が等しいとみなされます。

Oracle のデフォルト動作を変更する前に、データ伝播の最小化について『Oracle8i レプリケーション・ガイド』を参照してください。

例外

表 37-115 SEND_OLD_VALUES プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
missingobject	指定したオブジェクトが、行レベル・レプリケーション情報を待機している表として指定のスキーマ内に存在していません。
missingcolumn	表に存在しない列が少なくとも 1 つあります。
notquiesced	レプリケート・マスター・グループが停止中ではありません。
typefailure	無効な操作が指定されています。

SET_COLUMNS プロシージャ

このプロシージャは、行レベル・レプリケーションの使用時に、主キーのかわりに代替列または列グループを使用するために、表のどの列を比較するかを判断します。このプロシージャは、マスター定義サイトからコールする必要があります。

関連項目： 詳細は、『Oracle8i レプリケーション・ガイド』を参照してください。

構文

```
DBMS_REPCAT.SET_COLUMNS (
  sname          IN   VARCHAR2,
  oname          IN   VARCHAR2,
  { column_list  IN   VARCHAR2
  | column_table IN   DBMS_UTILITY.NAME_ARRAY } );
```

注意： このプロシージャはオーバーロードされています。column_list パラメータと column_table パラメータは、両方同時には指定できません。

パラメータ

表 37-116 SET_COLUMNS プロシージャのパラメータ

パラメータ	説明
sname	表が置かれているスキーマ。
oname	表の名前。
column_list	表の中で主キーとして使用する列のカンマで区切られたリスト。エン トリ間に空白を挿入しないでください。
column_table	リストのかわりに、列名を含んだ型 DBMS_UTILITY.NAME_ARRAY の PL/SQL 表を使用できます。最初の列名は位置 1、2 番目は位置 2、以 下同様に設定されている必要があります。

例外

表 37-117 SET_COLUMNS プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
missingobject	指定したオブジェクトが、行レベル・レプリケーション情報を待機し ている表として指定のスキーマ内に存在していません。
missingcolumn	表に存在しない列が少なくとも 1 つあります。
notquiesced	レプリケート・マスター・グループが休止中または停止中ではありま せん。

SUSPEND_MASTER_ACTIVITY プロシージャ

このプロシージャは、マスター・グループのレプリケーション・アクティビティを中断します。このプロシージャは、マスター定義サイトからコールする必要があります。

構文

```
DBMS_REPCAT.SUSPEND_MASTER_ACTIVITY (
    gname    IN    VARCHAR2);
```

パラメータ

表 37-118 SUSPEND_MASTER_ACTIVITY プロシージャのパラメータ

パラメータ	説明
gname	アクティビティを中断するマスター・グループの名前

例外

表 37-119 SUSPEND_MASTER_ACTIVITY プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
notnormal	レプリケート・マスター・グループが通常の操作モードではありません。
commfailure	アクセスできないマスター・サイトが少なくとも 1 つあります。

SWITCH_SNAPSHOT_MASTER プロシージャ

このプロシージャは、スナップショット・レプリケート・マスター・グループのマスター・データベースを別のマスター・サイトに変更します。影響するスナップショットは完全にリフレッシュされ、必要に応じて、トリガーと関連するパッケージが再生成されます。このプロシージャは、マスターの変更前に、元のマスター・サイトにキューを送信しません。

構文

```
DBMS_REPCAT.SWITCH_SNAPSHOT_MASTER (
    gname      IN    VARCHAR2,
    master     IN    VARCHAR2,
    gowner     IN    VARCHAR2 := 'PUBLIC');
```

パラメータ

表 37-120 SWITCH_SNAPSHOT_MASTER プロシージャのパラメータ

パラメータ	説明
gname	マスター・サイトを変更するスナップショット・オブジェクト・グループの名前
master	そのスナップショット・サイトに使用する新しいマスター・データベースの完全修飾データベース名
gowner	スナップショット・グループの所有者

例外

表 37-121 SWITCH_SNAPSHOT_MASTER プロシージャの例外

例外	説明
nonsnapshot	起動サイトがスナップショット・サイトではありません。
nonmaster	指定したデータベースはマスター・サイトではありません。
commfailure	指定したデータベースにアクセスできません。
missingrepgroup	スナップショット・グループが存在しません。
qrytoolong	スナップショット定義の間合せが 32KB を超えています。
alreadymastered	ローカル・サイトに、同じグループ名を持ち、元のマスター・サイトでマスターとなっている別のスナップショット・グループがあります。

UNREGISTER_SNAPSHOT_REPGROUP プロシージャ

このプロシージャは、DBA_REGISTERED_SNAPSHOT_GROUPS からスナップショット・グループを削除することによって、各マスター・サイトのスナップショット管理を容易にします。

構文

```
DBMS_REPCAT.UNREGISTER_SNAPSHOT_REPGROUP (  
  gname      IN   VARCHAR2,  
  snapsite   IN   VARCHAR2,  
  gowner     IN   VARCHAR2  := 'PUBLIC');
```


パラメータ

表 37-122 UNREGISTER_SNAPSHOT_REPGROUP プロシージャのパラメータ

パラメータ	説明
gname	登録解除するスナップショット・オブジェクト・グループの名前
snapsite	スナップショット・サイトのグローバル名
gowner	スナップショット・グループの所有者

VALIDATE ファンクション

このファンクションは、複数マスター・レプリケーション環境のキー状態が正しいかどうかを検証します。

構文

```
DBMS_REPCAT.VALIDATE (
    gname           IN  VARCHAR2,
    check_genflags  IN  BOOLEAN := FALSE,
    check_valid_objs IN  BOOLEAN := FALSE,
    check_links_sched IN  BOOLEAN := FALSE,
    check_links     IN  BOOLEAN := FALSE,
    error_table     OUT DBMS_REPCAT.VALIDATE_ERR_TABLE)
RETURN BINARY_INTEGER;
```

```
DBMS_REPCAT.VALIDATE (
    gname           IN  VARCHAR2,
    check_genflags  IN  BOOLEAN := FALSE,
    check_valid_objs IN  BOOLEAN := FALSE,
    check_links_sched IN  BOOLEAN := FALSE,
    check_links     IN  BOOLEAN := FALSE,
    error_msg_table  OUT DBMS_UTILITY.UNCL_ARRAY,
    error_num_table  OUT DBMS_UTILITY.NUMBER_ARRAY )
RETURN BINARY_INTEGER;
```

注意： このファンクションはオーバーロードされています。VALIDATE の戻り値は、検出されたエラーの数です。このファンクションの OUT パラメータには、検出されたエラーが戻されます。前述の構文に記述されている最初のインタフェース・ファンクションでは、ERROR_TABLE はレコードの配列で構成されています。各レコードには、VARCHAR2 と NUMBER があります。文字列フィールドにはエラー・メッセージが格納され、番号フィールドには Oracle のエラー番号が格納されます。

前述の構文に記述されている 2 番目のインタフェース・ファンクションは、2 つの OUT 配列があること以外は同じです。VARCHAR2 配列にエラー・メッセージが格納され、NUMBER 配列にエラー番号が格納されます。

パラメータ

表 37-123 VALIDATE ファンクションのパラメータ

パラメータ	説明
gname	検証するマスター・グループの名前。
check_genflags	グループ内のオブジェクトがすべて生成済であるかどうかをチェックします。このチェックは、マスター定義サイトでのみ実行してください。
check_valid_objs	グループ内のオブジェクトに対応する基礎オブジェクトが有効かどうかをチェックします。このチェックは、マスター定義サイトでのみ実行してください。マスター定義サイトは他のすべてのサイトを調べて、基礎となっているオブジェクトが有効であることを確認します。オブジェクトの有効性は、接続ユーザーのスキーマ内でチェックされます。
check_links_sched	リンクの実行がスケジュールされているかどうかをチェックします。このチェックは、各マスター・サイトで起動してください。
check_links	接続ユーザー（レプリケーション管理者）とプロバゲータに、レプリケーションが適切に動作するための正しいリンクがあるかどうかをチェックします。リンクがデータベースに存在していて、アクセス可能であることを確認します。このチェックは、各マスター・サイトで起動してください。
error_table	検出されたすべてのエラーのメッセージと番号を戻します。
error_msg_table	検出されたすべてのエラーのメッセージを戻します。
error_num_table	検出されたすべてのエラーの番号を戻します。

例外

表 37-124 VALIDATE ファンクションの例外

例外	説明
missingdblink	データベース・リンクがレプリケーション・プロパゲータのスキーマ内に存在していないか、またはスケジュールされていません。データベース・リンクがデータベースに存在していてアクセス可能であること、および実行がスケジュールされていることを確認してください。
dblinkmismatch	ローカル・ノードのデータベース・リンク名が、そのリンクがアクセスするデータベースのグローバル名と一致しません。global_names 初期化パラメータに TRUE が設定され、そのリンク名がグローバル名と一致していることを確認してください。
dblinkuidmismatch	ローカル・ノードのレプリケーション管理ユーザーのユーザー名と、そのデータベース・リンクに対応するノードのユーザー名が同じではありません。アドバンスド・レプリケーションでは、両方のユーザーが同じであることが必要です。ローカル・ノードのレプリケーション管理ユーザーのユーザー ID と、そのデータベース・リンクに対応するノードのユーザー ID を同じにしてください。
objectnotgenerated	オブジェクトが、他のマスター・サイトで生成されていないか、または生成中です。マスター定義サイトのオブジェクトに対して GENERATE_REPLICATION_SUPPORT と DO_DEFERRED_REPCAT_ADMIN をコールして、オブジェクトを確実に生成してください。
opnotsupported	オブジェクト・グループがバージョン 8 より前のノードでレプリケートされている場合は、操作がサポートされていません。レプリケート・マスター・グループのすべてのノードが Oracle バージョン 8 であることを確認してください。

使用上の注意

VALIDATE の戻り値は、検出されたエラーの数です。このファンクションの OUT パラメータには、検出されたエラーが戻されます。最初のインタフェース・ファンクションでは、ERROR_TABLE はレコードの配列で構成されています。各レコードには、VARCHAR2 と NUMBER があります。文字列フィールドにはエラー・メッセージが格納され、番号フィールドには Oracle のエラー番号が格納されます。

2 番目のインタフェースは、OUT 配列が 2 つある以外は最初のインタフェースと同じです。VARCHAR2 配列にエラー・メッセージが格納され、NUMBER 配列にエラー番号が格納されず。

WAIT_MASTER_LOG プロシージャ

このプロシージャは、マスター・サイトに非同期で伝播された変更内容が適用されたかどうかを判断します。

構文

```
DBMS_REPCAT.WAIT_MASTER_LOG (  
    gname          IN    VARCHAR2,  
    record_count    IN    NATURAL,  
    timeout         IN    NATURAL,  
    true_count      OUT   NATURAL);
```

パラメータ

表 37-125 WAIT_MASTER_LOG プロシージャのパラメータ

パラメータ	説明
gname	レプリケート・マスター・グループの名前。
record_count	未完了のアクティビティ件数がこのしきい値以下になるたびにプロシージャに戻ります。
timeout	プロシージャが戻るまで待機する最大秒数。
true_count (out parameter)	未完了のアクティビティの数を戻します。

例外

表 37-126 WAIT_MASTER_LOG プロシージャの例外

例外	説明
nonmaster	起動サイトがマスター・サイトではありません。

DBMS_REPCAT_ADMIN

DBMS_REPCAT_ADMIN によって、対称型レプリケーション機能に必要な権限を付与したユーザーを作成できます。

DBMS_REPCAT_ADMIN パッケージ

サブプログラムの要約

表 38-1 DBMS_REPCAT_ADMIN パッケージのサブプログラム

サブプログラム	説明
38-3 ページの GRANT_ADMIN_ANY_SCHEMA プロシージャ	現行のサイトでレプリケート・マスター・グループを管理するために必要な権限を、レプリケーション管理者に付与します。
38-3 ページの GRANT_ADMIN_SCHEMA プロシージャ	現行のサイトでスキーマを管理するために必要な権限を、レプリケーション管理者に付与します。
38-4 ページの REGISTER_USER_REPGROUP プロシージャ	リモート・サイトで使用するための代理スナップショット管理者権限または受信者権限を、マスター・サイトで割り当てます。
38-6 ページの REVOKE_ADMIN_ANY_SCHEMA プロシージャ	GRANT_ADMIN_ANY_SCHEMA によって付与された権限とロールを、レプリケーション管理者から取り消します。
38-7 ページの REVOKE_ADMIN_SCHEMA プロシージャ	GRANT_ADMIN_SCHEMA によって付与された権限とロールを、レプリケーション管理者から取り消します。
38-8 ページの UNREGISTER_USER_REPGROUP プロシージャ	REGISTER_USER_REPGROUP プロシージャによって付与された権限とロールを、代理スナップショット管理者または受信者から取り消します。

GRANT_ADMIN_ANY_SCHEMA プロシージャ

このプロシージャは、現行のサイトでレプリケート・マスター・グループを管理するために必要な権限を、レプリケーション管理者に付与します。

構文

```
DBMS_REPCAT_ADMIN.GRANT_ADMIN_ANY_SCHEMA (  
    username IN VARCHAR2);
```

パラメータ

表 38-2 GRANT_ADMIN_ANY_SCHEMA プロシージャのパラメータ

パラメータ	説明
username	現行のサイトでレプリケート・マスター・グループを管理するために必要な権限とロールを付与するレプリケーション管理者の名前

例外

表 38-3 GRANT_ADMIN_ANY_SCHEMA プロシージャの例外

例外	説明
ORA-01917	ユーザーが存在しません。

GRANT_ADMIN_SCHEMA プロシージャ

このプロシージャは、現行のサイトでスキーマを管理するために必要な権限をレプリケーション管理者に付与します。このプロシージャは、オブジェクト・グループが単一のスキーマ内にある場合に最も役立ちます。

構文

```
DBMS_REPCAT_ADMIN.GRANT_ADMIN_SCHEMA (  
    username IN VARCHAR2);
```

パラメータ

表 38-4 GRANT_ADMIN_SCHEMA プロシージャのパラメータ

パラメータ	説明
username	レプリケーション管理者の名前。このユーザーには、現行のサイトでレプリケート・マスター・グループにある同名のスキーマを管理するために必要な権限とロールが付与されます。

例外

表 38-5 GRANT_ADMIN_SCHEMA プロシージャの例外

例外	説明
ORA-01917	ユーザーが存在しません。

REGISTER_USER_REPGROUP プロシージャ

このプロシージャは、リモート・サイトで使用するための代理スナップショット管理者権限または受信者権限を、マスター・サイトで割り当てます。このプロシージャで付与されるのは、代理スナップショット管理者または受信者に必要な権限のみです。GRANT_ADMIN_SCHEMA または GRANT_ADMIN_ANY_SCHEMA プロシージャで付与される強力な権限は付与されません。

関連項目： 信頼性のあるセキュリティ・モデルと信頼性の低いセキュリティ・モデルの詳細は、『Oracle8i レプリケーション・マネージメント API リファレンス』の付録 A「セキュリティ・オプション」を参照してください。

構文

```
DBMS_REPCAT_ADMIN.REGISTER_USER_REPGROUP (  
  username          IN  VARCHAR2,  
  privilege_type    IN  VARCHAR2,  
  {list_of_gnames   IN  VARCHAR2 |  
  table_of_gnames   IN  DBMS_UTILITY.NAME_ARRAY} );
```

注意： このプロシージャはオーバーロードされています。list_of_gnames パラメータと table_of_gnames パラメータは、両方同時には指定できません。

パラメータ

表 38-6 REGISTER_USER_REPGROUP プロシージャのパラメータ

パラメータ	説明
username	代理スナップショット管理者権限または受信者権限のいずれかを付与するユーザーの名前。
privilege_type	割り当てる権限タイプ。privilege_type の定義には次の値を使用します。 RECEIVER: 受信者権限 PROXY_SNAPADMIN: 代理スナップショット管理者権限
list_of_gnames	ユーザーの受信者権限を登録するオブジェクト・グループのカンマで区切られたリスト。リストのエントリ間に空白を挿入しないでください。NULL を設定すると、このプロシージャのコール時点で認識されていないオブジェクト・グループも含め、すべてのオブジェクト・グループに対してそのユーザーが登録されます。NULL を設定するには、名前表記法を使用する必要があります。リスト内に無効なオブジェクト・グループがあると、リスト全体の登録に失敗します。
table_of_gnames	ユーザーの受信者権限を登録するオブジェクト・グループの PL/SQL 表。PL/SQL 表の型は DBMS_UTILITY.NAME_ARRAY にしてください。この表は 1 が基準（1 の位置から開始し、1 ずつ増加します）です。値に NULL を使用すると、すべてのオブジェクト・グループに対してそのユーザーが登録されます。表内に無効なオブジェクト・グループがあると、表全体の登録に失敗します。

例外

表 38-7 REGISTER_USER_REPGROUP プロシージャの例外

例外	説明
nonmaster	指定したオブジェクト・グループが存在していないか、または起動データベースがマスターではありません。
ORA-01917	ユーザーが存在しません。
typefailure	権限タイプの指定に誤りがあります。

REVOKE_ADMIN_ANY_SCHEMA プロシージャ

このプロシージャは、GRANT_ADMIN_ANY_SCHEMA によって付与された権限とロールを、レプリケーション管理者から取り消します。

注意： GRANT_ADMIN_ANY_SCHEMA とは別に付与された同一の権限とロールも取り消されます。

構文

```
DBMS_REPCAT_ADMIN.REVOKE_ADMIN_ANY_SCHEMA (  
    username IN VARCHAR2);
```

パラメータ

表 38-8 REVOKE_ADMIN_ANY_SCHEMA プロシージャのパラメータ

パラメータ	説明
username	権限を取り消すレプリケーション管理者の名前

例外

表 38-9 REVOKE_ADMIN_ANY_SCHEMA プロシージャの例外

例外	説明
ORA-01917	ユーザーが存在しません。

REVOKE_ADMIN_SCHEMA プロシージャ

このプロシージャは、GRANT_ADMIN_SCHEMA によって付与された権限とロールを、レプリケーション管理者から取り消します。

注意： GRANT_ADMIN_SCHEMA とは別に付与された同一の権限とロールも取り消されます。

構文

```
DBMS_REPCAT_ADMIN.REVOKE_ADMIN_SCHEMA (  
    username IN VARCHAR2);
```

パラメータ

表 38-10 REVOKE_ADMIN_SCHEMA プロシージャのパラメータ

パラメータ	説明
username	権限を取り消すレプリケーション管理者の名前

例外

表 38-11 REVOKE_ADMIN_SCHEMA プロシージャの例外

例外	説明
ORA-01917	ユーザーが存在しません。

UNREGISTER_USER_REPGROUP プロシージャ

このプロシージャは、REGISTER_USER_REPGROUP プロシージャによって付与された権限とロールを、代理スナップショット管理者または受信者から取り消します。

構文

```
DBMS_REPCAT_ADMIN.UNREGISTER_USER_REPGROUP (  
    username          IN   VARCHAR2,  
    privilege_type    IN   VARCHAR2,  
    {list_of_gnames  IN   VARCHAR2 |  
    table_of_gnames  IN   DBMS_UTILITY.NAME_ARRAY} );
```

注意： このプロシージャはオーバーロードされています。list_of_gnames パラメータと table_of_gnames パラメータは、両方同時には指定できません。

パラメータ

表 38-12 UNREGISTER_USER_REPGROUP プロシージャのパラメータ

パラメータ	説明
username	登録解除するユーザーの名前。
privilege_type	取り消す権限タイプ。privilege_type の定義には次の値を使用します。 RECEIVER: 受信者権限 PROXY_SNAPADMIN: 代理スナップショット管理者権限
list_of_gnames	ユーザーの受信者権限の登録を解除するオブジェクト・グループのカンマで区切られたリスト。リストのエントリ間に空白を挿入しないでください。NULL を設定すると、登録されているすべてのオブジェクト・グループに対する登録が解除されます。NULL を設定するには、名前表記法を使用する必要があります。リスト内に無効なオブジェクト・グループがあると、リスト全体の登録解除に失敗します。
table_of_gnames	ユーザーの受信者権限の登録を解除するオブジェクト・グループの PL/SQL 表。PL/SQL 表の型は DBMS_UTILITY.NAME_ARRAY にしてください。この表は 1 が基準（1 の位置から開始し、1 ずつ増加します）です。値に NULL を使用すると、登録されているすべてのオブジェクト・グループに対してそのユーザーの登録が解除されます。表内に無効なオブジェクト・グループがあると、表全体の登録解除に失敗します。

例外

表 38-13 UNREGISTER_USER_REPGROUP プロシージャの例外

例外	説明
nonmaster	指定したオブジェクト・グループが存在していないか、または起動データベースがマスターではありません。
ORA-01917	ユーザーが存在しません。
typefailure	権限タイプの指定に誤りがあります。

DBMS_REPCAT_INSTANTIATE

DBMS_REPCAT_INSTANTIATE パッケージは、配置テンプレートをインスタンス化します。

DBMS_REPCAT_INSTANTIATE パッケージ

サブプログラムの要約

表 39-1 DBMS_REPCAT_INSTANTIATE パッケージのサブプログラム

サブプログラム	説明
39-3 ページの DROP_SITE_INSTANTIATION プロ シージャ	DBA_REPCAT_TEMPLATE_SITES ビューからターゲット・サ イトを削除するパブリック・プロシージャ。
39-3 ページの INSTANTIATE_OFFLINE ファンク ション	マスター・サイトでスクリプトを生成するパブリック・ファン クション。オフライン時にリモート・スナップショット・サイ トでスナップショット環境を作成するために使用します。
39-6 ページの INSTANTIATE_OFFLINE_REPAPI ファンクション	マスター・サイトでバイナリ・ファイルを生成するパブリッ ク・ファンクション。オフライン時に RepAPI リモート・ス ナップショット・サイトでスナップショット環境を作成するた めに使用します。
39-9 ページの INSTANTIATE_ONLINE ファンク ション	マスター・サイトでスクリプトを生成するパブリック・ファン クション。オンライン時にリモート・スナップショット・サイ トでスナップショット環境を作成するために使用します。

DBMS_REPCAT_INSTANTIATE パッケージの詳細は、次のファイルを参照してください。

`ORACLE_HOME/rdbms/admin/dbmsrint.sql`

DROP_SITE_INSTANTIATION プロシージャ

このプロシージャは、ターゲット・サイトのテンプレート・インスタンスエーションを削除します。また、マスター・サイトの関連メタデータをすべて削除し、指定したサイトにおけるスナップショットのリフレッシュを使用禁止にします。このプロシージャは、テンプレートを最初にインスタンス化したユーザーとして実行する必要があります。テンプレートをインスタンス化したユーザーを調べるには、ALL_REPCAT_TEMPLATE_SITES ビューを参照してください。

構文

```
DBMS_REPCAT_INSTANTIATE.DROP_SITE_INSTANTIATION(
    refresh_template_name IN VARCHAR2,
    site_name              IN   VARCHAR2);
```

表 39-2 DROP_SITE_INSTANTIATION プロシージャのパラメータ

パラメータ	説明
refresh_template_name	削除する配置テンプレートの名前
site_name	指定したテンプレート・インスタンスエーションを削除する Oracle Server のサイト

INSTANTIATE_OFFLINE ファンクション

このファンクションは、マスター・サイトでファイルを生成します。このファイルは、オフライン時にリモート・スナップショット・サイトでスナップショット環境を作成するために使用します。生成されたファイルはオフライン・インスタンスエーション・ファイルです。このファイルは、長時間マスター・サイトに接続したままにできないリモート・スナップショット・サイトで使用してください。

これは、スナップショット・サイトがラップトップの場合に便利な方法です。Replication Manager のパッケージ作成ツールを使用して、生成したファイルとデータを 1 つのファイルにパッケージ化すると、このファイルを FTP サイトに転記したり、CD-ROM やフロッピー・ディスクなどにロードすることができます。

関連項目： 詳細は、『Oracle8i レプリケーション・ガイド』および Replication Manager のオンライン・ヘルプを参照してください。

このファンクションで生成されたスクリプトは、USER_REPCAT_TEMP_OUTPUT 一時ビューに格納され、配置テンプレートの配布時に、Replication Manager などの Oracle Tools で使用されます。このファンクションで戻される数値は、USER_REPCAT_TEMP_OUTPUT ビューから該当する情報を取り出すために使用されます。

このパブリック・ファンクションを実行するユーザーは、指定したサイトでインスタンス化されたテンプレートの登録ユーザーになります。

注意： このファンクションは、配置テンプレートのオフライン・インスタンスエーションの実行時に使用されます。

このファンクションを、DBMS_OFFLINE_OG パッケージ内のプロシージャ（マスター表のオフライン・インスタンスエーションの実行に使用）や DBMS_OFFLINE_SNAPSHOT パッケージ内のプロシージャ（スナップショットのオフライン・インスタンスエーションの実行に使用）と混同しないでください。それぞれのパッケージの使用方法は、該当の章を参照してください。

構文

```
DBMS_REPCAT_INSTANTIATE.INSTANTIATE_OFFLINE(  
    refresh_template_name    IN    VARCHAR2,  
    site_name                IN    VARCHAR2,  
    runtime_parm_id          IN    NUMBER    := -1e-130,  
    next_date                IN    DATE      := SYSDATE,  
    interval                 IN    VARCHAR2  := 'SYSDATE + 1',  
    use_default_gowner       IN    BOOLEAN   := TRUE)  
return NUMBER;
```

表 39-3 INSTANTIATE_OFFLINE ファンクションのパラメータ

パラメータ	説明
refresh_template_name	インスタンス化する配置テンプレートの名前。
site_name	配置テンプレートをインスタンス化するリモート・サイトの名前。
runtime_parm_id	INSERT_RUNTIME_PARMS プロシージャを使用してランタイム・パラメータ値を定義している場合は、ランタイム・パラメータの作成時に使用した ID (GET_RUNTIME_PARM_ID ファンクションを使用して取り出された ID です) を指定します。
next_date	リフレッシュ・グループの作成時に使用される次回リフレッシュ日付の値。
interval	リフレッシュ・グループの作成時に使用されるリフレッシュ間隔。
use_default_gowner	TRUE の場合、作成されたすべてのスナップショット・オブジェクト・グループはデフォルト・ユーザー PUBLIC が所有します。 FALSE の場合、作成されたすべてのスナップショット・オブジェクト・グループはインスタンス化を実施したユーザーが所有します。

表 39-4 INSTANTIATE_OFFLINE ファンクションの例外

例外	説明
miss_refresh_template	指定した配置テンプレート名が無効か、または存在しません。
dupl_template_site	スナップショット・サイトで配置テンプレートがすでにインスタンス化されています。特定のスナップショット・サイトでインスタンス化できる配置テンプレートは 1 つのみです。
not_authorized	ユーザーがインスタンス化しようとした配置テンプレートは、インスタンス化を認可されていません。

戻り値

表 39-5 INSTANTIATE_OFFLINE ファンクションの戻り値

戻り値	説明
< システム生成番号 >	生成したインスタンスエーション・スクリプトを USER_REPCAT_TEMP_OUTPUT ビューで取り出すように選択したとき、output_id に対して生成されるシステム番号を指定します。

INSTANTIATE_OFFLINE_REPAPI ファンクション

このファンクションは、マスター・サイトでファイルを生成します、このファイルは、オフライン時にリモートの RepAPI スナップショット・サイトでスナップショット環境を作成するために使用されます。オフライン・インスタンスエーション・ファイルは、長時間マスター・サイトに接続したままにできないリモートの RepAPI サイトで使用してください。

これは、リモート・スナップショット・サイトが Oracle8i Lite (RepAPI を含めて) を実行しているラップトップの場合に理想的なソリューションとなります。生成したファイルは、FTP サイトに転記したり、CD-ROM やフロッピー・ディスクなどにロードできます。

このファンクションで生成されたファイルは、パラメータ OFFLINE_DIRPATH で指定したディレクトリのマスター・サイトに格納されます。ファイルは USER_NAME、REFRESH_TEMPLATE_NAME および SITE_ID に基づいて命名され、ファイル拡張子 .oli が指定されます。たとえば、サイト 1234 でテンプレート名 MYTEMPLATE のユーザー SCOTT に対するオフライン・インスタンスエーションには、次のような名前が指定されます。

scott_mytemplate_1234.oli.

これは、接続ユーザーに対するオフライン・インスタンスエーション・ファイルを生成するためのパブリック・ファンクションです。

注意： このファンクションは、配置テンプレートのオフライン・インスタンスエーションを実行するときに使用されます。

このファンクションを、DBMS_OFFLINE_OG パッケージ内のプロシージャ（マスター表のオフライン・インスタンスエーションの実行に使用）や DBMS_OFFLINE_SNAPSHOT パッケージ内のプロシージャ（スナップショットのオフライン・インスタンスエーションの実行に使用）と混同しないでください。それぞれのパッケージの使用方法は、該当の章を参照してください。

構文

```
DBMS_REPCAT_INSTANTIATE.INSTANTIATE_OFFLINE_REPAPI (
    refresh_template_name    IN    VARCHAR2,
    site_id                  IN    VARCHAR2    := NULL,
    master                   IN    VARCHAR2    := NULL,
    url                      IN    VARCHAR2    := NULL,
    ssl                      IN    NUMBER      := 0,
    trace_vector             IN    NUMBER      := DBMS_REPCAT_RGT.NO_TRACE_DUMP,
    resultset_threshold      IN    NUMBER      := DBMS_REPCAT_INSTANTIATE.
                                                RESULTSET_THRESHOLD,
    lob_threshold            IN    NUMBER      := DBMS_REPCAT_INSTANTIATE.
                                                LOB_THRESHOLD);
```

表 39-6 INSTANTIATE_OFFLINE_REPAPI ファンクションのパラメータ

パラメータ	説明
refresh_template_name	インスタンス化する配置テンプレートの名前。
site_id	<p>このオフライン・インスタンスエーションに割り当てられた一時名。この一時名は、オフライン・インスタンスエーション・ファイル名の一部です。デフォルトの NULL 値を使用すると、この一時名に対する番号が生成されます。名前の指定は、オフライン・インスタンスエーション・ファイル名で目的のスナップショット・サイトを示す場合に便利です。</p> <p>site_id は一意にする必要があります。つまり、同一の site_id を 2 つのオフライン・インスタンスエーション・ファイルに指定することはできません。</p> <p>この一時名は、配置テンプレートがインスタンス化されるときに、RepAPI クライアントによって常に上書きされます。このパラメータには、デフォルトである NULL 値の使用をお勧めします。</p>
master	RepAPI クライアントがサーバーに使用するオプションの別名。指定した場合、RepAPI クライアントは常にこの別名でサーバーを参照します。
url	データベースにアクセスするためのマスター・サイトの公開 URL。指定した場合、RepAPI クライアントは常にこの URL でサーバーを参照します。
ssl	1 は、スナップショットがマスター・サイトとの通信でセキュア・ソケット・レイヤー (SSL) を使用することを示します。0 は、SSL を使用しないことを示します。
trace_vector	デバッグのトレース・レベル。
resultset_threshold	スナップショットのリフレッシュ処理時に送信する非 LOB 行データの最大サイズ。

表 39-6 INSTANTIATE_OFFLINE_REPAPI ファンクションのパラメータ

パラメータ	説明
lob_threshold	スナップショットのリフレッシュ処理時に送信する LOB 行データの最大サイズ

表 39-7 INSTANTIATE_OFFLINE_REPAPI ファンクションの例外

例外	説明
miss_refresh_template	テンプレートが存在しません。
miss_user	ユーザー名がデータベース内に存在しません。
miss_template_site	ユーザーおよびサイトに対するテンプレートがインスタンス化されていません。

戻り値

表 39-8 INSTANTIATE_OFFLINE_REPAPI ファンクションの戻り値

戻り値	説明
0	エラーが見つかりました。
1	エラーは見つかりませんでした。

INSTANTIATE_ONLINE ファンクション

このファンクションは、マスター・サイトでスクリプトを生成します。このスクリプトは、オンライン時にリモート・スナップショット・サイトでスナップショット環境を作成するために使用します。リモート・スナップショット・サイトでのインスタンス化プロセスは長くなる場合があるため（必要な時間は新しいスナップショットに移入されるデータの量によって異なります）、生成されたスクリプトは、マスター・サイトに長時間接続したままにできるリモート・スナップショット・サイトで使用してください。

このファンクションで生成されたスクリプトは、USER_REPCAT_TEMP_OUTPUT 一時ビューに格納され、配置テンプレートの配布時に、Replication Manager などの Oracle Tools で使用されます。このファンクションで戻される数値は、USER_REPCAT_TEMP_OUTPUT ビューから該当する情報を取り出すために使用されます。

このパブリック・ファンクションを実行するユーザーは、指定したサイトでインスタンス化されたテンプレートの登録ユーザーになります。

構文

```
DBMS_REPCAT_INSTANTIATE.INSTANTIATE_ONLINE(  
    refresh_template_name  IN   VARCHAR2,  
    site_name              IN   VARCHAR2,  
    runtime_parm_id        IN   NUMBER      := -1e-130,  
    next_date              IN   DATE         := SYSDATE,  
    interval               IN   VARCHAR2    := 'SYSDATE + 1',  
    use_default_gowner     IN   BOOLEAN     := TRUE)  
    return NUMBER;
```

表 39-9 INSTANTIATE_ONLINE ファンクションのパラメータ

パラメータ	説明
refresh_template_name	インスタンス化する配置テンプレートの名前。
site_name	配置テンプレートをインスタンス化するリモート・サイトの名前。
runtime_parm_id	INSERT_RUNTIME_PARMS プロシージャを使用してランタイム・パラメータ値を定義している場合は、ランタイム・パラメータの作成時に使用した ID (GET_RUNTIME_PARM_ID ファンクションを使用して取り出された ID です) を指定します。
next_date	リフレッシュ・グループの作成時に使用される次回リフレッシュ日付の値を指定します。
interval	リフレッシュ・グループの作成時に使用されるリフレッシュ間隔を指定します。
use_default_gowner	TRUE の場合、作成されたすべてのスナップショット・オブジェクト・グループはデフォルト・ユーザー PUBLIC が所有します。 FALSE の場合、作成されたすべてのスナップショット・オブジェクト・グループはインスタンス化を実施したユーザーが所有します。

表 39-10 INSTANTIATE_ONLINE ファンクションの例外

例外	説明
miss_refresh_template	指定した配置テンプレート名が無効か、または存在しません。
dupl_template_site	配置テンプレートは、スナップショット・サイトですすでにインスタンス化されています。配置テンプレート特定のスナップショット・サイトでインスタンス化できる回数は 1 回のみです
not_authorized	ユーザーがインスタンス化しようとした配置テンプレートは、インスタンス化を認可されていません。

戻り値

表 39-11 INSTANTIATE_ONLINE ファンクションの戻り値

戻り値	説明
< システム生成番号 >	生成したインスタンスエーション・スクリプトを USER_REPCAT_TEMP_OUTPUT ビューで取り出すように選択したとき、output_id に対して生成されるシステム番号を指定します。

DBMS_REPCAT_RGT

DBMS_REPCAT_RGT は、リフレッシュ・グループ・テンプレートのメンテナンスと定義を制御します。

DBMS_REPCAT_RGT パッケージ

サブプログラムの要約

表 40-1 DBMS_REPCAT_RGT パッケージのサブプログラム

サブプログラム	説明
40-5 ページの ALTER_REFRESH_TEMPLATE プロシ ージャ	DBA が既存の配置テンプレートを変更できます。
40-6 ページの ALTER_TEMPLATE_OBJECT プロシ ージャ	指定した配置テンプレートに追加されているオブジェクト を変更します。
40-9 ページの ALTER_TEMPLATE_PARM プロシージャ	DBA が特定の配置テンプレートのパラメータを変更でき ます。
40-11 ページの ALTER_USER_AUTHORIZATION プロ シージャ	DBA_REPCAT_USER_AUTHORIZATIONS ビューの内容を 変更します。
40-12 ページの ALTER_USER_PARM_VALUE プロシ ージャ	特定のユーザーに対して定義されている既存のパラメータ 値を変更します。
40-15 ページの COMPARE_TEMPLATES ファンクション	DBA が 2 つの配置テンプレートの内容を比較できます。
40-16 ページの COPY_TEMPLATE ファンクション	DBA が配置テンプレートをコピーできます。
40-18 ページの CREATE_OBJECT_FROM_EXISTING フ ァンクション	既存のデータベース・オブジェクトからテンプレート・オ ブジェクト定義を作成し、それをターゲット配置テンプ レートに追加します。
40-20 ページの CREATE_REFRESH_TEMPLATE ファンク ション	配置テンプレートを作成します。DBA がテンプレート名、 プライベートまたはパブリック・ステータス、およびター ゲット・リフレッシュ・グループを定義できます。
40-22 ページの CREATE_TEMPLATE_OBJECT ファンク ション	ターゲット配置テンプレートのコンテナにオブジェクト定 義を追加します。
40-26 ページの CREATE_TEMPLATE_PARM ファンクシ ョン	特定の配置テンプレートのパラメータを作成します。この 結果、リモート・スナップショット・サイトでカスタム・ データ・セットを作成できます。

表 40-1 DBMS_REPCAT_RGT パッケージのサブプログラム

サブプログラム	説明
40-28 ページの CREATE_USER_AUTHORIZATION ファンクション	特定のユーザーに、プライベート配置テンプレートのインスタンス化を認可します。
40-29 ページの CREATE_USER_PARM_VALUE ファンクション	特定のユーザーに対する配置テンプレートのパラメータ値を事前定義します。
40-32 ページの DELETE_RUNTIME_PARMS プロシージャ	INSERT_RUNTIME_PARMS プロシージャを使用して定義したランタイム・パラメータ値を削除します。
40-33 ページの DROP_ALL_OBJECTS プロシージャ	DBA が、配置テンプレートからすべてのまたは特定のオブジェクト型を削除できます。
40-34 ページの DROP_ALL_TEMPLATE_PARMS プロシージャ	DBA が、指定した配置テンプレートのテンプレート・パラメータを削除できます。
40-35 ページの DROP_ALL_TEMPLATE_SITES プロシージャ	DBA_REPCAT_TEMPLATE_SITES ビューからすべてのエントリを削除します。
40-36 ページの DROP_ALL_TEMPLATES プロシージャ	プロシージャがコールされるサイトの配置テンプレートをすべて削除します。
40-36 ページの DROP_ALL_USER_AUTHORIZATIONS プロシージャ	DBA が、指定した配置テンプレートに対するユーザー認証をすべて削除できます。
40-37 ページの DROP_ALL_USER_PARM_VALUES プロシージャ	特定の配置テンプレートに対するユーザー・パラメータ値を削除します。
40-39 ページの DROP_REFRESH_TEMPLATE プロシージャ	配置テンプレートを削除します。
40-40 ページの DROP_SITE_INSTANTIATION プロシージャ	DBA_REPCAT_TEMPLATE_SITES ビューからターゲット・サイトを削除します。
40-41 ページの DROP_TEMPLATE_OBJECT プロシージャ	特定の配置テンプレートからテンプレート・オブジェクトを削除します。
40-43 ページの DROP_TEMPLATE_PARM プロシージャ	DBA_REPCAT_TEMPLATE_PARMS ビューから既存のテンプレート・パラメータを削除します。

表 40-1 DBMS_REPCAT_RGT パッケージのサブプログラム

サブプログラム	説明
40-44 ページの DROP_USER_AUTHORIZATION プロシージャ	DBA_REPCAT_USER_AUTHORIZATIONS ビューからユーザー認証エントリを削除します。
40-45 ページの DROP_USER_PARM_VALUE プロシージャ	特定の配置テンプレートに対する事前定義ユーザー・パラメータ値を削除します。
40-46 ページの GET_RUNTIME_PARM_ID ファンクション	ランタイム・パラメータ値の定義時に使用する ID を取り出します。
40-47 ページの INSERT_RUNTIME_PARS プロシージャ	テンプレートのインスタンス化前にランタイム・パラメータ値を定義します。
40-49 ページの INSTANTIATE_OFFLINE ファンクション	マスター・サイトでスクリプトを生成します。このスクリプトは、オフライン時にリモート・スナップショット・サイトでスナップショット環境を作成するために使用します。
40-51 ページの INSTANTIATE_OFFLINE_REPAPI ファンクション	マスター・サイトでバイナリ・ファイルを生成します。このファイルは、オフライン時にリモートの RepAPI スナップショット・サイトでスナップショット環境を作成するために使用します。
40-54 ページの INSTANTIATE_ONLINE ファンクション	マスター・サイトでスクリプトを生成します。このスクリプトは、オンライン時にリモート・スナップショット・サイトでスナップショット環境を作成するために使用します。
40-57 ページの LOCK_TEMPLATE_EXCLUSIVE プロシージャ	配置テンプレートの更新中または変更中に、ユーザーがテンプレートの読み込みまたはインスタンス化を実行できないようにします。
40-57 ページの LOCK_TEMPLATE_SHARED プロシージャ	指定した配置テンプレートを読み取り専用にします。

ALTER_REFRESH_TEMPLATE プロシージャ

このプロシージャでは、DBA が既存の配置テンプレートを変更できます。新規配置テンプレート名、新規リフレッシュ・グループまたは新規所有者の定義や、パブリック / プライベート・ステータスの変更などを実行できます。

構文

```
DBMS_REPCAT_RGT.ALTER_REFRESH_TEMPLATE (
    refresh_template_name      IN   VARCHAR2,
    new_owner                  IN   VARCHAR2 := '-',
    new_refresh_group_name     IN   VARCHAR2 := '-',
    new_refresh_template_name  IN   VARCHAR2 := '-',
    new_template_comment       IN   VARCHAR2 := '-',
    new_public_template        IN   VARCHAR2 := '-',
    new_last_modified          IN   DATE := to_date('1', 'J'),
    new_modified_by            IN   NUMBER := -1e-130);
```

パラメータ

表 40-2 ALTER_REFRESH_TEMPLATE プロシージャのパラメータ

パラメータ	説明
refresh_template_name	変更する配置テンプレートの名前。
new_owner	配置テンプレートの新しい所有者名。現行の所有者を変更しないときは値を指定しないでください。
new_refresh_group_name	必要な場合は、このパラメータを使用してテンプレート・オブジェクトが追加される新規リフレッシュ・グループの名前を指定します。現行のリフレッシュ・グループを変更しないときは値を指定しないでください。
new_refresh_template_name	新しい配置テンプレート名を指定します。現行の配置テンプレート名を変更しないときは値を指定しないでください。
new_template_comment	配置テンプレートの新しいコメント。現行のテンプレート・コメントを変更しないときは値を指定しないでください。
new_public_template	配置テンプレートがパブリックかプライベートかを決定します。指定できる値は、'Y' と 'N' ('Y' = パブリック、'N' = プライベート) のみです。現行の値を変更しないときは値を指定しないでください。
new_last_modified	この配置テンプレートの最終更新日付。値を指定しないと、現行の日付が自動的に使用されます。
new_modified_by	この配置テンプレートの最終更新ユーザーの名前。値を指定しないと、カレント・ユーザーが自動的に使用されます。

例外

表 40-3 ALTER_REFRESH_TEMPLATE プロシージャの例外

例外	説明
miss_refresh_template	指定した配置テンプレート名が無効か、または存在しません。
bad_public_template	public_template パラメータの指定に誤りがあります。public_template パラメータには、'Y'（パブリック・テンプレート）または 'N'（プライベート・テンプレート）のいずれかを指定してください。
dupl_refresh_template	指定した名前のテンプレートは、すでに存在しています。DBA_REPCAT_REFRESH_TEMPLATES ビューを参照してください。

ALTER_TEMPLATE_OBJECT プロシージャ

このプロシージャは、指定した配置テンプレートに追加されているオブジェクトを変更します。最も一般的な変更は、オブジェクト DDL の変更および異なる配置テンプレートへのオブジェクトの割当てです。

テンプレートに対する変更内容は、配置テンプレートをインスタンス化する新規サイトにのみ反映されます。テンプレートをすでにインスタンス化しているリモート・サイトは、配置テンプレートを再インスタンス化して変更内容を適用する必要があります。

構文

```
DBMS_REPCAT_RGT.ALTER_TEMPLATE_OBJECT (  
  refresh_template_name      IN   VARCHAR2,  
  object_name                IN   VARCHAR2,  
  object_type                IN   VARCHAR2,  
  new_refresh_template_name  IN   VARCHAR2 := '-',  
  new_object_name            IN   VARCHAR2 := '-',  
  new_object_type            IN   VARCHAR2 := '-',  
  new_ddl_text               IN   CLOB    := '-',  
  new_master_rollback_seg    IN   VARCHAR2 := '-',  
  new_flavor_id              IN   NUMBER := -1e-130);
```

パラメータ

表 40-4 ALTER_TEMPLATE_OBJECT プロシージャのパラメータ

パラメータ	説明
refresh_template_name	変更するオブジェクトを含んだ配置テンプレートの名前。
object_name	変更するテンプレート・オブジェクトの名前。
object_type	変更するオブジェクトの型。
new_refresh_template_name	このオブジェクトを再割当てする新しい配置テンプレートの名前。オブジェクトを現行の配置テンプレートに割り当てたままにするときは値を指定しないでください。
new_object_name	テンプレート・オブジェクトの新しい名前。現行のオブジェクト名を変更しないときは値を指定しないでください。
new_object_type	指定済の場合は、新しいオブジェクト型を指定します。次の型のオブジェクトを指定できます。 SNAPSHOT PROCEDURE INDEX FUNCTION TABLE PACKAGE VIEW PACKAGE BODY SYNONYM TRIGGER SEQUENCE DATABASE LINK
new_ddl_text	指定したオブジェクトに対する新しいオブジェクト DDL。現行のオブジェクト DDL を変更しないときは値を指定しないでください。
new_master_rollback_seg	指定したオブジェクトの新しいマスター・ロールバック・セグメント。現行のロールバック・セグメントを変更しないときは値を指定しないでください。
new_flavor_id	内部使用のためのパラメータ。オラクル社カスタマ・サポート・センターから指示がない限り、このパラメータは設定しないでください。

例外

表 40-5 ALTER_TEMPLATE_OBJECT プロシージャの例外

例外	説明
miss_refresh_template	指定した配置テンプレート名が無効か、または存在しません。
miss_flavor_id	この例外が発生した場合、オラクル社カスタマ・サポート・センターに連絡してください。
bad_object_type	オブジェクト型の指定に誤りがあります。有効なオブジェクト型のリストは、表 40-4 を参照してください。
miss_template_object	指定したテンプレート・オブジェクト名が無効か、または存在しません。
dupl_template_object	new_refresh_template_name パラメータに指定した新規テンプレート名は、すでに存在しています。

使用上の注意

ALTER_TEMPLATE_OBJECT プロシージャは CLOB を使用しているため、このプロシージャを使用する場合は、DBMS_LOB パッケージを使用する必要があります。次の例は、ALTER_TEMPLATE_OBJECT プロシージャで DBMS_LOB パッケージを使用する方法を示しています。

```
DECLARE
    tempstring VARCHAR2(100);
    templob CLOB;
BEGIN
    DBMS_LOB.CREATETEMPORARY(templob, TRUE, DBMS_LOB.SESSION);
    tempstring := 'CREATE SNAPSHOT snap_sales AS SELECT *
        FROM sales WHERE salesperson = :salesid and region_id = :region';
    DBMS_LOB.WRITE(templob, length(tempstring), 1, tempstring);
    DBMS_REPCAT_RGT.ALTER_TEMPLATE_OBJECT(
        refresh_template_name => 'rgt_personnel',
        object_name => 'SNAP_SALES',
        object_type => 'SNAPSHOT',
        new_ddl_text => templob);
    DBMS_LOB.FREETEMPORARY(templob);
END;
/
```


ALTER_TEMPLATE_PARM プロシージャ

このプロシージャでは、DBA が特定の配置テンプレートのパラメータを変更できます。パラメータ名の変更およびデフォルト値やプロンプト文字列の再定義などを実行できます。

構文

```
DBMS_REPCAT_RGT.ALTER_TEMPLATE_PARM (  
    refresh_template_name      IN   VARCHAR2,  
    parameter_name             IN   VARCHAR2,  
    new_refresh_template_name  IN   VARCHAR2 := '-',  
    new_parameter_name         IN   VARCHAR2 := '-',  
    new_default_parm_value     IN   CLOB := NULL,  
    new_prompt_string          IN   VARCHAR2 := '-',  
    new_user_override          IN   VARCHAR2 := '-');
```

パラメータ

表 40-6 ALTER_TEMPLATE_PARM プロシージャのパラメータ

パラメータ	説明
refresh_template_name	変更するパラメータを含んだ配置テンプレートの名前。
parameter_name	変更するパラメータの名前。
new_refresh_template_name	指定したパラメータを再割当てる配置テンプレートの名前（あるテンプレートのパラメータを別のテンプレートに移動するときに便利です）。パラメータを現行のテンプレートに割り当てたままにするときは値を指定しないでください。
new_parameter_name	テンプレート・パラメータの新しい名前。現行のパラメータ名を変更しないときは値を指定しないでください。
new_default_parm_value	指定したパラメータの新しいデフォルト値。現行のデフォルト値を変更しないときは値を指定しないでください。
new_prompt_string	指定したパラメータの新しいプロンプト・テキスト。現行のプロンプト文字列を変更しないときは値を指定しないでください。
new_user_override	インスタンス化プロセス時のプロンプトで、ユーザーがデフォルト値を上書きできるかどうかを決定します。このパラメータにユーザー・パラメータ値が定義されていないと、プロンプトが表示されます。デフォルト値の上書きを許可する場合は 'Y' を、許可しない場合は 'N' を設定します。

例外

表 40-7 ALTER_TEMPLATE_PARM プロシージャの例外

例外	説明
miss_refresh_template	指定した配置テンプレート名が無効か、または存在しません。
miss_template_parm	指定したテンプレート・パラメータが無効か、または存在しません。
dupl_template_parm	new_refresh_template_name と new_parameter_name の組合せは、すでに存在しています。

使用上の注意

ALTER_TEMPLATE_PARM プロシージャは CLOB を使用しているため、このプロシージャを使用する場合は、DBMS_LOB パッケージを使用する必要があります。次の例は、ALTER_TEMPLATE_PARM プロシージャで DBMS_LOB パッケージを使用する方法を示しています。

```
DECLARE
    tempstring VARCHAR2(100);
    templob CLOB;
BEGIN
    DBMS_LOB.CREATETEMPORARY(templob, TRUE, DBMS_LOB.SESSION);
    tempstring := 'REGION 20';
    DBMS_LOB.WRITE(templob, length(tempstring), 1, tempstring);
    DBMS_REPCAT_RGT.ALTER_TEMPLATE_PARM(
        refresh_template_name => 'rgt_personnel',
        parameter_name => 'region',
        new_default_parm_value => templob);
    DBMS_LOB.FREETEMPORARY(templob);
END;
/
```

ALTER_USER_AUTHORIZATION プロシージャ

このプロシージャは、DBA_REPCAT_USER_AUTHORIZATIONS ビューの内容を変更します。具体的には、ユーザー・テンプレートまたは配置テンプレートの認可割当てを変更できます。たとえば、このプロシージャは、従業員が異動し、別の配置テンプレートのスナップショット環境が必要な場合に役立ちます。DBA が新規配置テンプレートを従業員に割り当てるだけで、そのユーザーはターゲット・テンプレートのインスタンス化を認可されます。

構文

```
DBMS_REPCAT_RGT.ALTER_USER_AUTHORIZATION (  
    user_name                IN    VARCHAR2,  
    refresh_template_name    IN    VARCHAR2,  
    new_user_name            IN    VARCHAR2 := '-',  
    new_refresh_template_name IN    VARCHAR2 := '-');
```

パラメータ

表 40-8 ALTER_USER_AUTHORIZATION プロシージャのパラメータ

パラメータ	説明
user_name	認可を変更対象とするユーザーの名前。
refresh_template_name	指定したユーザーに現在割り当てられている、変更対象の配置テンプレートの名前。
new_user_name	このパラメータは、このテンプレート認可に対して、新しいユーザーを定義するときに使用します。カレント・ユーザーを変更しないときは値を指定しないでください。
new_refresh_template_name	指定したユーザー（既存のユーザー、または指定した新規ユーザー）がインスタンス化を認可される配置テンプレート。現行の配置テンプレートを変更しないときは値を指定しないでください。

例外

表 40-9 ALTER_USER_AUTHORIZATION プロシージャの例外

例外	説明
miss_user_authorization	指定した user_name と refresh_template_name の組合せが DBA_REPCAT_USER_AUTHORIZATIONS ビューに存在しません。
miss_user	new_user_name または user_name パラメータに指定したユーザー名が無効か、または存在しません。
miss_refresh_template	new_refresh_template パラメータに指定した配置テンプレートが無効か、または存在しません。
dupl_user_authorization	指定したユーザー名と配置テンプレート名に対する行は、すでに存在しています。DBA_REPCAT_USER_AUTHORIZATIONS ビューを参照してください。

ALTER_USER_PARM_VALUE プロシージャ

このプロシージャは、特定のユーザーに対して定義されている既存のパラメータ値を変更します。スナップショット環境で割当て表を使用している場合に特に便利なプロシージャです。ユーザー・パラメータ値を変更することで、リモート・スナップショット・サイトのデータ・セットを安全にすばやく変更できます。

関連項目： 割当て表の使用方法は、『Oracle8i レプリケーション・ガイド』の「配置テンプレートの設計」を参照してください。

構文

```
DBMS_REPCAT_RGT.ALTER_USER_PARM_VALUE(  
  refresh_template_name      IN   VARCHAR2,  
  parameter_name             IN   VARCHAR2,  
  user_name                  IN   VARCHAR2,  
  new_refresh_template_name  IN   VARCHAR2 := '-',  
  new_parameter_name         IN   VARCHAR2 := '-',  
  new_user_name              IN   VARCHAR2 := '-',  
  new_parm_value             IN   CLOB := NULL);
```

パラメータ

表 40-10 ALTER_USER_PARM_VALUE プロシージャのパラメータ

パラメータ	説明
refresh_template_name	変更するユーザー・パラメータ値を含んだ配置テンプレートの名前。
parameter_name	変更するパラメータの名前。
user_name	パラメータ値を変更対象とするユーザーの名前。
new_refresh_template_name	指定したユーザー・パラメータ値を再割当てする配置テンプレートの名前（別のテンプレートに対してユーザーを認可するときに便利です）。パラメータを現行のテンプレートに割り当てたままにするときは値を指定しないでください。
new_parameter_name	新規テンプレート・パラメータ名。既存のパラメータに対して定義されたユーザー値を変更しないときは値を指定しないでください。
new_user_name	このパラメータ値を適用する新規ユーザー名。パラメータをクライアント・ユーザーに割り当てたままにするときは値を指定しないでください。
new_parm_value	指定したユーザー・パラメータの新しいパラメータ値。現行のパラメータ値を変更しないときは値を指定しないでください。

例外

表 40-11 ALTER_USER_PARM_VALUE プロシージャの例外

例外	説明
miss_refresh_template	指定した配置テンプレート名が無効か、または存在しません。
miss_template_parm	指定したテンプレート・パラメータが無効か、または存在しません。
miss_user	user_name または new_user_name パラメータに指定したユーザー名が無効か、または存在しません。
miss_user_parm_values	指定したユーザー・パラメータ値が存在しません。
dupl_user_parm_values	指定した新規ユーザー・パラメータはすでに存在しています。

使用上の注意

ALTER_USER_PARM_VALUE プロシージャは CLOB を使用しているため、このプロシージャを使用する場合は、DBMS_LOB パッケージを使用する必要があります。次の例は、ALTER_USER_PARM_VALUE プロシージャで DBMS_LOB パッケージを使用する方法を示しています。

```
DECLARE
    tempstring VARCHAR2(100);
    templob CLOB;
BEGIN
    DBMS_LOB.CREATETEMPORARY(templob, TRUE, DBMS_LOB.SESSION);
    tempstring := 'REGION 20';
    DBMS_LOB.WRITE(templob, length(tempstring), 1, tempstring);
    DBMS_REPCAT_RGT.ALTER_USER_PARM_VALUE(
        refresh_template_name => 'rgt_personnel',
        parameter_name => 'region',
        user_name => 'BOB',
        new_parm_value => templob);
    DBMS_LOB.FREETEMPORARY(templob);
END;
/
```

COMPARE_TEMPLATES ファンクション

このファンクションでは、DBA が 2 つの配置テンプレートの内容を比較できます。2 つの配置テンプレートの相違点は、USER_REPCAT_TEMP_OUTPUT 表に格納されます。

COMPARE_TEMPLATES ファンクションは数値を返します。この数値は、USER_REPCAT_TEMP_OUTPUT 表の問合せ時に WHERE 句に指定する値です。たとえば、COMPARE_TEMPLATES プロシージャが 10 を戻した場合に、指定した 2 つのテンプレート間の相違点をすべて表示するには、次の SELECT 文を実行します（テンプレートが同じ場合、SELECT 文は行を戻しません）。

```
SELECT text FROM user_repcat_temp_output
      WHERE output_id = 10 ORDER BY LINE;
```

USER_REPCAT_TEMP_OUTPUT の内容は、ユーザーが切断するか、または ROLLBACK が実行されると失われます。

構文

```
DBMS_REPCAT_RGT.COMPARE_TEMPLATES (
    source_template_name    IN    VARCHAR2,
    compare_template_name   IN    VARCHAR2)
return NUMBER;
```

パラメータ

表 40-12 COMPARE_TEMPLATES ファンクションのパラメータ

パラメータ	説明
source_template_name	比較対象の最初の配置テンプレートの名前
compare_template_name	比較対象の 2 番目の配置テンプレートの名前

例外

表 40-13 COMPARE_TEMPLATES ファンクションの例外

例外	説明
miss_refresh_template	比較対象の配置テンプレート名が無効か、または存在しません。

戻り値

表 40-14 COMPARE_TEMPLATES ファンクションの戻り値

戻り値	説明
<システム生成番号>	比較したテンプレート間の相違点を USER_REPCAT_TEMP_OUTPUT ビューで表示するように選択したときに、output_id の値に戻される番号を指定します。

COPY_TEMPLATE ファンクション

このファンクションによって、DBA は配置テンプレートをコピーできます。このファンクションは、新しく作成する配置テンプレートで既存の配置テンプレートに含まれているオブジェクトを多数使用する場合に便利です。さらに、このファンクションは、配置テンプレート、テンプレート・オブジェクト、テンプレート・パラメータおよびユーザー・パラメータ値をコピーします。DBA は、オプションでこのテンプレートに対するユーザー認証をコピーできます。このファンクションで戻される番号は、配置テンプレートを管理するために Oracle で内部的に使用されます。

注意： DBA_REPCAT_TEMPLATE_SITES ビュー内の値はコピーされません。

このファンクションでは、配置テンプレートを別のマスター・サイトにコピーすることもできます。この機能は、配置テンプレートの配布、および複数サイト間でのネットワーク負荷の分散に役立ちます。

構文

```
DBMS_REPCAT_RGT.COPY_TEMPLATE (
  old_refresh_template_name    IN   VARCHAR2,
  new_refresh_template_name    IN   VARCHAR2,
  copy_user_authorizations     IN   VARCHAR2,
  dblink                       IN   VARCHAR2 := NULL)
return NUMBER;
```


パラメータ

表 40-15 COPY_TEMPLATE ファンクションのパラメータ

パラメータ	説明
old_refresh_template_name	コピー元の配置テンプレートの名前。
new_refresh_template_name	新規配置テンプレートの名前。
copy_user_authorizations	コピー元のテンプレートのテンプレート認可を新しい配置テンプレート用にコピーするかどうかを指定します。有効な値は、'Y'、'N' および NULL です。 注意： すべてのユーザーがターゲット・データベースに存在している必要があります。
dblink	オプションで、配置テンプレートのコピー元を定義します（この機能は、他のマスター・サイトに配置テンプレートを配布するときに便利です）。指定しないと、配置テンプレートはローカル・マスター・サイトからコピーされます。

例外

表 40-16 COPY_TEMPLATE ファンクションの例外

例外	説明
miss_refresh_template	コピー元の配置テンプレート名が無効か、または存在しません。
dupl_refresh_template	指定した新規リフレッシュ・テンプレートの名前は、すでに存在しています。
bad_copy_auth	copy_user_authorization パラメータに指定した値が無効です。有効な値は、'Y'、'N' および NULL です。

戻り値

表 40-17 COPY_TEMPLATE ファンクションの戻り値

戻り値	説明
< システム生成番号 >	Oracle で内部的に使用されるシステム生成番号

CREATE_OBJECT_FROM_EXISTING ファンクション

このファンクションは、既存のデータベース・オブジェクトからテンプレート・オブジェクト定義を作成し、それをターゲット配置テンプレートに追加します。ターゲット配置テンプレートがリモート・スナップショット・サイトでインスタンス化されるときに、元のデータベース・オブジェクトを作成したオブジェクト DDL が実行されます。これは、既存のトリガーやプロシージャをテンプレートに追加するときに便利な方法です。このファンクションで戻される番号は、配置テンプレートを管理するために Oracle で内部的に使用されます。

構文

```
DBMS_REPCAT_RGT.CREATE_OBJECT_FROM_EXISTING(  
    refresh_template_name IN  VARCHAR2,  
    object_name           IN   VARCHAR2,  
    sname                 IN   VARCHAR2,  
    oname                 IN   VARCHAR2,  
    otype                 IN   VARCHAR2)  
return NUMBER;
```

パラメータ

表 40-18 CREATE_OBJECT_FROM_EXISTING ファンクションのパラメータ

パラメータ	説明										
refresh_template_name	このオブジェクトを追加する配置テンプレートの名前。										
object_name	オプションで、配置テンプレートに追加する既存オブジェクトの新規名を指定します（既存のオブジェクトに新しい名前を定義できます）。										
sname	テンプレート・オブジェクトの作成元のオブジェクトを含んだスキーマ。										
oname	テンプレート・オブジェクトの作成元のオブジェクト名。										
otype	テンプレートに追加するデータベース・オブジェクトの型（PROCEDURE、TRIGGER など）。オブジェクト型は、次の数値型識別子を使用して指定する必要があります（DATABASE LINK または SNAPSHOT は、このファンクションでは有効なオブジェクト型ではありません）。 <table><tr><td>SEQUENCE</td><td>PROCEDURE</td></tr><tr><td>INDEX</td><td>FUNCTION</td></tr><tr><td>TABLE</td><td>PACKAGE</td></tr><tr><td>VIEW</td><td>PACKAGE BODY</td></tr><tr><td>SYNONYM</td><td>TRIGGER</td></tr></table>	SEQUENCE	PROCEDURE	INDEX	FUNCTION	TABLE	PACKAGE	VIEW	PACKAGE BODY	SYNONYM	TRIGGER
SEQUENCE	PROCEDURE										
INDEX	FUNCTION										
TABLE	PACKAGE										
VIEW	PACKAGE BODY										
SYNONYM	TRIGGER										

例外

表 40-19 CREATE_OBJECT_FROM_EXISTING ファンクションの例外

例外	説明
miss_refresh_template	指定したリフレッシュ・テンプレート名が無効か、または存在しません。既存の配置テンプレートのリストは、DBA_REPCAT_REFRESH_TEMPLATES ビューを参照してください。
bad_object_type	オブジェクト型の指定に誤りがあります。
dupl_template_object	同じ名前と型のオブジェクトが、指定した配置テンプレートにすでに追加されています。
objectmissing	指定したオブジェクトが存在しません。

戻り値

表 40-20 CREATE_OBJECT_FROM_EXISTING ファンクションの戻り値

戻り値	説明
<システム生成番号>	Oracle で内部的に使用されるシステム生成番号

CREATE_REFRESH_TEMPLATE ファンクション

このファンクションは、配置テンプレートを作成します。テンプレート名、プライベートまたはパブリック・ステータス、およびターゲット・リフレッシュ・グループを定義できます。テンプレート・オブジェクト、ユーザー認証またはテンプレート・パラメータを作成するたびに、このファンクションで作成された配置テンプレートを参照します。このファンクションは、DBA_REPCAT_REFRESH_TEMPLATES ビューに行を追加します。このファンクションで戻される番号は、配置テンプレートを管理するために Oracle で内部的に使用されます。

構文

```
DBMS_REPCAT_RGT.CREATE_REFRESH_TEMPLATE (  
  owner                IN   VARCHAR2,  
  refresh_group_name   IN   VARCHAR2,  
  refresh_template_name IN   VARCHAR2,  
  template_comment     IN   VARCHAR2 := NULL,  
  public_template      IN   VARCHAR2 := NULL,  
  last_modified        IN   DATE := SYSDATE,  
  modified_by          IN   VARCHAR2 := USER,  
  creation_date        IN   DATE := SYSDATE,  
  created_by           IN   VARCHAR2 := USER)  
return NUMBER;
```

パラメータ

表 40-21 CREATE_REFRESH_TEMPLATE ファンクションのパラメータ

パラメータ	説明
owner	配置テンプレート所有者のユーザー名。指定しない場合、テンプレートの作成ユーザーが自動的に使用されます。
refresh_group_name	このテンプレートのインスタンス化時に作成されるリフレッシュ・グループの名前。このテンプレートで作成されたオブジェクトはすべて、指定したリフレッシュ・グループに割り当てられます。
refresh_template_name	作成する配置テンプレートの名前。この名前は、この配置テンプレートを含むすべてのアクティビティで参照されます。
template_comment	このパラメータで定義したユーザー・コメントは、DBA_REPCAT_REFRESH_TEMPLATES ビューに表示されます。
public_template	配置テンプレートがパブリックかプライベートかを指定します。指定できる値は 'Y' と 'N' ('Y' = パブリック、'N' = プライベート) のみです。
last_modified	この配置テンプレートの最終更新日付。値を指定しないと、現行の日付が自動的に使用されます。
modified_by	この配置テンプレートの最終更新ユーザーの名前。値を指定しないと、カレント・ユーザーが自動的に使用されます。
creation_date	この配置テンプレートの作成日付。値を指定しないと、現行の日付が自動的に使用されます。
created_by	この配置テンプレートの作成ユーザーの名前。値を指定しないと、カレント・ユーザーが自動的に使用されます。

例外

表 40-22 CREATE_REFRESH_TEMPLATE ファンクションの例外

例外	説明
dupl_refresh_template	指定した名前のテンプレートは、すでに存在しています。既存のテンプレートのリストは、DBA_REPCAT_REFRESH_TEMPLATES ビューを参照してください。
bad_public_template	public_template パラメータの指定に誤りがあります。public_template パラメータには、'Y' (パブリック・テンプレート) または 'N' (プライベート・テンプレート) のいずれかを指定してください。

戻り値

表 40-23 CREATE_REFRESH_TEMPLATE ファンクションの戻り値

戻り値	説明
<システム生成番号>	Oracle で内部的に使用されるシステム生成番号

CREATE_TEMPLATE_OBJECT ファンクション

このファンクションは、ターゲット配置テンプレートのコンテナにオブジェクト定義を追加します。指定したオブジェクト DDL は、ターゲット配置テンプレートがリモート・スナップショット・サイトでインスタンス化されるときに実行されます。スナップショット以外に、表、プロシージャおよびその他のオブジェクトをテンプレートに追加できます。このファンクションで戻される番号は、配置テンプレートを管理するために Oracle で内部的に使用されます。

構文

```
DBMS_REPCAT_RGT.CREATE_TEMPLATE_OBJECT (  
  refresh_template_name IN  VARCHAR2,  
  object_name           IN   VARCHAR2,  
  object_type           IN   VARCHAR2,  
  ddl_text              IN   CLOB,  
  master_rollback_seg   IN   VARCHAR2 := NULL,  
  flavor_id             IN    NUMBER := -1e-130)  
return NUMBER;
```

パラメータ

表 40-24 CREATE_TEMPLATE_OBJECT ファンクションのパラメータ

パラメータ	説明
refresh_template_name	このオブジェクトを追加する配置テンプレートの名前。
object_name	作成するテンプレート・オブジェクトの名前。
object_type	テンプレートに追加するデータベース・オブジェクトの型 (SNAPSHOT、TRIGGER、PROCEDURE など)。次の型のオブジェクトを指定できます。 <div>SNAPSHOTPROCEDURE</div> <div>INDEXFUNCTION</div> <div>TABLEPACKAGE</div> <div>VIEWPACKAGE BODY</div> <div>SYNONYM</div> <div>MATERIALIZED VIEW</div> <div>SEQUENCE</div> <div>DATABASE LINK</div> <div>TRIGGER</div>
ddl_text	テンプレートに追加するオブジェクトを作成する DDL。DDL は必ずセミコロンで終了してください。テンプレート・オブジェクトのテンプレート・パラメータの作成にはコロン (:) を使用できます。詳細は、『Oracle8i レプリケーション・マネージメント API リファレンス』を参照してください。 <div>CREATE SNAPSHOT 文でスナップショットを追加するときは、必ずスナップショットの問合せでマスター表所有者のスキーマ名を指定してください。</div>
master_rollback_seg	定義済のオブジェクト DDL をリモート・スナップショット・サイトで実行するときに使用するロールバック・セグメントの名前。
flavor_id	内部使用のためのパラメータ。オラクル社カスタマ・サポート・センターから指示がない限り、このパラメータは設定しないでください。

例外

表 40-25 CREATE_TEMPLATE_OBJECT ファンクションの例外

例外	説明
miss_refresh_template	指定したリフレッシュ・テンプレート名が無効か、または存在しません。既存の配置テンプレートのリストは、DBA_REPCAT_REFRESH_TEMPLATES ビューを参照してください。
bad_object_type	オブジェクト型の指定に誤りがあります。有効なオブジェクト型のリストは、表 40-24 を参照してください。
dupl_template_object	同じ名前と型のオブジェクトが、指定した配置テンプレートにすでに追加されています。

戻り値

表 40-26 CREATE_TEMPLATE_OBJECT ファンクションの戻り値

戻り値	説明
< システム生成番号 >	Oracle で内部的に使用されるシステム生成番号

使用上の注意

CREATE_TEMPLATE_OBJECT は CLOB を使用しているため、このファンクションを使用する場合は、DBMS_LOB パッケージを使用する必要があります。次の例は、CREATE_TEMPLATE_OBJECT ファンクションで DBMS_LOB パッケージを使用する方法を示しています。

```
DECLARE
    tempstring VARCHAR2(100);
    templob CLOB;
    a NUMBER;
BEGIN
    DBMS_LOB.CREATETEMPORARY(templob, TRUE, DBMS_LOB.SESSION);
    tempstring := 'CREATE SNAPSHOT snap_sales AS SELECT *
        FROM sales WHERE salesperson = :salesid';
    DBMS_LOB.WRITE(templob, length(tempstring), 1, tempstring);
    a := DBMS_REPCAT_RGT.CREATE_TEMPLATE_OBJECT(
        refresh_template_name => 'rgt_personnel',
        object_name => 'snap_sales',
        object_type => 'SNAPSHOT',
        ddl_text => templob,
        master_rollback_seg => 'RBS');
    DBMS_LOB.FREETEMPORARY(templob);
END;
/
```

CREATE_TEMPLATE_PARM ファンクション

このファンクションは、特定の配置テンプレートのパラメータを作成します。この結果、リモート・スナップショット・サイトでカスタム・データ・セットを作成できます。このファンクションは、DBA がテンプレート・オブジェクトの追加前に一連のテンプレート変数を定義するときのみ必要です。CREATE_TEMPLATE_OBJECT ファンクションを使用して、オブジェクトがテンプレートに追加されると、オブジェクト DDL 内の変数は DBA_REPCAT_TEMPLATE_PARS ビューに自動的に追加されます。

DBA は通常、ALTER_TEMPLATE_PARM ファンクションを使用して、デフォルトのパラメータ値やプロンプト文字列を変更します（詳細は、40-9 ページの「ALTER_TEMPLATE_PARM プロシージャ」を参照してください）。このファンクションで戻される番号は、配置テンプレートを管理するために Oracle で内部的に使用されます。

構文

```
DBMS_REPCAT_RGT.CREATE_TEMPLATE_PARM (  
    refresh_template_name  IN   VARCHAR2,  
    parameter_name         IN   VARCHAR2,  
    default_parm_value     IN   CLOB := NULL,  
    prompt_string          IN   VARCHAR2 := NULL,  
    user_override          IN   VARCHAR2 := NULL)  
return NUMBER;
```

パラメータ

表 40-27 CREATE_TEMPLATE_PARM ファンクションのパラメータ

パラメータ	説明
refresh_template_name	パラメータを作成する配置テンプレートの名前。
parameter_name	作成するパラメータの名前。
default_parm_value	作成するパラメータのデフォルト値。ユーザー・パラメータ値またはランタイム・パラメータ値が存在しない場合は、インスタンス化プロセス時にこのデフォルト値が使用されます。
prompt_string	インスタンス化プロセス時にこのテンプレート・パラメータに対して表示される説明的なプロンプト・テキスト。
user_override	インスタンス化プロセス時のプロンプトで、ユーザーがデフォルト値を上書きできるかどうかを決定します。このパラメータにユーザー・パラメータ値が定義されていないと、プロンプトが表示されます。デフォルト値の上書きを許可する場合は 'Y'、許可しない場合は 'N' を設定します。

例外

表 40-28 CREATE_TEMPLATE_PARM ファンクションの例外

例外	説明
miss_refresh_template	指定したリフレッシュ・テンプレート名が無効か、または存在しません。
dupl_template_parm	同じ名前のパラメータが、指定した配置テンプレートにすでに定義されています。

戻り値

表 40-29 CREATE_TEMPLATE_PARM ファンクションの戻り値

戻り値	説明
<システム生成番号>	Oracle で内部的に使用されるシステム生成番号

使用上の注意

CREATE_TEMPLATE_PARM ファンクションは CLOB を使用しているため、このファンクションを使用する場合は、DBMS_LOB パッケージを使用する必要があります。次の例は、CREATE_TEMPLATE_PARM ファンクションで DBMS_LOB パッケージを使用する方法を示しています。

```
DECLARE
    tempstring VARCHAR2(100);
    templob CLOB;
    a NUMBER;
BEGIN
    DBMS_LOB.CREATETEMPORARY(templob, TRUE, DBMS_LOB.SESSION);
    tempstring := 'REGION 20';
    DBMS_LOB.WRITE(templob, length(tempstring), 1, tempstring);
    a := DBMS_REPCAT_RGT.CREATE_TEMPLATE_PARM(
        refresh_template_name => 'rgt_personnel',
        parameter_name => 'region',
        default_parm_value => templob,
        prompt_string => 'Enter your region ID:',
        user_override => 'Y');
    DBMS_LOB.FREETEMPORARY(templob);
END;
/
```

CREATE_USER_AUTHORIZATION ファンクション

このファンクションは、特定のユーザーに、プライベート配置テンプレートのインスタンス化を認可します。プライベート配置テンプレートに対する認可を付与されていないユーザーは、プライベート・テンプレートをインスタンス化できません。このファンクションは、DBA_REPCAT_USER_AUTHORIZATIONS ビューに行を追加します。

ユーザーを認可する前に、配置テンプレートをインスタンス化するマスター・サイトに、そのユーザーが存在していることを検証してください。このファンクションで戻される番号は、配置テンプレートを管理するために Oracle で内部的に使用されます。

構文

```
DBMS_REPCAT_RGT.CREATE_USER_AUTHORIZATION (  
    user_name                IN    VARCHAR2,  
    refresh_template_name    IN    VARCHAR2)  
return NUMBER;
```

パラメータ

表 40-30 CREATE_USER_AUTHORIZATION ファンクションのパラメータ

パラメータ	説明
user_name	指定したテンプレートのインスタンス化を認可するユーザーの名前。複数のユーザーを指定するときは、ユーザー名をカンマで区切ります (例: 'john, mike, bob')。
refresh_template_name	指定したユーザーにインスタンス化を認可するテンプレートの名前。

例外

表 40-31 CREATE_USER_AUTHORIZATION ファンクションの例外

例外	説明
miss_user	指定したユーザー名が無効か、または存在しません。
miss_refresh_template	指定したリフレッシュ・テンプレート名が無効か、または存在しません。
dupl_user_authorization	指定したユーザーと配置テンプレートに対する認可はすでに作成されています。テンプレート認可のリストは、DBA_REPCAT_USER_AUTHORIZATIONS ビューを参照してください。

戻り値

表 40-32 CREATE_USER_AUTHORIZATION ファンクションの戻り値

戻り値	説明
<システム生成番号>	Oracle で内部的に使用されるシステム生成番号

CREATE_USER_PARM_VALUE ファンクション

このファンクションは、特定のユーザーに対する配置テンプレートのパラメータ値を事前定義します。たとえば、ユーザー 33456 のリージョン・パラメータを WEST に再定義する場合などに、このファンクションを使用します。

このファンクションで指定した値は、テンプレート・パラメータに対して指定したデフォルト値よりも優先されます。このファンクションで戻される番号は、配置テンプレートを管理するために Oracle で内部的に使用されます。

構文

```
DBMS_REPCAT_RGT.CREATE_USER_PARM_VALUE (
    refresh_template_name    IN    VARCHAR2,
    parameter_name           IN    VARCHAR2,
    user_name                 IN    VARCHAR2,
    parm_value                IN    CLOB := NULL)
return NUMBER;
```

パラメータ

表 40-33 CREATE_USER_PARM_VALUE ファンクションのパラメータ

パラメータ	説明
refresh_template_name	ユーザー・パラメータ値を作成するパラメータを含んだ配置テンプレートの名前。
parameter_name	ユーザー・パラメータ値を定義するテンプレート・パラメータの名前。
user_name	ユーザー・パラメータ値を事前定義するユーザーの名前。
parm_value	事前定義パラメータ値。指定したユーザーが開始したインスタンス化プロセス時に使用されます。

例外

表 40-34 CREATE_USER_PARM_VALUE ファンクションの例外

例外	説明
miss_refresh_template	指定した配置テンプレート名が無効か、または存在しません。
dupl_user_parm_values	指定したユーザー、パラメータおよび配置テンプレートに対するパラメータ値はすでに定義されています。既存のユーザー・パラメータ値のリストは、DBA_REPCAT_USER_PARM_VALUES ビューを参照してください。
miss_template_parm	指定した配置テンプレート・パラメータ名が無効か、または存在しません。
miss_user	指定したユーザー名が無効か、または存在しません。

戻り値

表 40-35 CREATE_USER_PARM_VALUE ファンクションの戻り値

戻り値	説明
< システム生成番号 >	Oracle で内部的に使用されるシステム生成番号

使用上の注意

CREATE_USER_PARM_VALUE ファンクションは CLOB を使用しているため、このファンクションを使用する場合は、DBMS_LOB パッケージを使用する必要があります。次の例は、CREATE_USER_PARM_VALUE ファンクションで DBMS_LOB パッケージを使用する方法を示しています。

```
DECLARE
    tempstring VARCHAR2(100);
    templob CLOB;
    a NUMBER;
BEGIN
    DBMS_LOB.CREATETEMPORARY(templob, TRUE, DBMS_LOB.SESSION);
    tempstring := 'REGION 20';
    DBMS_LOB.WRITE(templob, length(tempstring), 1, tempstring);
    a := DBMS_REPCAT_RGT.CREATE_USER_PARM_VALUE(
        refresh_template_name => 'rgt_personnel',
        parameter_name => 'region',
        user_name => 'BOB',
        user_parm_value => templob);
    DBMS_LOB.FREETEMPORARY(templob);
END;
/
```

DELETE_RUNTIME_PARS プロシージャ

このプロシージャは、配置テンプレートをインスタンス化する前に、INSERT_RUNTIME_PARS プロシージャを使用して定義したランタイム・パラメータ値を削除するために使用します。

構文

```
DBMS_REPCAT_RGT.DELETE_RUNTIME_PARS (
    runtime_parm_id      IN   NUMBER,
    parameter_name       IN   VARCHAR2);
```

パラメータ

表 40-36 DELETE_RUNTIME_PARS プロシージャのパラメータ

パラメータ	説明
runtime_parm_id	以前にランタイム・パラメータ値に割り当てた ID (GET_RUNTIME_PARM_ID ファンクションを使用して取り出された値です) を指定します。
parameter_name	削除するパラメータ値の名前 (配置テンプレート・パラメータのリストは、DBA_REPCAT_TEMPLATE_PARS を参照してください) を指定します。

例外

表 40-37 DELETE_RUNTIME_PARS プロシージャの例外

例外	説明
miss_template_parm	指定した配置テンプレート・パラメータ名が無効か、または存在しません。

DROP_ALL_OBJECTS プロシージャ

このプロシージャでは、DBA が配置テンプレートからすべてのオブジェクトまたは特定のオブジェクト型を削除できます。

注意： このプロシージャは、元に戻すことのできない危険度の高いプロシージャです。

構文

```
DBMS_REPCAT_RGT.DROP_ALL_OBJECTS (  
    refresh_template_name  IN   VARCHAR2,  
    object_type            IN   VARCHAR2 := NULL);
```

パラメータ

表 40-38 DROP_ALL_OBJECTS プロシージャのパラメータ

パラメータ	説明
refresh_template_name	削除するオブジェクトを含んだ配置テンプレートの名前。
object_type	NULL を指定すると、テンプレート内のすべてのオブジェクトが削除されます。オブジェクト型を指定すると、その型のオブジェクトのみ削除されます。次の型のオブジェクトを指定できます。 <div><div>SNAPSHOT</div><div>PROCEDURE</div><div>INDEX</div><div>FUNCTION</div><div>TABLE</div><div>PACKAGE</div><div>VIEW</div><div>PACKAGE BODY</div><div>SYNONYM</div><div>MATERIALIZED VIEW</div><div>SEQUENCE</div><div>DATABASE LINK</div><div>TRIGGER</div></div>

例外

表 40-39 DROP_ALL_OBJECTS プロシージャの例外

例外	説明
miss_refresh_template	指定した配置テンプレート名が無効か、または存在しません。
bad_object_type	オブジェクト型の指定に誤りがあります。有効なオブジェクト型のリストは、表 40-38 を参照してください。

DROP_ALL_TEMPLATE_PARMS プロシージャ

このプロシージャによって、指定した配置テンプレートのテンプレート・パラメータを削除できます。テンプレート・オブジェクトが参照していないすべてのパラメータ、またはパラメータを参照しているすべてのオブジェクトをそのパラメータ自体とともにすべてテンプレートから削除できます。

注意： このプロシージャは、元に戻すことのできない危険度の高いプロシージャです。

構文

```
DBMS_REPCAT_RGT.DROP_ALL_TEMPLATE_PARMS (  
    refresh_template_name    IN    VARCHAR2,  
    drop_objects              IN    VARCHAR2 := 'N');
```

パラメータ

表 40-40 DROP_ALL_TEMPLATE_PARMS プロシージャのパラメータ

パラメータ	説明
refresh_template_name	削除するパラメータとオブジェクトを含んだ配置テンプレートの名前。
drop_objects	値を指定しないと、このパラメータはデフォルトで 'N' に設定され、テンプレート・オブジェクトが参照していないパラメータはすべて削除されます。 'Y' を指定すると、テンプレート・パラメータを参照しているすべてのオブジェクトとそのテンプレート・パラメータ自体が削除されます。オブジェクトは、データベースからではなく、テンプレートから削除されます。

例外

表 40-41 DROP_ALL_TEMPLATE_PARMS プロシージャの例外

例外	説明
miss_refresh_template	指定した配置テンプレート名が無効か、または存在しません。

DROP_ALL_TEMPLATE_SITES プロシージャ

このプロシージャは、DBA_REPCAT_TEMPLATE_SITES ビューからエントリをすべて削除します。このビューには、特定の配置テンプレートをインスタンス化したサイトの記録が格納されています。

注意： このプロシージャは、元に戻すことのできない危険度の高いプロシージャです。また、削除されたテンプレートをインスタンス化している Oracle8i Lite のサイトでは、そのスナップショットをリフレッシュできなくなります。

構文

```
DBMS_REPCAT_RGT.DROP_ALL_TEMPLATE_SITES (
    refresh_template_name IN VARCHAR2);
```

パラメータ

表 40-42 DROP_ALL_TEMPLATE_SITES プロシージャのパラメータ

パラメータ	説明
refresh_template_name	削除するサイトを含んだ配置テンプレートの名前

例外

表 40-43 DROP_ALL_TEMPLATE_SITES プロシージャの例外

例外	説明
miss_refresh_template	指定した配置テンプレート名が無効か、または存在しません。

DROP_ALL_TEMPLATES プロシージャ

このプロシージャは、プロシージャがコールされるサイトの配置テンプレートをすべて削除します。

注意： このプロシージャは、元に戻すことのできない危険度の高いプロシージャです。

構文

```
DBMS_REPCAT_RGT.DROP_ALL_TEMPLATES;
```

パラメータ

なし

DROP_ALL_USER_AUTHORIZATIONS プロシージャ

このプロシージャでは、DBA が、指定した配置テンプレートに対するユーザー認証をすべて削除できます。このプロシージャを実行すると、DBA_REPCAT_USER_AUTHORIZATIONS ビューから行が削除されます。

このプロシージャは、プライベート・テンプレートがパブリック・テンプレートに変換され、ユーザー認証が不要になってからインプリメントされることがあります。

構文

```
DBMS_REPCAT_RGT.DROP_ALL_USER_AUTHORIZATIONS (  
    refresh_template_name IN VARCHAR2);
```

パラメータ

表 40-44 DROP_ALL_USER_AUTHORIZATIONS プロシージャのパラメータ

パラメータ	説明
refresh_template_name	削除するユーザー認証を含んだ配置テンプレートの名前

例外

表 40-45 DROP_ALL_USER_AUTHORIZATIONS プロシージャの例外

例外	説明
miss_refresh_template	指定した配置テンプレート名が無効か、または存在しません。

DROP_ALL_USER_PARM_VALUES プロシージャ

このプロシージャは、特定の配置テンプレートに対するユーザー・パラメータ値を削除します。このプロシージャは柔軟に設計されており、DBA が、削除するユーザー・パラメータ値のセットを定義できます。たとえば、パラメータの指定方法によって次のように処理されます。

`refresh_template_name`: 指定した配置テンプレートに対するユーザー・パラメータをすべて削除します。

`refresh_template_name`、`user_name`: 指定した配置テンプレートに対する指定したユーザー・パラメータをすべて削除します。

`refresh_template_name`、`parameter_name`: 指定した配置テンプレート・パラメータに対するユーザー・パラメータ値をすべて削除します。

`refresh_template_name`、`parameter_name`、`user_name`: 指定した配置テンプレート・パラメータに対する指定したユーザー値を削除します (DROP_USER_PARM と同じです)。

構文

```
DBMS_REPCAT_RGT.DROP_ALL_USER_PARS (  
    refresh_template_name    IN    VARCHAR2,  
    user_name                IN    VARCHAR2,  
    parameter_name           IN    VARCHAR2);
```

パラメータ

表 40-46 DROP_ALL_USER_PARMS プロシージャのパラメータ

パラメータ	説明
refresh_template_name	削除するパラメータ値を含んだ配置テンプレートの名前
user_name	パラメータ値を削除対象とするユーザーの名前
parameter_name	削除する値を含んだテンプレート・パラメータ

例外

表 40-47 DROP_ALL_USER_PARMS プロシージャの例外

例外	説明
miss_refresh_template	指定した配置テンプレート名が無効か、または存在しません。
miss_user	指定したユーザー名が無効か、または存在しません。
miss_user_parm_values	配置テンプレート、ユーザーおよびパラメータの組合せが DBA_REPCAT_USER_PARM_VALUES ビューに存在しません。

DROP_REFRESH_TEMPLATE プロシージャ

このプロシージャは配置テンプレートを削除します。配置テンプレートを削除すると、関連するすべてのテンプレート・パラメータ、ユーザー認証、テンプレート・オブジェクトおよびユーザー・パラメータも段階的に削除されます（このプロシージャでは、テンプレート・サイトは削除されません）。

構文

```
DBMS_REPCAT_RGT.DROP_REFRESH_TEMPLATE (  
    refresh_template_name IN VARCHAR2);
```

パラメータ

表 40-48 DROP_REFRESH_TEMPLATE プロシージャのパラメータ

パラメータ	説明
refresh_template_name	削除する配置テンプレートの名前

例外

表 40-49 DROP_REFRESH_TEMPLATE プロシージャの例外

例外	説明
miss_refresh_template	指定した配置テンプレート名が無効か、または存在しません。配置テンプレートのリストは、DBA_REPCAT_REFRESH_TEMPLATES ビューを参照してください。

DROP_SITE_INSTANTIATION プロシージャ

このプロシージャは、RepAPI サイトを含めて、ターゲット・サイトのテンプレート・インスタンスエーションを削除します。また、マスター・サイトの関連メタデータをすべて削除し、指定したサイトにおけるスナップショットのリフレッシュを使用禁止にします。

構文

```
DBMS_REPCAT_RGT.DROP_SITE_INSTANTIATION (  
    refresh_template_name  IN   VARCHAR2,  
    user_name              IN   VARCHAR2,  
    {site_name             IN   VARCHAR2,  
    | repapi_site_id       IN   NUMBER    := -1e-130,}  
    process_repapi_site    IN   VARCHAR2 := 'N');
```

注意： このプロシージャはオーバーロードされています。site_name と repapi_site_id パラメータは、両方同時には指定できません。

表 40-50 DROP_SITE_INSTANTIATION プロシージャのパラメータ

パラメータ	説明
refresh_template_name	削除する配置テンプレートの名前。
user_name	リモート・スナップショット・サイトでテンプレートを最初にインスタンス化したユーザーの名前。テンプレートをインスタンス化したユーザーを調べるには、ALL_REPCAT_TEMPLATE_SITES ビューを参照してください。
site_name	指定したテンプレート・インスタンスエーションを削除する Oracle Server のサイトを示します。site_name を指定した場合、repapi_site_id は指定しないでください。
repapi_site_id	指定したテンプレート・インスタンスエーションを削除する RepAPI の位置を示します。repapi_site_id を指定した場合、site_name は指定しないでください。
process_repapi_site	'Y' を設定すると、site_name は RepAPI site_name であるとみなされます。デフォルト値は 'N' です。このパラメータは repapi_site_id が NULL 以外の場合、意味を持ちません。

例外

表 40-51 DROP_SITE_INSTANTIATION プロシージャの例外

例外	説明
miss_refresh_template	指定した配置テンプレート名が無効か、または存在しません。
miss_user	指定したユーザー名が存在しません。
miss_template_site	ユーザーとサイトに対する配置テンプレートがインスタンス化されていません。

DROP_TEMPLATE_OBJECT プロシージャ

このプロシージャは、特定の配置テンプレートからテンプレート・オブジェクトを削除します。たとえば、DBA は、古いスナップショットを配置テンプレートから削除するためにこのプロシージャを使用できます。テンプレートの変更内容は、変更後に配置テンプレートをインスタンス化する新規サイトに反映されます。テンプレートをすでにインスタンス化しているリモート・サイトでは、配置テンプレートを再インスタンス化して、変更内容を適用する必要があります。

構文

```
DBMS_REPCAT_RGT.DROP_TEMPLATE_OBJECT (  
    refresh_template_name IN   VARCHAR2,  
    object_name           IN   VARCHAR2,  
    object_type           IN   VARCHAR2);
```

パラメータ

表 40-52 DROP_TEMPLATE_OBJECT プロシージャのパラメータ

パラメータ	説明
refresh_template_name	オブジェクトを削除する配置テンプレートの名前
object_name	削除するテンプレート・オブジェクトの名前
object_type	削除するオブジェクトの型。次の型のオブジェクトを指定できます。 <div>SNAPSHOTPROCEDUREINDEXFUNCTIONTABLEPACKAGEVIEWPACKAGE BODYSYNONYMMATERIALIZED VIEWSEQUENCEDATABASE LINKTRIGGER</div>

例外

表 40-53 DROP_TEMPLATE_OBJECT プロシージャの例外

例外	説明
miss_refresh_template	指定した配置テンプレート名が無効か、または存在しません。
miss_template_object	指定したテンプレート・オブジェクトが無効か、または存在しません。配置テンプレート・オブジェクトのリストは、DBA_REPCAT_TEMPLATE_OBJECTS ビューを参照してください。

DROP_TEMPLATE_PARM プロシージャ

このプロシージャは、DBA_REPCAT_TEMPLATE_PARS ビューから既存のテンプレート・パラメータを削除します。あるテンプレート・オブジェクトを削除して、特定のパラメータが不要になった場合に有効なプロシージャです。

構文

```
DBMS_REPCAT_RGT.DROP_TEMPLATE_PARM (  
    refresh_template_name  IN   VARCHAR2,  
    parameter_name         IN   VARCHAR2);
```

パラメータ

表 40-54 DROP_TEMPLATE_PARM プロシージャのパラメータ

パラメータ	説明
refresh_template_name	削除するパラメータを含んだ配置テンプレートの名前
parameter_name	削除するパラメータの名前

例外

表 40-55 DROP_TEMPLATE_PARM プロシージャの例外

例外	説明
miss_refresh_template	指定した配置テンプレート名が無効か、または存在しません。
miss_template_parm	指定したパラメータ名が無効か、または存在しません。テンプレート・パラメータのリストは、DBA_REPCAT_TEMPLATE_PARS ビューを参照してください。

DROP_USER_AUTHORIZATION プロシージャ

このプロシージャは、DBA_REPCAT_USER_AUTHORIZATIONS ビューからユーザー認証エントリを削除します。ユーザーのテンプレート認可を削除するときに使用します。ユーザーの認証が削除されると、そのユーザーはターゲット配置テンプレートをインスタンス化できません。

関連項目： 詳細は、40-36 ページの「[DROP_ALL_USER_AUTHORIZATIONS プロシージャ](#)」を参照してください。

構文

```
DBMS_REPCAT_RGT.DROP_USER_AUTHORIZATION (  
    refresh_template_name  IN   VARCHAR2,  
    user_name              IN   VARCHAR2);
```

パラメータ

表 40-56 DROP_USER_AUTHORIZATION プロシージャのパラメータ

パラメータ	説明
refresh_template_name	ユーザーの認証を削除する配置テンプレートの名前
user_name	認証を削除対象とするユーザーの名前

例外

表 40-57 DROP_USER_AUTHORIZATION プロシージャの例外

例外	説明
miss_user	指定したユーザー名が無効か、または存在しません。
miss_user_authorization	指定したユーザーと配置テンプレートの組合せが存在しません。ユーザーまたは配置テンプレートのリストは、DBA_REPCAT_USER_AUTHORIZATIONS を参照してください。
miss_refresh_template	指定した配置テンプレート名が無効か、または存在しません。

DROP_USER_PARM_VALUE プロシージャ

このプロシージャは、特定の配置テンプレートに対する事前定義のユーザー・パラメータ値を削除します。通常、ユーザーのテンプレート認可を削除した後に実行します。

構文

```
DBMS_REPCAT_RGT.DROP_USER_PARM_VALUE (
    refresh_template_name    IN    VARCHAR2,
    parameter_name           IN    VARCHAR2,
    user_name                 IN    VARCHAR2);
```

パラメータ

表 40-58 DROP_USER_PARM_VALUE プロシージャのパラメータ

パラメータ	説明
refresh_template_name	削除するパラメータ値を含んだ配置テンプレート名
parameter_name	削除する事前定義値を含んだテンプレートの名前
user_name	パラメータ値を削除対象とするユーザーの名前

例外

表 40-59 DROP_USER_PARM_VALUE プロシージャの例外

例外	説明
miss_refresh_template	指定した配置テンプレート名が無効か、または存在しません。
miss_user	指定したユーザー名が無効か、または存在しません。
miss_user_parm_values	配置テンプレート、ユーザーおよびパラメータの組合せが DBA_REPCAT_USER_PARM_VALUES ビューに存在しません。

GET_RUNTIME_PARM_ID ファンクション

このファンクションは、ランタイム・パラメータ値の定義時に使用する ID を取り出します。ランタイム・パラメータ値はすべてこの ID に割り当てられ、インスタンス化プロセス時にも使用されます。

構文

```
DBMS_REPCAT_RGT.GET_RUNTIME_PARM_ID
RETURN NUMBER;
```

パラメータ

なし

戻り値

表 40-60 GET_RUNTIME_PARM_ID ファンクションの戻り値

戻り値	対応するデータ型
<システム生成番号>	ランタイム・パラメータ値は、このシステム生成番号に割り当てられ、インスタンス化プロセス時にも使用されます。

INSERT_RUNTIME_PARMS プロシージャ

このプロシージャは、テンプレートのインスタンス化前にランタイム・パラメータ値を定義します。ユーザー・パラメータ値が未定義で、デフォルトのパラメータ値を使用しない場合は、このプロシージャを使用してパラメータ値を定義する必要があります。

このプロシージャを使用する前に、必ず GET_RUNTIME_PARM_ID ファンクションを実行して、ランタイム・パラメータの挿入時に使用するパラメータ ID を取り出してください。この ID は、ランタイム・パラメータ値の定義と配置テンプレートのインスタンス化に使用されます。

構文

```
DBMS_REPCAT_RGT.INSERT_RUNTIME_PARMS (  
    runtime_parm_id    IN    NUMBER,  
    parameter_name     IN    VARCHAR2,  
    parameter_value    IN    CLOB);
```

パラメータ

表 40-61 INSERT_RUNTIME_PARMS プロシージャのパラメータ

パラメータ	説明
runtime_parm_id	GET_RUNTIME_PARM_ID ファンクションで取り出された ID。この ID は、配置テンプレートのインスタンス化時にも使用されます。配置テンプレートのすべてのパラメータ値には、必ず同じ ID を使用してください。
parameter_name	ランタイム・パラメータ値を定義するテンプレート・パラメータの名前。テンプレート・パラメータのリストは、DBA_REPCAT_TEMPLATE_PARMS ビューを参照してください。
parameter_value	配置テンプレートのインスタンス化プロセス時に使用するランタイム・パラメータ値。

例外

表 40-62 INSERT_RUNTIME_PARS プロシージャの例外

例外	説明
miss_refresh_ template	指定した配置テンプレート名が無効か、または存在しません。
miss_user	指定したユーザー名が無効か、または存在しません。
miss_user_parm_ values	配置テンプレート、ユーザーおよびパラメータの組合せが DBA_ REPCAT_USER_PARM_VALUES ビューに存在しません。

使用上の注意

INSERT_RUNTIME_PARS は、CLOB を使用しているため、このプロシージャを使用する場合は、DBMS_LOB パッケージを使用する必要があります。次の例は、INSERT_RUNTIME_PARS プロシージャで DBMS_LOB パッケージを使用する方法を示しています。

```
DECLARE
    tempstring VARCHAR2(100);
    templob CLOB;
BEGIN
    DBMS_LOB.CREATETEMPORARY(templob, TRUE, DBMS_LOB.SESSION);
    tempstring := 'REGION 20';
    DBMS_LOB.WRITE(templob, length(tempstring), 1, tempstring);
    DBMS_REPCAT_RGT.INSERT_RUNTIME_PARS(
        runtime_parm_id => 20,
        parameter_name => 'region',
        parameter_value => templob);
    DBMS_LOB.FREETEMPORARY(templob);
END;
```


INstantiate_Offline ファンクション

このファンクションは、マスター・サイトでスクリプトを生成します。このスクリプトは、オフライン時にリモート・スナップショット・サイトでスナップショット環境を作成するために使用されます。リモート・スナップショット・サイトでのインスタンス化プロセスには時間を要する場合があるため（必要な時間は新しいスナップショットに移入されるデータの量によって異なります）、生成されたスクリプトは、マスター・サイトに長時間接続したままにできないリモート・スナップショット・サイトで使用してください。このファンクションは、ユーザー・インスタンスエーションごとに個別に実行する必要があります。

このファンクションで生成されたスクリプトは、USER_REPCAT_TEMP_OUTPUT 一時ビューに格納され、配置テンプレートの配布時に、Replication Manager などの Oracle Tools で使用されます。このファンクションで戻される数値は、USER_REPCAT_TEMP_OUTPUT ビューから該当する情報を取り出すために使用されます。

注意： このファンクションは、配置テンプレートのオフライン・インスタンスエーションの実行時に使用されます。また、別のユーザーのためにインスタンス化を実行しているレプリケーション管理者用でもあります。独自のインスタンス化を実行するユーザーは、パブリック・バージョンの INstantiate_Offline ファンクションを使用してください。詳細は、39-3 ページの「[INstantiate_Offline ファンクション](#)」を参照してください。

このファンクションを、DBMS_OFFLINE_OG パッケージ内のプロシージャ（マスター表のオフライン・インスタンスエーションの実行に使用）や DBMS_OFFLINE_SNAPSHOT パッケージ内のプロシージャ（スナップショットのオフライン・インスタンスエーションの実行に使用）と混同しないでください。それぞれのパッケージの使用方法は、該当の章を参照してください。

構文

```
DBMS_REPCAT_RGT.INSTANTIATE_OFFLINE (
    refresh_template_name  IN   VARCHAR2,
    site_name              IN   VARCHAR2,
    user_name              IN   VARCHAR2  := NULL,
    runtime_parm_id        IN   NUMBER    := -1e-130,
    next_date              IN   DATE      := SYSDATE,
    interval               IN   VARCHAR2  := 'SYSDATE + 1',
    use_default_gowner     IN   BOOLEAN   := TRUE)
return NUMBER;
```

パラメータ

表 40-63 INSTANTIATE_OFFLINE ファンクションのパラメータ

パラメータ	説明
refresh_template_name	インスタンス化する配置テンプレートの名前。
site_name	配置テンプレートをインスタンス化するリモート・サイトの名前。
user_name	配置テンプレートをインスタンス化する認可ユーザーの名前。
runtime_parm_id	INSERT_RUNTIME_PARMS プロシージャを使用してランタイム・パラメータ値を定義している場合は、ランタイム・パラメータの作成時に使用した ID (GET_RUNTIME_PARM_ID ファンクションを使用して取り出された ID です) を指定します。
next_date	リフレッシュ・グループの作成時に使用される次回リフレッシュ日付の値を指定します。
interval	リフレッシュ・グループの作成時に使用されるリフレッシュ間隔を指定します。
use_default_gowner	TRUE の場合、作成されたすべてのスナップショット・オブジェクト・グループはデフォルト・ユーザー PUBLIC が所有します。 FALSE の場合、作成されたすべてのスナップショット・オブジェクト・グループはインスタンス化を実施したユーザーが所有します。

例外

表 40-64 INSTANTIATE_OFFLINE ファンクションの例外

例外	説明
miss_refresh_template	指定した配置テンプレート名が無効か、または存在しません。
miss_user	指定した認可ユーザーの名前が無効か、または存在しません。指定したユーザーが DBA_REPCAT_USER_AUTHORIZATIONS ビューにリストされていることを検証してください。リストされていない場合、指定したユーザーは、ターゲット配置テンプレートのインスタンス化を認可されていません。

戻り値

表 40-65 INSTANTIATE_OFFLINE ファンクションの戻り値

戻り値	説明
<システム生成番号>	生成したインスタンス・スクリプトを USER_REPCAT_TEMP_OUTPUT ビューで取り出すように選択したとき、output_id に対して生成されるシステム番号を指定します。

INSTANTIATE_OFFLINE_REPAPI ファンクション

このファンクションは、マスター・サイトでファイルを生成します。このファイルは、オフライン時にリモートの RepAPI スナップショット・サイトでスナップショット環境を作成するために使用されます。オフライン・インスタンス・ファイルは、長時間マスター・サイトに接続したままにできない RepAPI サイトで使用してください。

これは、リモート・スナップショット・サイトが Oracle8i Lite (RepAPI を含めて) を実行しているラップトップの場合に理想的なソリューションとなります。生成したファイルは、FTP サイトに転記したり、CD-ROM やフロッピー・ディスクなどにロードできます。

このファンクションで生成されたファイルは、パラメータ OFFLINE_DIRPATH で指定したディレクトリのマスター・サイトに格納されます。ファイルは user_name、refresh_template_name および site_id に基づいて命名され、ファイル拡張子 .oli が指定されます。たとえば、サイト 1234 でテンプレート名 MYTEMPLATE のユーザー SCOTT に対するオフライン・インスタンス・ファイルは、次のように名前が指定されます。

scott_mytemplate_1234.oli.

注意： このファンクションは、配置テンプレートのオフライン・インスタンス・スクリプトの実行時に使用されます。また、別のユーザーのためにインスタンス化を実行しているレプリケーション管理者用でもあります。独自のインスタンス化を実行するユーザーは、パブリック・バージョンの [INSTANTIATE_OFFLINE_REPAPI ファンクション](#) を使用してください。詳細は、39-6 ページの「[INSTANTIATE_OFFLINE_REPAPI ファンクション](#)」を参照してください。

このファンクションを、DBMS_OFFLINE_OG パッケージ内のプロシージャ (マスター表のオフライン・インスタンス・スクリプトの実行に使用) や DBMS_OFFLINE_SNAPSHOT パッケージ内のプロシージャ (スナップショットのオフライン・インスタンス・スクリプトの実行に使用) と混同しないでください。それぞれのパッケージの使用方法は、該当の章を参照してください。

構文

```
DBMS_REPCAT_RGT.INSTANTIATE_OFFLINE_REPAPI (
  refresh_template_name  IN  VARCHAR2,
  site_id                IN  VARCHAR2,
  user_name              IN  VARCHAR2  := USER,
  master                 IN  VARCHAR2  := NULL,
  url                    IN  VARCHAR2  := NULL,
  ssl                    IN  NUMBER    := 0,
  trace_vector           IN  NUMBER    := DBMS_REPCAT_RGT.NO_TRACE_DUMP,
  resultset_threshold    IN  NUMBER    := DBMS_REPCAT_INSTANTIATE.
                                     RESULTSET_THRESHOLD,
  lob_threshold          IN  NUMBER    := DBMS_REPCAT_INSTANTIATE.
                                     LOB_THRESHOLD);
```

表 40-66 INSTANTIATE_OFFLINE_REPAPI ファンクションのパラメータ

パラメータ	説明
refresh_template_name	インスタンス化する配置テンプレートの名前。
site_id	配置テンプレートをインスタンス化するリモート・サイトの識別番号。この番号は、スナップショット・サイトのサイト識別子です。通常、このパラメータに指定される値は一時的な値であるため、後続の操作で RepAPI クライアントによって更新される可能性があります。
user_name	インスタンスエーション・ファイルが生成されるユーザーの名前。
master	RepAPI クライアントがサーバーに使用するオプションの別名。指定した場合、RepAPI クライアントは常にこの別名でサーバーを参照します。
url	データベースにアクセスするためのマスター・サイトの公開 URL。指定した場合、RepAPI クライアントは常にこの URL でサーバーを参照します。
ssl	1 は、スナップショットがマスター・サイトとの通信で保護ソケット・レイヤー（SSL）を使用することを示します。0 は、SSL を使用しないことを意味します。
trace_vector	デバッグのトレース・レベル。
resultset_threshold	スナップショット・リフレッシュ処理時に送信する非 LOB 行データの最大サイズ。
lob_threshold	スナップショット・リフレッシュ処理時に送信する LOB 行データの最大サイズ。

表 40-67 INSTANTIATE_OFFLINE_REPAPI ファンクションの例外

例外	説明
miss_refresh_template	テンプレートが存在しません。
miss_user	ユーザー名がデータベース内に存在しません。
miss_template_site	ユーザーとサイトに対するテンプレートがインスタンス化されていません。

戻り値

表 40-68 INSTANTIATE_OFFLINE_REPAPI ファンクションの戻り値

戻り値	説明
0	エラーが見つかりました。
1	エラーはありません。

INstantiate ONLINE ファンクション

このファンクションは、マスター・サイトでスクリプトを生成します。このスクリプトは、オンライン時にリモート・スナップショット・サイトでスナップショット環境を作成するために使用します。リモート・スナップショット・サイトでのインスタンス化プロセスは長くなる場合があります（必要な時間は新しいスナップショットに移入されるデータの量によって異なります）、生成されたスクリプトは、マスター・サイトに長時間接続したままにできるリモート・スナップショット・サイトで使用してください。このファンクションは、ユーザー・インスタンスエーションごとに別々に実行する必要があります。

このファンクションで生成されたスクリプトは、USER_REPCAT_TEMP_OUTPUT 一時ビューに格納され、配置テンプレートの配布時に、Replication Manager などの Oracle Tools で使用されます。このファンクションで戻される数値は、USER_REPCAT_TEMP_OUTPUT ビューから該当する情報を取り出すために使用されます。

注意： このファンクションは、別のユーザーのためにインスタンス化を実行しているレプリケーション管理者用でもあります。独自のインスタンス化を実行するユーザーは、39-9 ページの「[INstantiate ONLINE ファンクション](#)」セクションに記述されている [INstantiate ONLINE ファンクション](#) のパブリック・バージョンを使用してください。

構文

```
DBMS_REPCAT_RGT.INSTANTIATE_ONLINE(  
    refresh_template_name  IN   VARCHAR2,  
    site_name              IN   VARCHAR2  := NULL,  
    user_name              IN   VARCHAR2  := NULL,  
    runtime_parm_id        IN   NUMBER    := -1e-130,  
    next_date              IN   DATE      := SYSDATE,  
    interval               IN   VARCHAR2  := 'SYSDATE + 1',  
    use_default_gowner     IN   BOOLEAN   := TRUE)  
    return NUMBER;
```

パラメータ

表 40-69 INSTANTIATE_ONLINE ファンクションのパラメータ

パラメータ	説明
refresh_template_name	インスタンス化する配置テンプレートの名前。
site_name	配置テンプレートをインスタンス化するリモート・サイトの名前。
user_name	配置テンプレートをインスタンス化する認可ユーザーの名前。
runtime_parm_id	INSERT_RUNTIME_PARMS プロシーダを使用してランタイム・パラメータ値を定義している場合は、ランタイム・パラメータの作成時に使用した ID (GET_RUNTIME_PARM_ID ファンクションを使用して取り出された ID です) を指定します。
next_date	リフレッシュ・グループの作成時に使用される次回リフレッシュ日付の値を指定します。
interval	リフレッシュ・グループの作成時に使用されるリフレッシュ間隔を指定します。
use_default_gowner	TRUE の場合、作成されたすべてのスナップショット・オブジェクト・グループはデフォルト・ユーザー PUBLIC が所有します。FALSE の場合、作成されたすべてのスナップショット・オブジェクト・グループはインスタンス化を実施したユーザーが所有します。

例外

表 40-70 INSTANTIATE_ONLINE ファンクションの例外

例外	説明
miss_refresh_template	指定した配置テンプレート名が無効か、または存在しません。
miss_user	指定した認可ユーザーの名前が無効か、または存在しません。指定したユーザーが DBA_REPCAT_USER_AUTHORIZATIONS ビューにリストされていることを検証してください。リストされていない場合、指定したユーザーは、ターゲット配置テンプレートのインスタンス化を認可されていません。
bad_parms	定義したユーザー・パラメータ値またはテンプレート・デフォルト値によって移入されていないテンプレート・パラメータがあります。事前定義値の数とテンプレート・パラメータ数が一致していないか、または事前定義値がターゲット・パラメータに対して無効です（型の不一致など）。

戻り値

表 40-71 INSTANTIATE_ONLINE ファンクションの戻り値

戻り値	説明
<システム生成番号>	生成したインスタンスエーション・スクリプトを USER_REPCAT_TEMP_OUTPUT ビューで取り出すように選択したとき、output_id に対してシステムが生成した番号を指定します。

LOCK_TEMPLATE_EXCLUSIVE プロシージャ

このプロシージャは、配置テンプレートの更新中または変更中に、ユーザーがテンプレートの読み込みまたはインスタンス化を実行できないようにします。

ロックは、ROLLBACK または COMMIT が実行されるとリリースされます。

注意： 配置テンプレートを変更する前に、このプロシージャを必ず実行してください。

構文

```
DBMS_REPCAT_RGT.LOCK_TEMPLATE_EXCLUSIVE();
```

パラメータ

なし

LOCK_TEMPLATE_SHARED プロシージャ

この LOCK_TEMPLATE_SHARED プロシージャは、指定した配置テンプレートを読み取り専用にします。テンプレートをインスタンス化する前に、このプロシージャを必ずコールしてください。この結果、インスタンス化中に他のユーザーによって配置テンプレートが変更されないことが保証されます。

ロックは、ROLLBACK または COMMIT が実行されるとリリースされます。

構文

```
DBMS_REPCAT_RGT.LOCK_TEMPLATE_SHARED();
```

パラメータ

なし

DBMS_REPUTIL には、表のレプリケーションで使用するシャドウ表、トリガーおよびパッケージを生成するサブプログラムの他に、スタンドアロン・プロシージャ起動とパッケージ・プロシージャ起動のレプリケーションに使用するラッパーを生成するサブプログラムが含まれています。

このパッケージは、生成コードでのみ参照されます。

DBMS_REPUTIL パッケージ

サブプログラムの要約

表 41-1 DBMS_REPUTIL パッケージのサブプログラム

サブプログラム	説明
41-3 ページの REPLICATION_OFF プロシージャ	変更内容をレプリケート環境にある他のサイトにレプリケートせずに表を変更したり、またはプロシージャ型レプリケーションの使用時に行レベル・レプリケーションを使用禁止にします。
41-3 ページ の REPLICATION_ON プロシージャ	レプリケーションが一時的に中断された後に、変更内容のレプリケーションを再び使用可能にします。
41-4 ページの REPLICATION_IS_ON ファンクション	レプリケーションが実行中かどうかを判別します。
41-4 ページの FROM_REMOTE ファンクション	内部レプリケーション・パッケージにあるプロシージャの開始時に TRUE を戻し、終了時に FALSE を戻します。
41-5 ページの GLOBAL_NAME ファンクション	ローカル・データベースのグローバル・データベース名を判別します（戻り値はグローバル名）。
41-5 ページの MAKE_INTERNAL_PKG プロシージャ	レプリケーション・カタログの内部パッケージと表を同期化します。このプロシージャは、オラクル社カスタマ・サポート・センターの指示があった場合のみ実行してください。
41-6 ページの SYNC_UP_REP プロシージャ	レプリケーション・カタログの内部トリガーと表またはスナップショットを同期化します。このプロシージャは、オラクル社カスタマ・サポート・センターの指示があった場合のみ実行してください。

REPLICATION_OFF プロシージャ

このプロシージャによって、変更内容をレプリケート環境にある他のサイトにレプリケートせずに表を変更したり、プロシージャ型レプリケーションの使用時に行レベル・レプリケーションを使用禁止にすることができます。一般的に、このフラグを設定する前には、レプリケート環境にあるすべてのマスター・グループに対するレプリケーション・アクティビティを中断する必要があります。

構文

```
DBMS_REPUTIL.REPLICATION_OFF();
```

パラメータ

なし

REPLICATION_ON プロシージャ

このプロシージャは、レプリケーションが一時的に中断された後に、変更内容のレプリケーションを再び使用可能にします。

構文

```
DBMS_REPUTIL.REPLICATION_ON();
```

パラメータ

なし

REPLICATION_IS_ON ファンクション

このファンクションは、レプリケーションが実行中かどうかを判別します。戻り値が TRUE の場合は、生成されたレプリケーション・トリガーが使用可能であることを示します。戻り値が FALSE の場合は、レプリケート・マスター・グループに対する現行のサイトでのレプリケーションは使用禁止であることを示します。

このファンクションの戻り値は、DBMS_REPUTIL パッケージの REPLICATION_ON または REPLICATION_OFF プロシージャをコールして設定されます。

構文

```
DBMS_REPUTIL.REPLICATION_IS_ON()  
    return BOOLEAN;
```

パラメータ

なし

FROM_REMOTE ファンクション

このファンクションは、内部レプリケーション・パッケージにあるプロシージャの開始時に TRUE を返し、終了時に FALSE を返します。内部パッケージによる更新の結果、起動するトリガーがある場合は、このファンクションをチェックする必要があります。

構文

```
DBMS_REPUTIL.FROM_REMOTE()  
    return BOOLEAN;
```

パラメータ

なし

GLOBAL_NAME ファンクション

このファンクションは、ローカル・データベースのグローバル・データベース名を判別します（戻り値はグローバル名）。

構文

```
DBMS_REPUTIL.GLOBAL_NAME()  
    return VARCHAR2;
```

パラメータ

なし

MAKE_INTERNAL_PKG プロシージャ

このプロシージャは、実在する内部パッケージとレプリケーション・カタログの表を同期化します。表にレプリケーションのサポートがある場合は、このプロシージャを実行することによって内部パッケージを作成できます。レプリケーション・サポートがない場合、関連する内部パッケージは破棄されます。

注意： このプロシージャは、オラクル社カスタマ・サポート・センターの指示があった場合のみ実行してください。

構文

```
DBMS_REPUTIL.MAKE_INTERNAL_PKG (  
    canon_sname    IN   VARCHAR2,  
    canon_onsame   IN   VARCHAR2);
```

パラメータ

表 41-2 MAKE_INTERNAL_PKG プロシージャのパラメータ

パラメータ	説明
canon_sname	同期化する表を含んだスキーマ。 このパラメータ値は規範的に定義する必要があります（オブジェクトと合致する大文字を使用し、二重引用符で囲まないでください）。
canon_onsame	同期化する表の名前。 このパラメータ値は規範的に定義する必要があります（オブジェクトと合致する大文字を使用し、二重引用符で囲まないでください）。

SYNC_UP_REP プロシージャ

このプロシージャは、実在する内部トリガーとレプリケーション・カタログの表またはスナップショットを同期化します。表またはスナップショットにレプリケーションのサポートがある場合は、このプロシージャを実行することによって内部レプリケーション・トリガーを作成できます。レプリケーション・サポートがない場合、関連する内部トリガーは破棄されます。

注意： このプロシージャは、オラクル社カスタマ・サポート・センターの指示があった場合のみ実行してください。

構文

```
DBMS_REPUTIL.SYNC_UP_REP (  
    canon_sname      IN   VARCHAR2,  
    canon_onsame      IN   VARCHAR2);
```

パラメータ

表 41-3 SYNC_UP_REP プロシージャのパラメータ

パラメータ	説明
canon_sname	同期化する表またはスナップショットを含んだスキーマ。 このパラメータ値は規範的に定義する必要があります（オブジェクトと合致する大文字を使用し、二重引用符で囲まないでください）。
canon_onsame	同期化する表またはスナップショットの名前。 このパラメータ値は規範的に定義する必要があります（オブジェクトと合致する大文字を使用し、二重引用符で囲まないでください）。

DBMS_RESOURCE_MANAGER

DBMS_RESOURCE_MANAGER パッケージは、プラン、コンシューマ・グループおよびプラン・ディレクティブをメンテナンスします。また、プラン・スキーマへの変更内容をグループ化する方法も提供します。

関連項目： データベース・リソース・マネージャの使用の詳細は、『Oracle8i 管理者ガイド』を参照してください。

要件

実行者には、このプロシージャを実行するための ADMINISTER_RESOURCE_MANAGER システム権限が必要です。この権限の付与と取消しを行うプロシージャは、DBMS_RESOURCE_MANAGER_PRIVS パッケージにあります。

サブプログラムの要約

表 42-1 DBMS_RESOURCE_MANAGER パッケージのサブプログラム

サブプログラム	説明
42-4 ページの CREATE_PLAN プロシージャ	リソース・プランを定義するエントリを作成します。
42-5 ページの UPDATE_PLAN プロシージャ	リソース・プランを定義するエントリを更新します。
42-5 ページの DELETE_PLAN プロシージャ	指定のプランとそれが参照するすべてのプラン・ディレクティブを削除します。
42-6 ページの DELETE_PLAN_CASCADE プロシージャ	指定のプランとそのすべての子（プラン・ディレクティブ、サブプラン、コンシューマ・グループ）を削除します。
42-7 ページの CREATE_CONSUMER_GROUP プロシージャ	リソース・コンシューマ・グループを定義するエントリを作成します。
42-7 ページの UPDATE_CONSUMER_GROUP プロシージャ	リソース・コンシューマ・グループを定義するエントリを更新します。
42-8 ページの DELETE_CONSUMER_GROUP プロシージャ	リソース・コンシューマ・グループを定義するエントリを削除します。
42-9 ページの CREATE_PLAN_DIRECTIVE プロシージャ	リソース・プラン・ディレクティブを作成します。
42-10 ページの UPDATE_PLAN_DIRECTIVE プロシージャ	リソース・プラン・ディレクティブを更新します。
42-11 ページの DELETE_PLAN_DIRECTIVE プロシージャ	リソース・プラン・ディレクティブを削除します。
42-12 ページの CREATE_PENDING_AREA プロシージャ	リソース・マネージャ・オブジェクトへの変更を行うための作業領域を作成します。
42-13 ページの VALIDATE_PENDING_AREA プロシージャ	リソース・マネージャに対する保留中の変更を検証します。
42-13 ページの CLEAR_PENDING_AREA プロシージャ	リソース・マネージャに対する作業領域を消去します。

表 42-1 DBMS_RESOURCE_MANAGER パッケージのサブプログラム

サブプログラム	説明
42-14 ページの SUBMIT_PENDING_AREA プロシージャ	リソース・マネージャに対する保留中の変更を実行します。
42-17 ページの SET_INITIAL_CONSUMER_GROUP プロシージャ	ユーザーに対して、初期リソース・コンシューマ・グループを割り当てます。
42-18 ページの SWITCH_CONSUMER_GROUP_FOR_SESS プロシージャ	指定のセッションのリソース・コンシューマ・グループを変更します。
42-19 ページの SWITCH_CONSUMER_GROUP_FOR_USER プロシージャ	指定のユーザー名で、すべてのセッションのリソース・コンシューマ・グループを変更します。

CREATE_PLAN プロシージャ

このプロシージャは、リソース・プランを定義するエントリを作成します。

構文

```
DBMS_RESOURCE_MANAGER.CREATE_PLAN (  
    plan                               IN VARCHAR2,  
    comment                           IN VARCHAR2,  
    cpu_mth                           IN VARCHAR2 DEFAULT 'EMPHASIS',  
    max_active_sess_target_mth        IN VARCHAR2 DEFAULT 'MAX_ACTIVE_SESS_ABSOLUTE',  
    parallel_degree_limit_mth         IN VARCHAR2 DEFAULT 'PARALLEL_DEGREE_LIMIT_  
ABSOLUTE');
```

パラメータ

表 42-2 CREATE_PLAN プロシージャのパラメータ

パラメータ	説明
plan	リソース・プランの名前
comment	ユーザー・コメント
cpu_mth	CPU リソースに対する割当て方法
max_active_sess_target_mth	アクティブな最大セッションに対する割当て方法
parallel_degree_limit_mth	並列度に対する割当て方法

UPDATE_PLAN プロシージャ

このプロシージャは、リソース・プランを定義するエントリを更新します。

構文

```
DBMS_RESOURCE_MANAGER.UPDATE_PLAN (  
    plan                               IN VARCHAR2,  
    new_comment                        IN VARCHAR2 DEFAULT NULL,  
    new_cpu_mth                       IN VARCHAR2 DEFAULT NULL,  
    new_max_active_sess_target_mth    IN VARCHAR2 DEFAULT NULL,  
    new_parallel_degree_limit_mth     IN VARCHAR2 DEFAULT NULL);
```

パラメータ

表 42-3 UPDATE_PLAN プロシージャのパラメータ

パラメータ	説明
plan	リソース・プランの名前
new_comment	新規のユーザー・コメント
new_cpu_mth	CPU リソースに対する新規の割当て方法名
new_max_active_sess_target_mth	アクティブな最大セッションに対する新規の方法名
new_parallel_degree_limit_mth	並列度に対する新規の方法名

使用上の注意

UPDATE_PLAN に対するパラメータを指定しない場合、これらのパラメータは、データ・ディクショナリ内で変更されないまま残ります。

DELETE_PLAN プロシージャ

このプロシージャは、指定のプランとそれが参照するすべてのプラン・ディレクティブを削除します。

構文

```
DBMS_RESOURCE_MANAGER.DELETE_PLAN (  
    plan IN VARCHAR2);
```

パラメータ

表 42-4 DELETE_PLAN プロシージャのパラメータ

パラメータ	説明
plan	削除するリソース・プランの名前

DELETE_PLAN_CASCADE プロシージャ

このプロシージャは、指定のプランとそのすべての子（プラン・ディレクティブ、サブプラン、コンシューマ・グループ）を削除します。必須オブジェクトと必須ディレクティブは削除されません。

構文

```
DBMS_RESOURCE_MANAGER.DELETE_PLAN_CASCADE (  
    plan IN VARCHAR2);
```

パラメータ

表 42-5 DELETE_PLAN_CASCADE プロシージャのパラメータ

パラメータ	説明
plan	プランの名前

エラー

DELETE_PLAN_CASCADE プロシージャでエラーが発生した場合は、ロールバックされるため、何も削除されません。

注意： デフォルトのリソース割当て方法を使用する場合は、プランの作成または更新時に方法を指定する必要はありません。

使用上の注意

デフォルトは次のとおりです。

- cpu_method = EMPHASIS
- parallel_degree_limit_mth = PARALLEL_DEGREE_LIMIT_ABSOLUTE
- max_active_sess_target_mth = MAX_ACTIVE_SESS_ABSOLUTE

注意： パラメータ max_active_sess_target_mth は、このリリースでは記載されていません。このパラメータは、将来使用するために予約されています。

CREATE_CONSUMER_GROUP プロシージャ

このプロシージャによって、リソース・コンシューマ・グループを定義するエントリを作成します。

構文

```
DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP (  
    consumer_group IN VARCHAR2,  
    comment        IN VARCHAR2,  
    cpu_mth        IN VARCHAR2 DEFAULT 'ROUND-ROBIN');
```

パラメータ

表 42-6 CREATE_CONSUMER_GROUP プロシージャのパラメータ

パラメータ	説明
consumer_group	コンシューマ・グループの名前
comment	ユーザー・コメント
cpu_mth	CPU リソース割当て方法名

UPDATE_CONSUMER_GROUP プロシージャ

このプロシージャによって、リソース・コンシューマ・グループを定義するエントリを更新します。

構文

```
DBMS_RESOURCE_MANAGER.UPDATE_CONSUMER_GROUP (  
    consumer_group IN VARCHAR2,  
    new_comment    IN VARCHAR2 DEFAULT NULL,  
    new_cpu_mth    IN VARCHAR2 DEFAULT NULL);
```

パラメータ

表 42-7 UPDATE_CONSUMER_GROUP プロシージャのパラメータ

パラメータ	説明
consumer_group	コンシューマ・グループの名前
new_comment	新規のユーザー・コメント
new_cpu_mth	CPU リソース割当てに対する新規の方法名

UPDATE_CONSUMER_GROUP に対するパラメータを指定しない場合、これらのパラメータは、データ・ディクショナリ内で変更されないまま残ります。

DELETE_CONSUMER_GROUP プロシージャ

このプロシージャによって、リソース・コンシューマ・グループを定義するエントリを削除します。

構文

```
DBMS_RESOURCE_MANAGER.DELETE_CONSUMER_GROUP (  
    consumer_group IN VARCHAR2);
```

パラメータ

表 42-8 DELETE_CONSUMER_GROUP プロシージャのパラメータ

パラメータ	説明
consumer_group	削除するコンシューマ・グループの名前

CREATE_PLAN_DIRECTIVE プロシージャ

このプロシージャによって、リソース・プラン・ディレクティブを作成します。

構文

```
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE (
    plan                      IN VARCHAR2,
    group_or_subplan         IN VARCHAR2,
    comment                  IN VARCHAR2,
    cpu_p1                   IN NUMBER    DEFAULT NULL,
    cpu_p2                   IN NUMBER    DEFAULT NULL,
    cpu_p3                   IN NUMBER    DEFAULT NULL,
    cpu_p4                   IN NUMBER    DEFAULT NULL,
    cpu_p5                   IN NUMBER    DEFAULT NULL,
    cpu_p6                   IN NUMBER    DEFAULT NULL,
    cpu_p7                   IN NUMBER    DEFAULT NULL,
    cpu_p8                   IN NUMBER    DEFAULT NULL,
    max_active_sess_target_p1 IN NUMBER    DEFAULT NULL,
    parallel_degree_limit_p1 IN NUMBER    DEFAULT NULL);
```

パラメータ

表 42-9 CREATE_PLAN_DIRECTIVE プロシージャのパラメータ

パラメータ	説明
plan	リソース・プランの名前
group_or_subplan	コンシューマ・グループの名前またはサブプランの名前
comment	プラン・ディレクティブについてのコメント
cpu_p1	CPU リソース割当て方法に対する第 1 パラメータ
cpu_p2	CPU リソース割当て方法に対する第 2 パラメータ
cpu_p3	CPU リソース割当て方法に対する第 3 パラメータ
cpu_p4	CPU リソース割当て方法に対する第 4 パラメータ
cpu_p5	CPU リソース割当て方法に対する第 5 パラメータ
cpu_p6	CPU リソース割当て方法に対する第 6 パラメータ
cpu_p7	CPU リソース割当て方法に対する第 7 パラメータ
cpu_p8	CPU リソース割当て方法に対する第 8 パラメータ
max_active_sess_target_p1	最大アクティブ・セッション割当て方法に対する第 1 パラメータ (将来使用のために予約)

表 42-9 CREATE_PLAN_DIRECTIVE プロシージャのパラメータ

パラメータ	説明
parallel_degree_limit_p1	並列度の割当て方法に対する第 1 パラメータ

すべてのパラメータは、NULL にデフォルト設定されます。ただし、EMPHASIS CPU リソース割当て方法の場合は、ユーザーがすべてのパラメータを入力する必要があります。

UPDATE_PLAN_DIRECTIVE プロシージャ

このプロシージャによって、リソース・プラン・ディレクティブを更新します。

構文

```
DBMS_RESOURCE_MANAGER.UPDATE_PLAN_DIRECTIVE (  
  plan                                IN VARCHAR2,  
  group_or_subplan                   IN VARCHAR2,  
  new_comment                        IN VARCHAR2 DEFAULT NULL,  
  new_cpu_p1                         IN NUMBER   DEFAULT NULL,  
  new_cpu_p2                         IN NUMBER   DEFAULT NULL,  
  new_cpu_p3                         IN NUMBER   DEFAULT NULL,  
  new_cpu_p4                         IN NUMBER   DEFAULT NULL,  
  new_cpu_p5                         IN NUMBER   DEFAULT NULL,  
  new_cpu_p6                         IN NUMBER   DEFAULT NULL,  
  new_cpu_p7                         IN NUMBER   DEFAULT NULL,  
  new_cpu_p8                         IN NUMBER   DEFAULT NULL,  
  new_max_active_sess_target_p1     IN NUMBER   DEFAULT NULL,  
  new_parallel_degree_limit_p1      IN NUMBER   DEFAULT NULL);
```

パラメータ

表 42-10 UPDATE_PLAN_DIRECTIVE プロシージャのパラメータ

パラメータ	説明
plan	リソース・プランの名前
group_or_subplan	コンシューマ・グループの名前またはサブプランの名前
new_comment	プラン・ディレクティブについてのコメント
new_cpu_p1	CPU リソース割当て方法に対する第 1 パラメータ
new_cpu_p2	CPU リソース割当て方法に対する第 2 パラメータ
new_cpu_p3	CPU リソース割当て方法に対する第 3 パラメータ

表 42-10 UPDATE_PLAN_DIRECTIVE プロシージャのパラメータ

パラメータ	説明
new_cpu_p4	CPU リソース割当て方法に対する第 4 パラメータ
new_cpu_p5	CPU リソース割当て方法に対する第 5 パラメータ
new_cpu_p6	CPU リソース割当て方法に対する第 6 パラメータ
new_cpu_p7	CPU リソース割当て方法に対する第 7 パラメータ
new_cpu_p8	CPU リソース割当て方法に対する第 8 パラメータ
new_max_active_sess_target_p1	最大アクティブ・セッション割当て方法に対する第 1 パラメータ (将来使用のために予約)
new_parallel_degree_limit_p1	並列度の割当て方法に対する第 1 パラメータ

UPDATE_PLAN_DIRECTIVE に対してパラメータを指定しない場合、これらのパラメータは、データ・ディクショナリ内で変更されないまま残ります。

DELETE_PLAN_DIRECTIVE プロシージャ

このプロシージャによって、リソース・プラン・ディレクティブを削除します。

構文

```
DBMS_RESOURCE_MANAGER.DELETE_PLAN_DIRECTIVE (
    plan                IN VARCHAR2,
    group_or_subplan IN VARCHAR2);
```

パラメータ

表 42-11 DELETE_PLAN_DIRECTIVE プロシージャのパラメータ

パラメータ	説明
plan	リソース・プランの名前
group_or_subplan	グループの名前またはサブプランの名前

CREATE_PENDING_AREA プロシージャ

このプロシージャによって、リソース・マネージャ・オブジェクトに変更を加えます。

プラン・スキーマへのすべての変更は、保留領域内で行う必要があります。保留領域は、プラン・スキーマを変更するためのスクラッチ領域とみなすことができます。管理者は、この保留領域を作成し、必要に応じて変更を加え、場合によってその変更を検証します。そして、その実行が完了したときのみ、その変更内容がアクティブになります。

保留領域がアクティブな間は、変更された現行のプラン・スキーマを適切なユーザー・ビューから選択して、いつでも表示できます。

現行の変更を中止する場合は、いつでも保留領域を消去できます。また、VALIDATE プロシージャをコールして、変更が有効になっているかどうかを確認できます。変更は、エントリ・グループの一貫性を維持するための指定の順序で行う必要はありません。これらのチェックは、保留領域が実行されるときにも暗黙的に行われます。

注意： Oracle では、孤立したコンシューマ・グループ（つまり、そのコンシューマ・グループを参照するプラン・ディレクティブがないコンシューマ・グループ）が可能です。これは、現在は使用しないが将来使用するコンシューマ・グループを管理者があらかじめ作成できるようにするためです。

構文

```
DEMS_RESOURCE_MANAGER.CREATE_PENDING_AREA;
```

パラメータ

なし

使用上の注意

次のルールを厳守してください。これらのルールは、VALIDATE または SUBMIT プロシージャが実行されるたびにチェックされます。

1. プラン・スキーマにループがないこと。
2. プラン・ディレクティブが参照するすべてのプランとコンシューマ・グループがあること。
3. すべてのプランに、プランまたはコンシューマ・グループのいずれかを参照するプラン・ディレクティブがあること。
4. リソース割当て方法が EMPHASIS の場合は、指定レベルでのパーセントの合計が 100 を超えないこと。

5. アクティブなインスタンスでトップレベルのプランとして現在使用されているプランを削除しないこと。
6. Oracle8i の場合、プラン・ディレクティブのパラメータ `parallel_degree_limit_pl` は、コンシューマ・グループ（サブプランではなく）を参照するプラン・ディレクティブでのみ表示されます。
7. 指定のプランでのプラン・ディレクティブは 32 を超えないこと（つまり、プランは 32 を超える子を持つことはできません）。
8. アクティブなプラン・スキーマ内のコンシューマ・グループは 32 を超えないこと。
9. プランとコンシューマ・グループは同じ名前領域を使用するため、コンシューマ・グループと同じ名前のプランがないこと。
10. アクティブなプラン・スキーマ内に `OTHER_GROUPS` に対するプラン・ディレクティブがあること。これにより、現在アクティブなプランがカバーしていないセッションに `OTHER_GROUPS` ディレクティブが指定したリソースが割り当てられます。

`VALIDATE` または `SUBMIT` プロシージャによるチェック時に、前述のルールのいずれかに違反していると、それを通知するエラー・メッセージが戻されます。変更して問題を修正し、`VALIDATE` または `SUBMIT` プロシージャを再発行できます。

VALIDATE_PENDING_AREA プロシージャ

このプロシージャによって、リソース・マネージャに対する保留中の変更内容を検証します。

構文

```
DBMS_RESOURCE_MANAGER.VALIDATE_PENDING_AREA;
```

パラメータ

なし

CLEAR_PENDING_AREA プロシージャ

このプロシージャによって、リソース・マネージャに対する保留中の変更内容を消去します。

構文

```
DBMS_RESOURCE_MANAGER.CLEAR_PENDING_AREA;
```

パラメータ

なし

SUBMIT_PENDING_AREA プロシージャ

このプロシージャによって、リソース・マネージャに対する保留中の変更を発行します。変更内容を検証してコミットした後（その変更内容が有効な場合）、保留領域を消去します。

注意： SUBMIT_PENDING_AREA へのコールは、VALIDATE_PENDING_AREA が成功していても失敗する場合があります。これは、削除するプランがインスタンスによって VALIDATE_PENDING_AREA へのコール後、SUBMIT_PENDING_AREA へのコールまでにロードされると発生する可能性があります。

構文

```
DBMS_RESOURCE_MANAGER.SUBMIT_PENDING_AREA;
```

パラメータ

なし

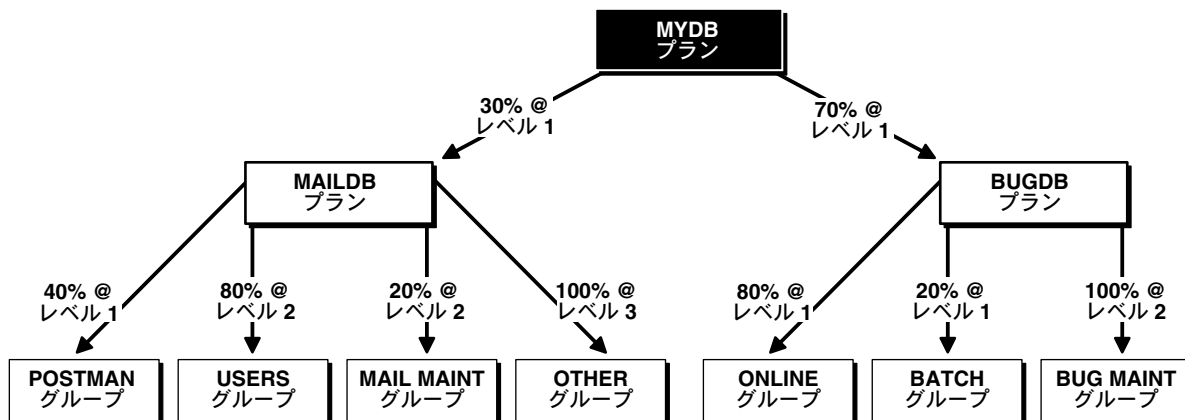
例

プランの利点の 1 つは、相互に参照できることです。プランのエントリは、コンシューマ・グループまたはサブプランのいずれかになります。次に、有効な CPU プラン・ディレクティブのセット例を示します。

表 42-12 MYDB PLAN CPU プラン・ディレクティブ

サブプラン / グループ	CPU_Level 1
MAILDB プラン	30%
BUGDB プラン	70%

これらのプラン・ディレクティブが有効で、すべてのコンシューマ・グループに実行可能なセッションが無限にある場合、MAILDB プランには使用可能な CPU リソースの 30% が割り当てられ、一方、BUGDB プランにはその 70% が割り当てられます。これをさらに分割すると、"Postman" コンシューマ・グループにあるセッションは 12%（30% の内の 40%）の時間を実行し、"Online" コンシューマ・グループにあるセッションは 56%（70% の内の 80%）の時間を実行します。次の図は、このシナリオを表しています。



次の説明では、コンシューマ・グループはアクティブなセッションです。つまり、セッションはリソース・コンシューマ・グループに所属し、このコンシューマ・グループは、処理するリソースの割当てを決定するためにプランが使用します。

CPU プラン・ディレクティブのマルチプラン（1 つ以上のサブプランを持つプラン）定義は、各プランが自分自身をエンティティとして所有するため、1 セットのプラン・ディレクティブを持つ単一プランに縮小できません。プランまたはサブプランに割り当てられた CPU 量は、アクティブ・セッションを持つコンシューマ・グループがそのプランに含まれていない限り、そのプラン内でのみ使用されます。したがって、この例では、Bug Maintenance グループが CPU 量をまったく使用しなかった場合はそのプラン内でリサイクルされるので、BUGDB プラン内のレベル 1 に戻ります。前述の例で、マルチプラン定義が複数のコンシューマ・グループを持つ単一プランに縮小された場合は、Bug Maintenance Group で使用されていない CPU 量を明示的にリサイクルする方法はありません。CPU 量はグローバルにリサイクルされるので、MAIL セッションにもそれを使用する機会が与えられます。

データベースのリソースは、複数のアプリケーション間の高いレベルでパーティション化でき、アプリケーション内で再パーティション化できます。アプリケーション内の指定のグループで、割り当てられたすべてのリソースが必要ない場合、そのリソースは、同じアプリケーション内でのみ再パーティション化されます。

次の例では、デフォルトのプランとコンシューマ・グループの割当て方法が使用されます。

```
BEGIN
DBMS_RESOURCE_MANAGER.CREATE_PENDING_AREA();
DBMS_RESOURCE_MANAGER.CREATE_PLAN(PLAN => 'bugdb_plan',
    COMMENT => 'Resource plan/method for bug users sessions');
DBMS_RESOURCE_MANAGER.CREATE_PLAN(PLAN => 'maildb_plan',
    COMMENT => 'Resource plan/method for mail users sessions');
DBMS_RESOURCE_MANAGER.CREATE_PLAN(PLAN => 'mydb_plan',
    COMMENT => 'Resource plan/method for bug and mail users sessions');
DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP(CONSUMER_GROUP => 'Bug_Online_group',
    COMMENT => 'Resource consumer group/method for online bug users sessions');
DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP(CONSUMER_GROUP => 'Bug_Batch_group',
    COMMENT => 'Resource consumer group/method for bug users sessions who run batch jobs');
DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP(CONSUMER_GROUP => 'Bug_Maintenance_group',
    COMMENT => 'Resource consumer group/method for users sessions who maintain
the bug db');
DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP(CONSUMER_GROUP => 'Mail_users_group',
    COMMENT => 'Resource consumer group/method for mail users sessions');
DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP(CONSUMER_GROUP => 'Mail_Postman_group',
    COMMENT => 'Resource consumer group/method for mail postman');
DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP(CONSUMER_GROUP => 'Mail_Maintenance_group',
    COMMENT => 'Resource consumer group/method for users sessions who maintain the mail
db');
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(PLAN => 'bugdb_plan', GROUP_OR_SUBPLAN =>
'Bug_Online_group',
    COMMENT => 'online bug users sessions at level 1', CPU_P1 => 80, CPU_P2=> 0,
    PARALLEL_DEGREE_LIMIT_P1 => 8);
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(PLAN => 'bugdb_plan', GROUP_OR_SUBPLAN =>
'Bug_Batch_group',
    COMMENT => 'batch bug users sessions at level 1', CPU_P1 => 20, CPU_P2 => 0,
    PARALLEL_DEGREE_LIMIT_P1 => 2);
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(PLAN => 'bugdb_plan', GROUP_OR_SUBPLAN =>
'Bug_Maintenance_group',
    COMMENT => 'bug maintenance users sessions at level 2', CPU_P1 => 0, CPU_P2 => 100,
    PARALLEL_DEGREE_LIMIT_P1 => 3);
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(PLAN => 'bugdb_plan', GROUP_OR_SUBPLAN =>
'OTHER_GROUPS',
    COMMENT => 'all other users sessions at level 3', CPU_P1 => 0, CPU_P2 => 0, CPU_P3 =>
100);
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(PLAN => 'maildb_plan', GROUP_OR_SUBPLAN =>
'Mail_Postman_group',
    COMMENT => 'mail postman at level 1', CPU_P1 => 40, CPU_P2 => 0,
    PARALLEL_DEGREE_LIMIT_P1 => 4);
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(PLAN => 'maildb_plan', GROUP_OR_SUBPLAN =>
'Mail_users_group',
    COMMENT => 'mail users sessions at level 2', CPU_P1 => 0, CPU_P2 => 80,
    PARALLEL_DEGREE_LIMIT_P1 => 4);
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(PLAN => 'maildb_plan', GROUP_OR_SUBPLAN =>
'Mail_Maintenance_group',
```



```
COMMENT => 'mail maintenance users sessions at level 2', CPU_P1 => 0, CPU_P2 => 20,
PARALLEL_DEGREE_LIMIT_P1 => 2);
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(PLAN => 'maildb_plan', GROUP_OR_SUBPLAN =>
'OTHER_GROUPS',
COMMENT => 'all other users sessions at level 3', CPU_P1 => 0, CPU_P2 => 0, CPU_P3 =>
100);
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(PLAN => 'mydb_plan', GROUP_OR_SUBPLAN =>
'maildb_plan',
COMMENT=> 'all mail users sessions at level 1', CPU_P1 => 30);
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(PLAN => 'mydb_plan', GROUP_OR_SUBPLAN =>
'bugdb_plan',
COMMENT => 'all bug users sessions at level 1', CPU_P1 => 70);
DBMS_RESOURCE_MANAGER.VALIDATE_PENDING_AREA();
DBMS_RESOURCE_MANAGER.SUBMIT_PENDING_AREA();
end;
```

検証は SUBMIT_PENDING_AREA で暗黙的に行われるので、前述の VALIDATE_PENDING_AREA へのコールはオプションです。

SET_INITIAL_CONSUMER_GROUP プロシージャ

ユーザーの初期コンシューマ・グループは、そのユーザーが作成したセッションが最初に所
属しているコンシューマ・グループです。このプロシージャは、ユーザーに対して、初期の
リソース・コンシューマ・グループを設定します。

構文

```
DBMS_RESOURCE_MANAGER.SET_INITIAL_CONSUMER_GROUP (
    user          IN VARCHAR2,
    consumer_group IN VARCHAR2);
```

パラメータ

表 42-13 SET_INITIAL_CONSUMER_GROUP プロシージャのパラメータ

パラメータ	説明
user	ユーザーの名前
consumer_group	ユーザーの初期コンシューマ・グループ

使用上の注意

このプロシージャを実行するためには、ADMINISTER_RESOURCE_MANAGER または ALTER
USER システム権限が必要です。ユーザーの初期コンシューマ・グループが設定される前に、
ユーザーまたは PUBLIC に対して、コンシューマ・グループへのスイッチ権限が直接付与さ

れている必要があります。初期コンシューマ・グループに対するスイッチ権限は、そのユーザーに付与されているロールから与えることはできません。

注意： この方法は、ALTER USER DEFAULT ROLE に対する方法に類似しています。

ユーザーの初期コンシューマ・グループが設定されていない場合、そのユーザーの初期コンシューマ・グループは、自動的にコンシューマ・グループ DEFAULT_CONSUMER_GROUP になります。

DEFAULT_CONSUMER_GROUP は PUBLIC に付与されたスイッチ権限を持つため、すべてのユーザーはこのコンシューマ・グループに対するスイッチ権限を自動的に付与されます。コンシューマ・グループを削除するとき、削除するグループを初期コンシューマ・グループとしていたすべてのユーザーは、DEFAULT_CONSUMER_GROUP を初期コンシューマ・グループとします。削除するコンシューマ・グループに所属している現行のアクティブなセッションは、すべて DEFAULT_CONSUMER_GROUP に切り替えられます。

SWITCH_CONSUMER_GROUP_FOR_SESS プロシージャ

このプロシージャによって、指定のセッションのリソース・コンシューマ・グループを変更します。また、トップレベルのユーザー・セッションに関連する（PQ）スレーブ・セッションのコンシューマ・グループも変更します。

構文

```
DBMS_RESOURCE_MANAGER.SWITCH_CONSUMER_GROUP_FOR_SESS (  
    session_id      IN NUMBER,  
    session_serial  IN NUMBER,  
    consumer_group  IN VARCHAR2);
```

パラメータ

表 42-14 SWITCH_CONSUMER_GROUP_FOR_SESS プロシージャのパラメータ

パラメータ	説明
session_id	ビュー V\$SESSION での SID 列
session_serial	ビュー V\$SESSION での SERIAL# 列
consumer_group	切り替えるコンシューマ・グループの名前

SWITCH_CONSUMER_GROUP_FOR_USER プロシージャ

このプロシージャによって、指定のユーザー ID を持つすべてのセッションに対するリソース・コンシューマ・グループを変更します。また、トップレベルのユーザー・セッションに関連する（PQ）スレーブ・セッションのコンシューマ・グループも変更します。

構文

```
DBMS_RESOURCE_MANAGER.SWITCH_CONSUMER_GROUP_FOR_USER (  
    user          IN VARCHAR2,  
    consumer_group IN VARCHAR2);
```

パラメータ

表 42-15 SWITCH_CONSUMER_GROUP_FOR_USER プロシージャのパラメータ

パラメータ	説明
user	ユーザーの名前
consumer_group	切り替えるコンシューマ・グループの名前

使用上の注意

SWITCH_CONSUMER_GROUP_FOR_SESS プロシージャと SWITCH_CONSUMER_GROUP_FOR_USER プロシージャによって、特定のセッションまたはユーザーの CPU リソース割当てを増減します。これにより、UNIX の nice コマンドと類似した機能性が提供されます。

これらのプロシージャは、新規に指定されたコンシューマ・グループにセッションを即時に移動します。

DBMS_RESOURCE_MANAGER_PRIVS

DBMS_RESOURCE_MANAGER_PRIVS パッケージは、リソース・マネージャに関連付けられている権限をメンテナンスします。

関連項目： データベース・リソース・マネージャの使用方法的詳細は、『Oracle8i 管理者ガイド』を参照してください。

サブプログラムの要約

表 43-1 DBMS_RESOURCE_MANAGER_PRIVS パッケージのサブプログラム

サブプログラム	説明
43-2 ページの GRANT_SYSTEM_PRIVILEGE プロシージャ	システム権限を付与します。
43-3 ページの REVOKE_SYSTEM_PRIVILEGE プロシージャ	システム権限を取り消します。
43-4 ページの GRANT_SWITCH_CONSUMER_GROUP プロ シージャ	リソース・コンシューマ・グループに切り替える権限 を付与します。
43-5 ページの REVOKE_SWITCH_CONSUMER_GROUP プロ シージャ	リソース・コンシューマ・グループに切り替える権限 を取り消します。

GRANT_SYSTEM_PRIVILEGE プロシージャ

このプロシージャは、ユーザーまたはロールにシステム権限を付与します。

構文

```
DBMS_RESOURCE_MANAGER_PRIVS.GRANT_SYSTEM_PRIVILEGE (  
    grantee_name    IN VARCHAR2,  
    privilege_name  IN VARCHAR2 DEFAULT 'ADMINISTER_RESOURCE_MANAGER',  
    admin_option    IN BOOLEAN);
```

パラメータ

表 43-2 GRANT_SYSTEM_PRIVILEGE プロシージャのパラメータ

パラメータ	説明
grantee_name	権限を付与されるユーザーまたはロールの名前
privilege_name	付与される権限の名前
admin_option	admin_option の付いた権限付与の場合は TRUE、そうでない場 合は FALSE

Oracle では現在、リソース・マネージャに対するシステム権限は ADMINISTER_RESOURCE_MANAGER のみ提供しています。データベース管理者には、admin option を伴うこのシステム権限があります。その権限の付与と取消しは、ユーザーまたはロールに対して行うことができます。admin option を伴うシステム権限を付与されたユーザーは、この権限を他のユーザーにも付与できます。

例

次のコールは、scott というユーザーに対して admin option を付けずにこの権限を付与します。

```
DBMS_RESOURCE_MANAGER_PRIVS.GRANT_SYSTEM_PRIVILEGE (  
    grantee_name => 'scott',  
    admin_option => FALSE);
```

REVOKE_SYSTEM_PRIVILEGE プロシージャ

このプロシージャは、ユーザーまたはロールからシステム権限を取り消します。

構文

```
DBMS_RESOURCE_MANAGER_PRIVS.REVOKE_SYSTEM_PRIVILEGE (  
    revokee_name    IN VARCHAR2,  
    privilege_name  IN VARCHAR2 DEFAULT 'ADMINISTER_RESOURCE_MANAGER');
```

パラメータ

表 43-3 REVOKE_SYSTEM_PRIVILEGE プロシージャのパラメータ

パラメータ	説明
revokee_name	権限を取り消すユーザーまたはロールの名前
privilege_name	取り消す権限の名前

例

次のコールは、ユーザー名 scott から ADMINISTER_RESOURCE_MANAGER を取り消します。

```
DBMS_RESOURCE_MANAGER_PRIVS.REVOKE_SYSTEM_PRIVILEGE ('scott');
```

GRANT_SWITCH_CONSUMER_GROUP プロシージャ

このプロシージャは、リソース・コンシューマ・グループに切り替える権限を付与します。

構文

```
DBMS_RESOURCE_MANAGER_PRIVS.GRANT_SWITCH_CONSUMER_GROUP (  
    grantee_name      IN VARCHAR2,  
    consumer_group    IN VARCHAR2,  
    grant_option      IN BOOLEAN);
```

パラメータ

表 43-4 GRANT_SWITCH_CONSUMER_GROUP プロシージャのパラメータ

パラメータ	説明
grantee_name	権限を付与されるユーザーまたはロールの名前
consumer_group	コンシューマ・グループの名前
grant_option	権限を付与されたユーザーが他のユーザーにアクセス権を付与できる場合は TRUE、そうでない場合は FALSE

使用上の注意

ユーザーに対して特定のコンシューマ・グループに切り替える許可を付与すると、そのユーザーは、即時に現行のコンシューマ・グループを新規のコンシューマ・グループに切り替えることができます。

ロールに対して特定のコンシューマ・グループに切り替える許可を付与すると、そのロールを付与され、そのロールを使用可能にしたユーザーは、即時に現行のコンシューマ・グループを新規のコンシューマ・グループに切り替えることができます。

PUBLIC に対して特定のコンシューマ・グループに切り替える許可を付与すると、すべてのユーザーがそのコンシューマ・グループに切り替えることができます。

grant_option パラメータが TRUE の場合、コンシューマ・グループに対するスイッチ権限が付与されたユーザーは、そのコンシューマ・グループに対するスイッチ権限を他のユーザーにも付与できます。

ユーザーの初期のコンシューマ・グループを設定するには、そのグループに対するスイッチ権限をユーザーに付与する必要があります。

関連項目： [第 42 章「DBMS_RESOURCE_MANAGER」](#)

例

```
DBMS_RESOURCE_MANAGER_PRIVS.GRANT_SWITCH_CONSUMER_GROUP (  
    'scott', 'mail_maintenance_group', true);  
  
DBMS_RESOURCE_MANAGER.SET_INITIAL_CONSUMER_GROUP (  
    'scott', 'mail_maintenance_group');
```

REVOKE_SWITCH_CONSUMER_GROUP プロシージャ

このプロシージャは、リソース・コンシューマ・グループに切り替える権限を取り消します。

構文

```
DBMS_RESOURCE_MANAGER_PRIVS.REVOKE_SWITCH_CONSUMER_GROUP (  
    revokee_name    IN VARCHAR2,  
    consumer_group  IN VARCHAR2);
```

パラメータ

表 43-5 REVOKE_SWITCH_CONSUMER_GROUP プロシージャのパラメータ

パラメータ	説明
revokee_name	アクセス権を取り消すユーザーまたはロールの名前
consumer_group	コンシューマ・グループの名前

使用上の注意

特定のコンシューマ・グループに対するスイッチ権限をユーザーから取り消した後で、そのユーザーがそのコンシューマ・グループに切り替えようとしても失敗します。

ユーザーから初期のコンシューマ・グループを取り消した場合、そのユーザーは、ログイン時に自動的に DEFAULT_CONSUMER_GROUP コンシューマ・グループに所属します。

コンシューマ・グループに対するスイッチ権限をロールから取り消すと、そのロールを介してのみコンシューマ・グループに対するスイッチ権限を持っていたユーザーは、以降そのコンシューマ・グループに切り替えることはできません。

コンシューマ・グループに対するスイッチ権限を PUBLIC から取り消すと、それまで PUBLIC を介してのみコンシューマ・グループを使用できたユーザーは、以降そのコンシューマ・グループに切り替えることはできません。

例

次の例は、mail_maintenance_group に切り替える権限を scott から取り消します。

```
DEMS_RESOURCE_MANAGER_PRIVS.REVOKE_SWITCH_CONSUMER_GROUP (  
    'scott', 'mail_maintenance_group');
```

DBMS_RLS パッケージには、ファイングレイン・アクセス・コントロールの管理インタフェースが含まれています。

注意： DBMS_RLS は、Enterprise Edition でのみ使用できます。

動的な述語

ファイングレイン・アクセス・コントロールをサポートする機能は、動的な述語に基づいています。セキュリティ・ルールはビューに埋め込まれていませんが、ベース表またはビューが DML 文で参照される、文の解析時に取得されます。

表またはビューに対する動的な述語は PL/SQL ファンクションが生成し、PL/SQL インタフェースを介してセキュリティ・ポリシーに関連付けられます。例は次のとおりです。

```
DBMS_RLS.ADD_POLICY (  
    'scott', 'emp', 'emp_policy', 'secusr', 'emp_sec', 'select');
```

SCOTT スキーマの下にある EMP 表が問合せまたは副問合せ (SELECT) で参照されるたびに、サーバーは、EMP_SEC ファンクション (SECUSR スキーマの下にある) をコールします。このファンクションは、EMP_POLICY ポリシーについて、カレント・ユーザーに固有の述語を戻します。ポリシー・ファンクションは、ファンクション・コール時に使用可能なすべてのセッション環境変数に基づいて述語を生成できます。これらの変数は通常、アプリケーション・コンテキストのフォームで表示されます。

次に、サーバーは、テキストがある一時ビューを作成します。

```
SELECT * FROM scott.emp WHERE P1
```

ここで、P1 (SAL > 10000 や副問合せなど) は、EMP_SEC ファンクションから戻された述語です。サーバーは、EMP 表をビューとして処理し、ビューのテキストをデータ・ディクショナリからではなく一時ビューから取得すること以外は、通常のビューと同様にビューの展開を行います。

述語に副問合せがある場合は、ポリシー・ファンクションの所有者 (定義者) を使用して副問合せ内のオブジェクトを解決し、そのオブジェクトのセキュリティをチェックします。つまり、ポリシー保護されたオブジェクトへのアクセス権限を持つユーザーには、ポリシーについての知識は不要です。このユーザーには、基礎となるセキュリティ・ポリシーに対するオブジェクト権限の付与は不要です。さらに、サーバーはファンクション定義者の権限でコールを行うため、このユーザーにはポリシー・ファンクションでの EXECUTE 権限は不要です。

注意： 一時ビューは、単一表または述語のみを持つビュー (つまり、JOIN、ORDER BY、GROUP BY などが無い) から導出されるため、親オブジェクトの更新可能性を保持することができます。

DBMS_RLS パッケージは、セキュリティ・ポリシーを削除したり、使用可能または使用禁止にするためのインタフェースも提供します。たとえば、次の PL/SQL 文を使用して、EMP_POLICY を削除または使用禁止にできます。

```
DBMS_RLS.DROP_POLICY('scott', 'emp', 'emp_policy');  
DBMS_RLS.ENABLE_POLICY('scott', 'emp', 'emp_policy', FALSE)
```

セキュリティ

一時ビューが副問合せを使用して作成されると、セキュリティ・チェックが実行されます。ポリシー・ファンクションを持ち、動的な述語を生成するスキーマは、セキュリティ・チェックとオブジェクト検索のために、一時ビューの定義者となります。

使用上の注意

DBMS_RLS プロシージャは、現行の DML トランザクションがある場合は、操作前にコミットします。ただし、プロシージャが DDL イベント・トリガーの内部にある場合、プロシージャは最初にコミットを実行しません。DDL トランザクションに関して、DBMS_RLS は、DDL トランザクションの一部です。

たとえば、ユーザーは CREATE TABLE のトリガーを作成できます。トリガー内部で、ALTER TABLE を介して列を追加でき、DBMS_RLS を介してポリシーを追加できます。これらすべての操作は、それぞれが DDL 文であっても、CREATE TABLE と同じトランザクション内にあります。CREATE TABLE は、トリガーが正常終了した場合のみ成功します。

サブプログラムの要約

表 44-1 DBMS_RLS のサブプログラム

サブプログラム	説明
44-4 ページの ADD_POLICY プロシージャ	ファイングレイン・アクセス・コントロールのポリシーを表またはビューに作成します。
44-6 ページの DROP_POLICY プロシージャ	ファイングレイン・アクセス・コントロールのポリシーを表またはビューから削除します。
44-6 ページの REFRESH_POLICY プロシージャ	ポリシーに関連付けられているすべてのキャッシュ済の文を再解析します。
44-7 ページの ENABLE_POLICY プロシージャ	ファイングレイン・アクセス・コントロールのポリシーを使用可能または使用禁止にします。

ADD_POLICY プロシージャ

このプロシージャは、ファイングレイン・アクセス・コントロールのポリシーを表またはビューに作成します。

トランザクションがある場合、その現行トランザクションはプロシージャによって操作の実行前にコミットを実行します。ただし、トランザクションが DDL イベント・トリガー内にある場合は、最初にコミットを実行しません。

関連項目： 44-3 ページの「[使用上の注意](#)」

コミットは、操作の最後にも実行されます。

構文

```
DEMS_RLS.ADD_POLICY (
    object_schema    IN VARCHAR2 := NULL,
    object_name      IN VARCHAR2,
    policy_name      IN VARCHAR2,
    function_schema  IN VARCHAR2 := NULL,
    policy_function  IN VARCHAR2,
    statement_types  IN VARCHAR2 := NULL,
    update_check     IN BOOLEAN  := FALSE,
    enable           IN BOOLEAN  := TRUE);
```

パラメータ

表 44-2 ADD_POLICY プロシージャのパラメータ

パラメータ	説明
object_schema	表またはビューを含んでいるスキーマ（NULL の場合はログオン・ユーザー）。
object_name	ポリシーを追加する表またはビューの名前。
policy_name	追加するポリシーの名前。この名前は、表またはビュー内に対して一意である必要があります。
function_schema	ポリシー・ファンクションのスキーマ（NULL の場合はログオン・ユーザー）。
policy_function	ポリシーの述語を生成するファンクションの名前。ファンクションがパッケージ内で定義されている場合、パッケージ名は必ず存在する必要があります。
statement_types	ポリシーを適用する文タイプ SELECT、INSERT、UPDATE および DELETE を任意に組み合わせることができます。デフォルトでは、すべてのタイプが適用されます。

表 44-2 ADD_POLICY プロシージャのパラメータ

パラメータ	説明
update_check	文タイプ INSERT または UPDATE に対するオプションの引数。デフォルトは FALSE です。update_check を TRUE に設定すると、サーバーは、挿入または更新後の値に対してもポリシーをチェックします。
enable	ポリシーの追加時に、そのポリシーを使用可能にするかどうかを示します。デフォルトは TRUE です。

使用上の注意

- SYS は、セキュリティ・ポリシーの制約を受けません。
- 動的な述語を生成するポリシー・ファンクションは、サーバーがコールします。次の例は、ファンクションのインタフェースを示します。

```
FUNCTION policy_function (object_schema IN VARCHAR2, object_name VARCHAR2)
RETURN VARCHAR2
--- object_schema is the schema owning the table of view.
--- object_name is the name of table of view that the policy will apply.
```

ポリシー・ファンクションが戻す述語の最大長は、2,000 バイトです。

- ポリシー・ファンクションには、WNDS（データベースへの書き込み禁止状態）の純粹さレベルが必要です。

関連項目： RESTRICT_REFERENCES プラグマの詳細は、『Oracle8i アプリケーション開発者ガイド 基礎編』を参照してください。

- 同じオブジェクトにある複数のポリシーから生成された動的な述語は、全述語の論理積（AND 条件）の結合効果があります。
- セキュリティ・チェックとオブジェクト検索は、動的な述語の副問合せで、オブジェクトのポリシー・ファンクションの所有者に対して実行されます。
- ファンクションが長さ 0（ゼロ）の述語を戻す場合は、ポリシーについてカレント・ユーザーに適用する制限はないと解釈されます。
- 述語で表の別名が必要な場合（たとえば、親オブジェクトがタイプ表の場合）は、表またはビューの名前自体を別名として使用する必要があります。サーバーは、一時ビューを "select c1, c2, ... from tab where <predicate>" のように構成します。
- ファンクションの妥当性チェックは、インストレーションを容易にしたり、インポートまたはエクスポート時に発生するその他の依存する問題を軽減するため、実行時に行われます。

DROP_POLICY プロシージャ

このプロシージャは、ファイングレイン・アクセス・コントロールのポリシーを表またはビューから削除します。

トランザクションがある場合、その現行トランザクションはプロシージャによって操作の実行前にコミットを実行します。ただし、トランザクションが DDL イベント・トリガー内にある場合は、最初にコミットを実行しません。

関連項目： 44-3 ページの「[使用上の注意](#)」

コミットは、操作の最後にも実行されます。

構文

```
DBMS_RLS.DROP_POLICY (  
    object_schema IN VARCHAR2 := NULL,  
    object_name   IN VARCHAR2,  
    policy_name   IN VARCHAR2);
```

パラメータ

表 44-3 DROP_POLICY プロシージャのパラメータ

パラメータ	説明
object_schema	表またはビューを含んでいるスキーマ（NULL の場合はログオン・ユーザー）
object_name	表またはビューの名前
policy_name	表またはビューから削除するポリシーの名前

REFRESH_POLICY プロシージャ

このプロシージャは、ポリシーに関連付けられたすべてのキャッシュ済の文を再解析します。これにより、ポリシーへの最新の変更内容がプロシージャの実行直後に有効になります。

トランザクションがある場合、その現行トランザクションはプロシージャによって操作の実行前にコミットを実行します。ただし、トランザクションが DDL イベント・トリガー内にある場合は、最初にコミットを実行しません。

関連項目： 44-3 ページの「[使用上の注意](#)」

コミットは、操作の最後にも実行されます。

構文

```
DBMS_RLS.REFRESH_POLICY (  
    object_schema IN VARCHAR2 := NULL,  
    object_name   IN VARCHAR2 := NULL,  
    policy_name   IN VARCHAR2 := NULL);
```

パラメータ

表 44-4 REFRESH_POLICY プロシージャのパラメータ

パラメータ	説明
object_schema	表またはビューを含んでいるスキーマ
object_name	ポリシーが関連付けられている表またはビューの名前
policy_name	リフレッシュするポリシーの名前

エラー

使用禁止になっているポリシーをリフレッシュしようとする、プロシージャはエラーを戻します。

ENABLE_POLICY プロシージャ

このプロシージャは、ファイングレイン・アクセス・コントロールのポリシーを使用可能または使用禁止にします。ポリシーは、その作成時に使用可能になっています。

トランザクションがある場合、その現行トランザクションはプロシージャによって操作の実行前にコミットを実行します。ただし、トランザクションが DDL イベント・トリガー内にある場合は、最初にコミットを実行しません。

関連項目： 44-3 ページの「[使用上の注意](#)」

コミットは、操作の最後にも実行されます。

構文

```
DBMS_RLS.ENABLE_POLICY (  
    object_schema IN VARCHAR2 := NULL,  
    object_name   IN VARCHAR2,  
    policy_name   IN VARCHAR2,  
    enable        IN BOOLEAN);
```

パラメータ

表 44-5 ENABLE_POLICY プロシージャのパラメータ

パラメータ	説明
object_schema	表またはビューを含んでいるスキーマ（NULL の場合はログオン・ユーザー）
object_name	ポリシーが関連付けられている表またはビューの名前
policy_name	使用可能または使用禁止にするポリシーの名前
enable	ポリシーを使用可能にする場合は TRUE、使用禁止にする場合は FALSE

例

次の例は、ファイングレイン・アクセス・コントロールのポリシーを実施するために必要なステップを示します。

Oracle HR アプリケーションで、PER_PEOPLE は PER_ALL_PEOPLE 表に対するビューで、両方のオブジェクトとも APPS スキーマの下にあります。

```
CREATE TABLE per_all_people
  (person_id NUMBER(15),
   last_name VARCHAR2(30),
   emp_no VARCHAR2(15), ...);
CREATE VIEW per_people AS
  SELECT * FROM per_all_people;
```

社内のユーザー・ロールに基づいて、PER_PEOPLE ビューへのアクセスを制限するセキュリティ・ポリシーが必要です。ポリシーの述語は、HR_SECURITY パッケージにある SECURE_PERSON ファンクションで生成できます。このパッケージは、スキーマ APPS の下にあり、HR アプリケーションに関連するすべてのセキュリティ・ポリシーをサポートするファンクションが含まれています。また、すべてのアプリケーション・コンテキストは、APPS_SEC 名前領域の下にあります。

```
CREATE PACKAGE BODY hr_security IS
  FUNCTION secure_person(obj_schema VARCHAR2, obj_name VARCHAR2)
    RETURN VARCHAR2 IS
    d_predicate VARCHAR2(2000);
BEGIN
  -- for users with HR_ROLE set to EMP, map logon user name
  -- to employee id. FND_USER table stores relationship
  -- among database users, application users,
  -- and people held in the HR person table.
  IF SYS_CONTEXT('apps_sec', 'hr_role') = 'EMP' THEN
    d_predicate = 'person_id IN
```

```

        (SELECT employee_id FROM apps.fnd_user
        WHERE user_name = SYS_CONTEXT('userenv', 'session_
user')));
    -- for users with HR_ROLE set to MGR (manager), map
    -- security profile id to a list of employee id that
    -- the user can access
    ELSE IF SYS_CONTEXT('apps_sec', 'hr_role') = 'MGR' THEN
        d_predicate = 'person_id IN
        (SELECT ppl.employee_id FROM per_person_list ppl WHERE
        ppl.security_profile_id = SYS_CONTEXT('apps_sec',
        'security_profile_id'))
        OR EXISTS (SELECT NULL FROM apps.per security_profiles psp
WHERE
        SYS_CONTEXT('apps_sec', 'security_profile_id') =
        psp.security_profile_id AND psp.view_all_flag = 'Y'))';
    ELSE
        d_predicate = '1=2'; -- deny access to other users, may use something
like 'keycol=null'
    END IF;
    RETURN d_predicate;
END secure_person;
END hr_security;

```

次のステップでは、PER_PEOPLE ビューのポリシー（ここでは PER_PEOPLE_SEC）を、動的な述語を生成する HR_SECURITY.SECURE_PERSON ファンクションに関連付けます。

```

DBMS_RLS.ADD_POLICY('apps', 'per_people', 'per_people_sec', 'apps'
        'hr_security.secure_person', 'select, update, delete');

```

ここで、PER_PEOPLE ビューを含んだ SELECT、UPDATE および DELETE 文は、アプリケーション・コンテキスト HR_ROLE の値に基づいて 3 つの述語から 1 つを取得します。

注意： PER_ALL_PEOPLE 表を保護するセキュリティ・ファンクションと同じファンクションを使用して、PER_ADDRESSES 表を保護する動的な述語も生成できます。これは、どちらの表も、データへのアクセスを制限するポリシーが同じためです。

DBMS_ROWID パッケージによって、ユーザーは ROWID を作成し、PL/SQL プログラムと SQL 文から ROWID に関する情報を取得できます。ベース 64 文字の外部 ROWID を解析するためのコードを記述しないで、データ・ブロック番号、オブジェクト番号および他の ROWID コンポーネントを検索できます。

注意： DBMS_ROWID は、ユニバーサルな ROWID (UROWID) とともに使用しません。

使用上の注意

このパッケージにあるファンクションの一部は、ROWID などの単一のパラメータを必要とします。このパラメータは、1 つの文字または PL/SQL ROWID で、その内容は必要に応じて制限または拡張されます。

DBMS_ROWID のファンクションとプロシージャは PL/SQL コードからコールでき、そのファンクションは SQL 文で使用することもできます。

注意： ROWID_INFO は 1 つのプロシージャです。このプロシージャは、PL/SQL コードでのみ使用できます。

DBMS_ROWID パッケージのファンクションは、他の組込み式の SQL ファンクションと同じように使用できます。つまり、式が使用できればどこでも使用できます。次の例では、EMP 表にある単一行のブロック番号のみを戻すために、ROWID_BLOCK_NUMBER が使用されています。

```
SELECT dbms_rowid.rowid_block_number(rowid)
FROM emp
WHERE ename = 'KING';
```

RESTRICT_REFERENCES プラグマ使用上のトラブルシューティング

"ORA: 452.0, サブプログラム *string* が対応付けられたプラグマに違反しています。" (*string* に文字列が入ります) のエラーが発生した場合は、次の理由が考えられます。

- 現行のプロシージャまたはファンクションに問題があるため
- プラグマなしでプロシージャまたはファンクションをコールしたか、または制限が不十分なプラグマでプロシージャまたはファンクションをコールしたため
- パッケージの初期化コードに関連したパッケージ・プロシージャまたはファンクションをコールしたか、またはデフォルト値を設定するパッケージ・プロシージャまたはファンクションをコールしたため

PL/SQL の例

次の例では、EMP 表にある行の ROWID を戻し、DBMS_ROWID パッケージの ROWID_OBJECT ファンクションを使用して、その ROWID からデータ・オブジェクト番号を抽出して表示します。

```
DECLARE
    object_no    INTEGER;
    row_id       ROWID;
    ...
BEGIN
```

```
SELECT ROWID INTO row_id FROM emp
WHERE empno = 7499;
object_no := dbms_rowid.rowid_object(row_id);
dbms_output.put_line('The obj. # is ' || object_no);
...
```

要件

このパッケージは、パッケージ所有者（sys）ではなく、コール・ユーザーの権限で実行されます。

ROWID のタイプ

RESTRICTED	制限 ROWID
EXTENDED	拡張 ROWID

例：

```
rowid_type_restricted constant integer := 0;
rowid_type_extended   constant integer := 1;
```

注意： 拡張 ROWID は、Oracle8i 以降でのみ使用されます。

ROWID の検証結果

VALID	有効 ROWID
INVALID	無効 ROWID

例：

```
rowid_is_valid   constant integer := 0;
rowid_is_invalid constant integer := 1;
```

オブジェクト型

UNDEFINED	（制限 ROWID に対する）オブジェクト番号が定義されていません。
-----------	------------------------------------

例：

```
rowid_object_undefined constant integer := 0;
```

ROWID の変換タイプ

- INTERNAL ROWID タイプの列から、またはその列への変換
- EXTERNAL 文字列形式から、または文字列形式への変換

例 :

```
rowid_convert_internal constant integer := 0;
rowid_convert_external constant integer := 1;
```

例外

- ROWID_INVALID 無効な ROWID 形式。
- ROWID_BAD_BLOCK ブロックがファイルの最後を超えています。

例 :

```
ROWID_INVALID exception;
pragma exception_init(ROWID_INVALID, -1410);

ROWID_BAD_BLOCK exception;
pragma exception_init(ROWID_BAD_BLOCK, -28516);
```

サブプログラムの要約

表 45-1 DBMS_ROWID パッケージのサブプログラム

サブプログラム	説明
45-5 ページの ROWID_CREATE ファンクション	ROWID をテスト用に限って作成します。
45-6 ページの ROWID_INFO プロシージャ	ROWID のタイプとコンポーネントを戻します。
45-7 ページの ROWID_TYPE ファンクション	ROWID のタイプを戻します。0 は制限付き、1 は拡張です。
45-8 ページの ROWID_OBJECT ファンクション	拡張 ROWID のオブジェクト番号を戻します。
45-9 ページの ROWID_RELATIVE_FNO ファンクション	ROWID のファイル番号を戻します。
45-10 ページの ROWID_BLOCK_NUMBER ファンクション	ROWID のブロック番号を戻します。

表 45-1 DBMS_ROWID パッケージのサブプログラム

サブプログラム	説明
45-10 ページの ROWID_ROW_NUMBER ファンクション	行番号を戻します。
45-11 ページの ROWID_TO_ABSOLUTE_FNO ファンクション	特定の表にある行について、ROWID に関連する絶対 ファイル番号を戻します。
45-12 ページの ROWID_TO_EXTENDED ファンクション	ROWID を制限形式から拡張形式に変換します。
45-14 ページの ROWID_TO_RESTRICTED ファンクション	拡張 ROWID を制限形式に変換します。
45-14 ページの ROWID_VERIFY ファンクション	ROWID_TO_EXTENDED ファンクションが ROWID を適 切に拡張できるかどうかをチェックします。

ROWID_CREATE ファンクション

このファンクションによって、コンポーネント部分をパラメータとして ROWID を作成します。

Oracle Server では、データベース内のデータを指す有効な ROWID を作成することしかできないため、このファンクションは、ROWID 操作のテストに役立ちます。

構文

```
DBMS_ROWID.ROWID_CREATE (  
    rowid_type      IN NUMBER,  
    object_number   IN NUMBER,  
    relative_fno    IN NUMBER,  
    block_number    IN NUMBER,  
    row_number      IN NUMBER)  
RETURN ROWID;
```

プラグマ

```
pragma RESTRICT_REFERENCES (rowid_create,WNDS,RNDS,WNPS,RNPS);
```

パラメータ

表 45-2 ROWID_CREATE ファンクションのパラメータ

パラメータ	説明
rowid_type	タイプ（制限または拡張）。 制限 ROWID については、rowid_type パラメータを 0 に設定します。拡張 ROWID を作成するには、1 を設定します。 rowid_type を 0 に指定すると、指定された object_number パラメータは無視され、ROWID_CREATE ファンクションは制限 ROWID を戻します。
object_number	データ・オブジェクト番号（制限の場合は ROWID_OBJECT_UNDEFINED）。
relative_fno	相対ファイル番号。
block_number	このファイル内のブロック番号。
file_number	このブロック内のファイル番号。

例

ダミーの拡張 ROWID を作成します。

```
my_rowid := DBMS_ROWID.ROWID_CREATE(1, 9999, 12, 1000, 13);
```

rowid_object ファンクションが戻す値を検索します。

```
obj_number := DBMS_ROWID.ROWID_OBJECT(my_rowid);
```

変数 obj_number には、9999 が入っています。

ROWID_INFO プロシージャ

このプロシージャは、ROWID のタイプ（制限または拡張）を含めた情報と、ROWID のコンポーネントを戻します。これはプロシージャで、SQL 文では使用できません。

構文

```
DBMS_ROWID.ROWID_INFO (  
  rowid_in      IN   ROWID,  
  rowid_type    OUT  NUMBER,  
  object_number OUT  NUMBER,  
  relative_fno  OUT  NUMBER,  
  block_number  OUT  NUMBER,  
  row_number    OUT  NUMBER);
```

プラグマ

```
pragma RESTRICT_REFERENCES (rowid_info,WNDS,RNDS,WNPS,RNPS);
```

パラメータ

表 45-3 ROWID_INFO プロシージャのパラメータ

パラメータ	説明
rowid_in	解釈する ROWID。ROWID が制限 (0) か、または拡張 (1) ROWID かを判別します。
rowid_type	タイプ (制限または拡張) を戻します。
object_number	データ・オブジェクト番号 (制限の場合は ROWID_OBJECT_UNDEFINED) を戻します。
relative_fno	相対ファイル番号を戻します。
block_number	このファイル内のブロック番号を戻します。
file_number	このブロック内のファイル番号を戻します。

関連項目： 45-7 ページの「[ROWID_TYPE ファンクション](#)」

例

この例では、ROWID_CREATE で作成した ROWID の値を読み込んで戻します。

```
DBMS_ROWID.ROWID_INFO(my_rowid, rid_type, obj_num,
    file_num, block_num, row_num);

DBMS_OUTPUT.PUT_LINE('The type is ' || rid_type);
DBMS_OUTPUT.PUT_LINE('Data object number is ' || obj_num);
-- and so on...
```

ROWID_TYPE ファンクション

このファンクションは、ROWID が制限 ROWID の場合は 0、拡張の場合は 1 を戻します。

構文

```
DBMS_ROWID.ROWID_TYPE (
    rowid_id IN ROWID)
RETURN NUMBER;
```

プラグマ

```
pragma RESTRICT_REFERENCES (rowid_type,WNDS,RNDS,WNPS,RNPS);
```

パラメータ

表 45-4 ROWID_TYPE ファンクションのパラメータ

パラメータ	説明
row_id	解釈する ROWID

例

```
IF DBMS_ROWID.ROWID_TYPE(my_rowid) = 1 THEN
    my_obj_num := DBMS_ROWID.ROWID_OBJECT(my_rowid);
```

ROWID_OBJECT ファンクション

このファンクションは、拡張 ROWID のデータ・オブジェクト番号を戻します。入力された ROWID が制限 ROWID の場合は、0 を戻します。

構文

```
DBMS_ROWID.ROWID_OBJECT (
    rowid_id IN ROWID)
RETURN NUMBER;
```

プラグマ

```
pragma RESTRICT_REFERENCES (rowid_object,WNDS,RNDS,WNPS,RNPS);
```

パラメータ

表 45-5 ROWID_OBJECT ファンクションのパラメータ

パラメータ	説明
row_id	解釈する ROWID

注意： 制限 ROWID については、ROWID_OBJECT_UNDEFINED の定数が戻されます。

例

```
SELECT dbms_rowid.rowid_object (ROWID)
FROM emp
WHERE empno = 7499;
```

ROWID_RELATIVE_FNO ファンクション

このファンクションは、IN パラメータで指定された ROWID の相対ファイル番号を戻します。（ファイル番号は表領域に関連しています。）

構文

```
DBMS_ROWID.ROWID_RELATIVE_FNO (
    rowid_id IN ROWID)
RETURN NUMBER;
```

プラグマ

```
pragma RESTRICT_REFERENCES (rowid_relative_fno,WNDS,RNDS,WNPS,RNPS);
```

パラメータ

表 45-6 ROWID_RELATIVE_FNO ファンクションのパラメータ

パラメータ	説明
row_id	解釈する ROWID

例

次の例の PL/SQL コードは、相対ファイル番号を戻します。

```
DECLARE
    file_number    INTEGER;
    rowid_val      ROWID;
BEGIN
    SELECT ROWID INTO rowid_val
    FROM dept
    WHERE loc = 'Boston';
    file_number :=
        dbms_rowid.rowid_relative_fno(rowid_val);
    ...
END;
```

ROWID_BLOCK_NUMBER ファンクション

このファンクションは、入力された ROWID のデータベース・ブロック番号を戻します。

構文

```
DBMS_ROWID.ROWID_BLOCK_NUMBER (  
    row_id IN ROWID)  
RETURN NUMBER;
```

プラグマ

```
pragma RESTRICT_REFERENCES (rowid_block_number, WNDS, RNDS, WNPS, RNPS);
```

パラメータ

表 45-7 ROWID_BLOCK_NUMBER ファンクションのパラメータ

パラメータ	説明
row_id	解釈する ROWID

例

次の SQL 文の例では、ROWID からブロック番号を選択し、別の表に挿入します。

```
INSERT INTO T2 (SELECT dbms_rowid.rowid_block_number(ROWID)  
    FROM some_table  
    WHERE key_value = 42);
```

ROWID_ROW_NUMBER ファンクション

このファンクションは、ROWID IN パラメータから行番号を抽出します。

構文

```
DBMS_ROWID.ROWID_ROW_NUMBER (  
    row_id IN ROWID)  
RETURN NUMBER;
```

プラグマ

```
pragma RESTRICT_REFERENCES (rowid_row_number, WNDS, RNDS, WNPS, RNPS);
```

パラメータ

表 45-8 ROWID_ROW_NUMBER ファンクションのパラメータ

パラメータ	説明
row_id	解釈する ROWID

例

行番号を選択します。

```
SELECT dbms_rowid.rowid_row_number(ROWID)
FROM emp
WHERE ename = 'ALLEN';
```

ROWID_TO_ABSOLUTE_FNO ファンクション

このファンクションは、指定のスキーマと表にある行について、ファイル番号が絶対番号である ROWID から、絶対ファイル番号を抽出します。スキーマ名とスキーマ・オブジェクト名（表名など）がこのファンクションの IN パラメータとして提供されます。

構文

```
DBMS_ROWID.ROWID_TO_ABSOLUTE_FNO (
    row_id      IN ROWID,
    schema_name IN VARCHAR2,
    object_name IN VARCHAR2)
RETURN NUMBER;
```

プラグマ

```
pragma RESTRICT_REFERENCES(rowid_to_absolute_fno,WNDS,WNPS,RNPS);
```

パラメータ

表 45-9 ROWID_TO_ABSOLUTE_FNO ファンクションのパラメータ

パラメータ	説明
row_id	解釈する ROWID
schema_name	表が含まれるスキーマの名前
object_name	表名

例

```
DECLARE
    abs_fno          INTEGER;
    rowid_val        CHAR(18);
    object_name      VARCHAR2(20) := 'EMP';
BEGIN
    SELECT ROWID INTO rowid_val
    FROM emp
    WHERE empno = 9999;
    abs_fno := dbms_rowid.rowid_to_absolute_fno(
        rowid_val, 'SCOTT', object_name);
```

注意： パーティション・オブジェクトの名前には、パーティション名やサブパーティション名ではなく、表名を指定する必要があります。

ROWID_TO_EXTENDED ファンクション

このファンクションは、ユーザーが指定するスキーマや表にある行をアドレス指定している制限形式の ROWID を、拡張 ROWID 形式に変換します。このファンクションは、今後、このパッケージから削除されて別の場所に移動する可能性があります。

構文

```
DBMS_ROWID.ROWID_TO_EXTENDED (
    old_rowid      IN ROWID,
    schema_name    IN VARCHAR2,
    object_name    IN VARCHAR2,
    conversion_type IN INTEGER)
RETURN ROWID;
```

プラグマ

```
pragma RESTRICT_REFERENCES(rowid_to_extended,WNDS,WNPS,RNPS);
```

パラメータ

表 45-10 ROWID_TO_EXTENDED ファンクションのパラメータ

パラメータ	説明
old_rowid	変換する ROWID
schema_name	表が含まれるスキーマの名前（オプション）
object_name	表名（オプション）

表 45-10 ROWID_TO_EXTENDED ファンクションのパラメータ

パラメータ	説明
conversion_type	rowid_convert_internal/external_convert_external (old_rowid が ROWID タイプの列に格納されていたか、または 文字列に格納されていたかによります)

戻り値

ROWID_TO_EXTENDED は、拡張文字列形式で ROWID を戻します。入力された ROWID が NULL の場合、ファンクションは NULL を戻します。ゼロ値 (00000000.0000.0000) の ROWID が提供されると、ゼロ値の制限 ROWID が戻されます。

例

SCOTT スキーマに RIDS と呼ばれる表があり、その表には、ROWID (制限) を保持する列 ROWID_COL と、SCOTT スキーマのその他の表を指す列 TABLE_COL が含まれていると仮定します。次の文を使用して、ROWID を拡張形式に変換できます。

```
UPDATE SCOTT.RIDS
  SET rowid_col =
    dbms_rowid.rowid_to_extended (
      rowid_col, 'SCOTT', 'TABLE_COL', 0);
```

使用上の注意

スキーマ名とオブジェクト名が IN パラメータとして提供されると、このファンクションは、指定の表における SELECT 認可レベルを検証し、表のデータ・オブジェクト番号を使用して、提供された制限 ROWID を拡張 ROWID に変換します。ROWID_TO_EXTENDED は値を戻しますが、このファンクションがコールされたとき、または拡張 ROWID が実際に使用されたときに、変換された ROWID が表内の有効な行を実際に参照していることは保証しません。

スキーマ名とオブジェクト名が提供されない場合 (NULL として渡された場合)、このファンクションは、提供された制限 ROWID が指定しているページをフェッチしようとします。この ROWID に格納されているファイル番号は、絶対ファイル番号として処理されます。このため、そのファイルが削除されていたり、その番号が移動前に再使用されている場合は問題が生じます。フェッチされたページが有効な表に属している場合、この表のデータ・オブジェクト番号は拡張 ROWID 値への変換時に使用されます。これは非常に効率が悪いため、ターゲット表が不明な場合に行う最終手段としてのみお勧めします。ユーザーは、変換された値の使用時まで、正しい表名を覚えておく必要があります。

拡張 ROWID 値が提供された場合、入力した拡張 ROWID のデータ・オブジェクト番号は、表名パラメータから計算されたデータ・オブジェクト番号と照合して検証されます。2つの番号が一致しない場合は、INVALID_ROWID 例外が発生します。一致する場合は、入力した ROWID が戻されます。

関連項目： [ROWID_VERIFY](#) ファンクションには、指定の ROWID を拡張形式に変換できるかどうかの判別方法が記述されています

ROWID_TO_RESTRICTED ファンクション

このファンクションは、拡張 ROWID を制限 ROWID 形式に変換します。

構文

```
DEMS_ROWID.ROWID_TO_RESTRICTED (  
    old_rowid      IN ROWID,  
    conversion_type IN INTEGER)  
RETURN ROWID;
```

プラグマ

```
pragma RESTRICT_REFERENCES (rowid_to_restricted,WNDS,RNDS,WNPS,RNPS);
```

パラメータ

表 45-11 ROWID_TO_RESTRICTED ファンクションのパラメータ

パラメータ	説明
old_rowid	変換する ROWID
conversion_type	内部または外部 - 戻される ROWID の形式 rowid_convert_internal/external_convert_external (戻される ROWID が ROWID タイプの列に格納されるか、または文 字列に格納されるかによります)

ROWID_VERIFY ファンクション

このファンクションは、ROWID を検証します。入力した制限 ROWID が、入力スキーマ名と表名を指定して、拡張形式に変換できる場合は、0 を返し、変換できない場合は 1 を返します。

注意： 次の例に示すように、このファンクションを SQL 文の WHERE 句で使用できます。

構文

```
DBMS_ROWID.ROWID_VERIFY (  
    rowid_in          IN ROWID,  
    schema_name       IN VARCHAR2,  
    object_name       IN VARCHAR2,  
    conversion_type   IN INTEGER  
    RETURN NUMBER;
```

プラグマ

```
pragma RESTRICT_REFERENCES (rowid_verify,WNDS,WNPS,RNPS);
```

パラメータ

表 45-12 ROWID_VERIFY ファンクションのパラメータ

パラメータ	説明
rowid_in	検証する ROWID
schema_name	表が含まれるスキーマの名前
object_name	表名
conversion_type	rowid_convert_internal/external_convert_external (old_rowid が ROWID タイプの列に格納されていたか、または 文字列に格納されていたかによります)

例

ROWID_TO_EXTENDED ファンクションの例にあるスキーマを考慮しながら、次の文を使用すると、無効な ROWID を変換前に検索できます。これにより、無効を事前に修正できます。

```
SELECT ROWID, rowid_col  
FROM SCOTT.RIDS  
WHERE dbms_rowid.rowid_verify(rowid_col, NULL, NULL, 0) =1;
```

関連項目： [第 63 章「UTL_RAW」](#) および [第 64 章「UTL_REF」](#)

DBMS_SESSION

このパッケージは、PL/SQL から `SQL ALTER SESSION` 文と `SET ROLE` 文へのアクセスおよび他のセッション情報へのアクセスを提供します。このパッケージを使用すると、作業環境とセキュリティ・レベルを設定できます。

要件

このパッケージは、パッケージ所有者 SYS ではなく、コール・ユーザーの権限で実行されま
す。

サブプログラムの要約

表 46-1 DBMS_SESSION のサブプログラム

サブプログラム	説明
46-3 ページの SET_ROLE プロシージャ	ロールを設定します。
46-3 ページの SET_SQL_TRACE プロシージャ	トレースをオンまたはオフにします。
46-4 ページの SET_NLS プロシージャ	各国語サポート（NLS）を設定します。
46-4 ページの CLOSE_DATABASE_LINK プロシージャ	データベース・リンクをクローズします。
46-5 ページの RESET_PACKAGE プロシージャ	セッション内のすべてのパッケージのインスタンス化 の解除をします。
46-6 ページの UNIQUE_SESSION_ID ファンクション	データベースに現在接続しているすべてのセッション に対して一意の識別子を戻します。
46-7 ページの IS_ROLE_ENABLED ファンクション	指定のロールがセッションで使用可能かどうかを判別 します。
46-7 ページの IS_SESSION_ALIVE ファンクション	指定されたセッションが有効かどうかを判別します。
46-8 ページの SET_CLOSE_CACHED_OPEN_CURSORS プロ シージャ	close_cached_open_cursors をオンまたはオフ にします。
46-8 ページの FREE_UNUSED_USER_MEMORY プロシージャ	大量のメモリーが必要な操作を実行した後で未使用の メモリーを再要求できます。
46-11 ページの SET_CONTEXT プロシージャ	コンテキスト属性の値を設定または再設定します。
46-11 ページの LIST_CONTEXT プロシージャ	現行のセッションについて、アクティブな名前領域と コンテキストのリストを戻します。
46-12 ページの SWITCH_CURRENT_CONSUMER_GROUP プロ シージャ	ユーザーの現行セッションについて、現行のリソー ス・コンシューマ・グループの変更を容易にします。

SET_ROLE プロシージャ

このプロシージャは、ロールを使用可能または使用禁止にします。このプロシージャは、SET ROLE SQL 文と同じです。

構文

```
DBMS_SESSION.SET_ROLE (  
    role_cmd VARCHAR2);
```

パラメータ

表 46-2 SET_ROLE プロシージャのパラメータ

パラメータ	説明
role_cmd	このテキストは "set role" に追加され、SQL として実行されます。

SET_SQL_TRACE プロシージャ

このプロシージャは、トレースをオンまたはオフにします。このプロシージャは、次の SQL 文と同じです。

```
ALTER SESSION SET SQL_TRACE ...
```

構文

```
DBMS_SESSION.SET_SQL_TRACE (  
    sql_trace boolean);
```

パラメータ

表 46-3 SET_SQL_TRACE プロシージャのパラメータ

パラメータ	説明
sql_trace	TRUE はトレースをオンにし、FALSE はトレースをオフにします。

SET_NLS プロシージャ

このプロシージャは、各国語サポート（NLS）を設定します。このプロシージャは、次の SQL 文と同じです。

```
ALTER SESSION SET <nls_parameter> = <value>
```

構文

```
DBMS_SESSION.SET_NLS (  
    param VARCHAR2,  
    value VARCHAR2);
```

パラメータ

表 46-4 SET_NLS プロシージャのパラメータ

パラメータ	説明
param	NLS パラメータ。このパラメータ名は、'NLS' で開始する必要があります。
value	パラメータ値。 パラメータがテキスト・リテラルの場合は、埋込みの一重引用符が必要です。たとえば、次のように指定します。"set_nls(nls_date_format,'DD-MON-YY')"

CLOSE_DATABASE_LINK プロシージャ

このプロシージャは、オープンしているデータベース・リンクをクローズします。このプロシージャは、次の SQL 文と同じです。

```
ALTER SESSION CLOSE DATABASE LINK <name>
```

構文

```
DBMS_SESSION.CLOSE_DATABASE_LINK (  
    dblink VARCHAR2);
```

パラメータ

表 46-5 CLOSE_DATABASE_LINK プロシージャのパラメータ

パラメータ	説明
dblink	クローズするデータベース・リンク名

RESET_PACKAGE プロシージャ

このプロシージャは、このセッションのすべてのパッケージのインスタンス化の解除をします。このプロシージャは、すべてのパッケージ状態を解放します。

実行状態をキャッシュするために使用するメモリーは、セッションで実行された PL/SQL ファンクション、プロシージャおよびパッケージに関連付けられています。

パッケージに関して、このメモリーのコレクションはパッケージ変数の現行の値を保持し、各 PL/SQL プログラムがオープンしたカーソルのキャッシュを制御します。RESET_PACKAGE へのコールは、以前実行した各 PL/SQL プログラムに関連付けられていたメモリーをセッションから解放します。したがって、グローバルなパッケージの現行の値は消去され、キャッシュされたカーソルがクローズします。

RESET_PACKAGE は、セッションで失敗したプログラムを確実に再起動するためにも使用できます。パッケージ変数を含んだプログラムが失敗すると、どの変数を初期化し直す必要があるかを判別することは困難です。RESET_PACKAGE は、すべてのパッケージ変数が初期値に再設定されることを保証します。

構文

```
DBMS_SESSION.RESET_PACKAGE;
```

パラメータ

なし

使用上の注意

すべての実行 PL/SQL が消費するメモリーは大量になるため、RESET_PACKAGE を使用して、データベース・アプリケーション内のある時点でセッション・メモリー・フットプリントを削減できます。ただし、パッケージ変数値の再設定がアプリケーションに影響を与えないことを確認してください。また、キャッシュしたメモリーとカーソルのないプログラムを後で実行すると、解放されたメモリーとカーソルを再作成する必要があるため、実行速度が遅くなることに留意してください。

RESET_PACKAGE は、メモリー、カーソルおよびパッケージ変数をコール直後に解放しません。

注意： RESET_PACKAGE は、起動した PL/SQL コールの実行が完了した後でのみ、メモリー、カーソルおよびパッケージ変数を解放します。

たとえば、PL/SQL プロシージャ P1 が PL/SQL プロシージャ P2 をコールし、P2 が RESET_PACKAGE をコールしたとします。プロシージャ P1 の実行が完了するまで（PL/SQL コールが終了するまで）、RESET_PACKAGE の処理は行われません。

例

SQL*Plus スクリプトは、グローバル変数を使用する場合もしない場合もある多数の PL/SQL プログラム・ユニットを伴った大きいプログラムを実行します。ただし、実行後はグローバル変数は必要ありません。

```
EXCECUTE large_plsql_program1;
```

キャッシュされた PL/SQL セッション・メモリーを解放します。

```
EXECUTE DBMS_SESSION.RESET_PACKAGE;
```

別の大きいプログラムを実行します。

```
EXECUTE large_plsql_program2;
```

UNIQUE_SESSION_ID ファンクション

このファンクションは、データベースに現在接続しているすべてのセッションに対して一意の識別子を戻します。同じセッション中にこのファンクションを複数回コールしても、常に同じ結果が戻されます。

構文

```
DBMS_SESSION.UNIQUE_SESSION_ID  
    RETURN VARCHAR2;
```

パラメータ

なし

プラグマ

```
pragma restrict_references(unique_session_id,WNDS,RNDS,WNPS);
```

戻り値

表 46-6 UNIQUE_SESSION_ID ファンクションの戻り値

戻り値	説明
unique_session_id	最大 24 バイトまで戻します。

IS_ROLE_ENABLED ファンクション

このファンクションは、指定のロールがこのセッションで使用可能かどうかを判別します。

構文

```
DBMS_SESSION.IS_ROLE_ENABLED (  
    rolename VARCHAR2)  
RETURN BOOLEAN;
```

パラメータ

表 46-7 IS_ROLE_ENABLED ファンクションのパラメータ

パラメータ	説明
rolename	ロール名

戻り値

表 46-8 IS_ROLE_ENABLED ファンクションの戻り値

戻り値	説明
is_role_enabled	ロールが使用可能かどうかによって TRUE または FALSE

IS_SESSION_ALIVE ファンクション

このファンクションは、指定されたセッションが有効かどうかを判別します。

構文

```
DBMS_SESSION.IS_SESSION_ALIVE (  
    uniqueid VARCHAR2)  
RETURN BOOLEAN;
```

パラメータ

表 46-9 IS_SESSION_ALIVE ファンクションのパラメータ

パラメータ	説明
uniqueid	セッションの一意の ID。これは、UNIQUE_SESSION_ID で戻される ID と同じです。

戻り値

表 46-10 IS_SESSION_ALIVE ファンクションの戻り値

戻り値	説明
is_session_alive	セッションが有効かどうかによって TRUE または FALSE

SET_CLOSE_CACHED_OPEN_CURSORS プロシージャ

このプロシージャは、close_cached_open_cursors をオンまたはオフにします。このプロシージャは、次の SQL 文と同じです。

```
ALTER SESSION SET CLOSE_CACHED_OPEN_CURSORS ...
```

構文

```
DBMS_SESSION.SET_CLOSE_CACHED_OPEN_CURSORS (  
    close_cursors BOOLEAN);
```

パラメータ

表 46-11 SET_CLOSE_CACHED_OPEN_CURSORS プロシージャのパラメータ

パラメータ	説明
close_cursors	TRUE または FALSE

FREE_UNUSED_USER_MEMORY プロシージャ

このプロシージャは、大量のメモリー（100K を超えるメモリー）が必要な操作の実行後、未使用のメモリーを再要求します。

大量のメモリーを使用する操作の例を次に示します。

- sort_area_size 全部を使用し、sort_area_size が数百 KB になる大規模なソート処理
- 大規模な PL/SQL パッケージ、プロシージャまたはファンクションのコンパイル処理
- PL/SQL 索引表内にある数百 KB のデータを格納する処理

V\$SESSTAT または V\$STATNAME の固定ビューにある統計情報 "session uga memory" と "session pga memory" を追跡調査して、ユーザー・メモリーをモニターできます。これらの統計情報のモニターでは、このプロシージャが解放したメモリー量も表示されます。

注意： このプロシージャは、メモリーが非常に不足している場合のみ使用してください。頻繁に使用せず、慎重に使用してください。

構文

```
DBMS_SESSION.FREE_UNUSED_USER_MEMORY;
```

パラメータ

なし

戻り値

このプロシージャの動作は、クライアントのかわりに稼動しているサーバーの構成によって決まります。

- **専用サーバー：**使用されていない PGA メモリーとセッション・メモリーをオペレーティング・システムに戻します。セッション・メモリーは、この構成内の PGA から割り当てられます。
- **MTS サーバー：**使用されていないセッション・メモリーを `shared_pool` に戻します。セッション・メモリーは、この構成内の `shared_pool` から割り当てられます。

使用上の注意

このプロシージャを使用してメモリーを解放するためには、そのメモリーが使用中でないことが必要です。

ある操作でメモリーを割り当てた後は、同じタイプの操作でのみ、割り当てられたメモリーを再使用できます。たとえば、ソート用にメモリーが割り当てられ、ソート完了後にそのメモリーが使用されなくなると、そのソート用に割り当てられたメモリーは別のソートでのみ再使用できます。ソートとコンパイルについては、操作の完了後にメモリーが使用されなくなると、ユーザーはこのプロシージャをコールして、この未使用メモリーを解放できます。

索引表にはメモリーが暗黙的に割り当てられ、その索引表の要素に割り当てられた値が格納されます。したがって、索引表内の要素が多いほど、RDBMS は多くのメモリーを索引表に割り当てます。索引表に要素がある間は、索引表に関連するメモリーは使用中になります。

索引表の有効範囲によって、メモリーの使用期間が決まります。グローバルに宣言された索引表は、パッケージまたはパッケージ本体で宣言された索引表です。これらの索引表には、セッション・メモリーからメモリーが割り当てられます。グローバルに宣言された索引表について、メモリーは、ユーザーのログイン中（ユーザーのセッションの間）は使用中のままとなり、Oracle から切断した後に解放されます。

ローカルで宣言された索引表は、ファンクション、プロシージャまたは無名ブロック内で宣言された索引表です。このような索引表には、PGA メモリーからメモリーが割り当てられます。ローカルに宣言された索引表については、索引表が宣言されたプロシージャ、ファンク

ションまたは無名ブロックをユーザーが実行している限り、メモリーは使用中のままになります。プロシージャ、ファンクションまたは無名ブロックの実行が完了すると、そのメモリーはローカルに宣言された他の索引表で使用可能になります（つまり、メモリーは使用中ではなくなります）。

初期化されていない空の索引表を既存の索引表に割り当てることは、索引表とその索引表に関連付けられたメモリーを明示的に再初期化するための1つの方法です。この操作を行うと、索引表に関連付けられているメモリーは使用中ではなくなり、このプロシージャをコールして解放できます。この方法は、グローバルに宣言した索引表がユーザー・セッションの期間中に大きくなる可能性があり、ユーザーが索引表の内容を必要としない場合は、特に役立ちます。

索引表の有効範囲に関連するメモリー・ルールは適用されたままです。しかし、この方法とプロシージャによって、ユーザーが介入して、索引表に関連付けられているメモリーを明示的に解放できます。

例

次の PL/SQL は、この方法と `FREE_UNUSED_USER_MEMORY` プロシージャの使用方法を示します。

```
CREATE PACKAGE foobar
    type number_idx_tbl is table of number indexed by binary_integer;

    store1_table number_idx_tbl;      -- PL/SQL indexed table
    store2_table number_idx_tbl;      -- PL/SQL indexed table
    store3_table number_idx_tbl;      -- PL/SQL indexed table
    ...
END;                                -- end of foobar

DECLARE
    ...
    empty_table number_idx_tbl;      -- uninitialized ("empty") version
BEGIN
    FOR i in 1..1000000 loop
        store1_table(i) := i;        -- load data
    END LOOP;
    ...
    store1_table := empty_table;      -- "truncate" the indexed table
    ...
    -
    dbms_session.free_unused_user_memory; -- give memory back to system

    store1_table(1) := 100;           -- index tables still declared;
    store2_table(2) := 200;           -- but truncated.
    ...
END;
```

SET_CONTEXT プロシージャ

このプロシージャは、コンテキスト属性の値を設定または再設定します。

構文

```
DBMS_SESSION.SET_CONTEXT (  
    namespace VARCHAR2,  
    attribute  VARCHAR2,  
    value      VARCHAR2);
```

パラメータ

表 46-12 SET_CONTEXT プロシージャのパラメータ

パラメータ	説明
namespace	アプリケーションのコンテキストで使用する名前領域の名前（最大 30 バイト）
attribute	設定する属性の名前（最大 30 バイト）
value	設定する値（最大 256 バイト）

使用上の注意

このファンクションのコール側は、CREATE CONTEXT 文を介してコンテキスト名前領域に関連付けられたプロシージャのコール・スタックに存在している必要があります。コール・スタックのチェックは、データベース管理システムの境界を越えません。

名前領域に設定できる属性の数に制限はありません。属性値は、ユーザーが再設定するまでユーザー・セッションの間そのまま残ります。

LIST_CONTEXT プロシージャ

このプロシージャは、現行セッションに関するアクティブな名前領域とコンテキストのリストを戻します。

構文

```
TYPE AppCtxRecTyp IS RECORD (  
    namespace VARCHAR2(30),  
    attribute  VARCHAR2(30),  
    value      VARCHAR2(256));  
  
TYPE AppCtxTabTyp IS TABLE OF AppCtxRecTyp INDEX BY BINARY_INTEGER;  
  
DBMS_SESSION.LIST_CONTEXT (  

```

```
list OUT AppCtxTabTyp,  
size OUT NUMBER);
```

パラメータ

表 46-13 LIST_CONTEXT プロシージャのパラメータ

パラメータ	説明
list	現行セッション内のアプリケーション・コンテキスト設定リストを格納するバッファ

戻り値

表 46-14 LIST_CONTEXT プロシージャの戻り値

戻り値	説明
list	現行セッションにある設定（名前領域、属性、値）リスト。
size	戻されたバッファ内のエントリ数を戻します。

使用上の注意

リスト内のコンテキスト情報は、<namespace><attribute><value> の順に表示されます。list は表タイプの変数なので、そのサイズは戻されるリストのサイズに合わせて動的に調整されます。

SWITCH_CURRENT_CONSUMER_GROUP プロシージャ

このプロシージャは、ユーザーの現行セッションでの現行のリソース・コンシューマ・グループを変更します。

ある特定のグループに対してスイッチ権限がある場合は、コンシューマ・グループを切り替えることができます。コール側が別のプロシージャの場合、ユーザーは、そのプロシージャの所有者がスイッチ権限を持っているコンシューマ・グループに切り替えることができます。

構文

```
DBMS_SESSION.switch_current_consumer_group (  
  new_consumer_group      IN  VARCHAR2,  
  old_consumer_group      OUT VARCHAR2,  
  initial_group_on_error  IN   BOOLEAN);
```


パラメータ

表 46-15 SWITCH_CURRENT_CONSUMER_GROUP プロシージャのパラメータ

パラメータ	説明
new_consumer_group	切り替える先のコンシューマ・グループの名前。
old_consumer_group	切り替える元のコンシューマ・グループの名前。
initial_group_on_error	TRUE の場合は、エラー発生時にコール側の現行コンシューマ・グループが初期コンシューマ・グループとして設定されます。

戻り値

このプロシージャは、old_consumer_group パラメータにある、ユーザーの古いコンシューマ・グループを出力します。

注意： old_consumer_group で戻された値を使用して、古いコンシューマ・グループに後で切り替えることができます。

例外

表 46-16 SWITCH_CURRENT_CONSUMER_GROUP プロシージャの例外

例外	説明
29368	コンシューマ・グループが存在しません。
1031	権限が不十分です。
29396	OTHER_GROUPS コンシューマ・グループに切り替えることができません。

使用上の注意

プロシージャの所有者には、ユーザーを古いコンシューマ・グループに再切替えるために、ユーザーの古いグループ (old_consumer_group) に対する権限が必要です。例外が 1 つあります。このプロシージャでは、ユーザーを初期のコンシューマ・グループにいつでも切り替えることができます (権限チェックはスキップ)。

initial_group_on_error を TRUE に設定すると、SWITCH_CURRENT_CONSUMER_GROUP プロシージャは、new_consumer_group が指定したグループに現行セッションを設定できない場合、そのセッションをデフォルト・グループに設定します。現行のコンシューマ・グループが初期のコンシューマ・グループに変更されていても、new_consumer_group へのセッションの移動に関連するエラーは発生します。

例

```
CREATE OR REPLACE PROCEDURE high_priority_task is
    old_group varchar2(30);
    prev_group varchar2(30);
    curr_user varchar2(30);
BEGIN
    -- switch invoker to privileged consumer group. If we fail to do so, an
    -- error will be thrown, but the consumer group will not change
    -- because 'initial_group_on_error' is set to FALSE

    dbms_session.switch_current_consumer_group('tkrogrp1', old_group, FALSE);
    -- set up exception handler (in the event of an error, we do not want to
    -- return to caller while leaving the session still in the privileged
    -- group)

    BEGIN
        -- perform some operations while under privileged group

    EXCEPTION
        WHEN OTHERS THEN
            -- It is possible that the procedure owner does not have privileges
            -- on old_group. 'initial_group_on_error' is set to TRUE to make sure
            -- that the user is moved out of the privileged group in such a
            -- situation

            dbms_session.switch_current_consumer_group(old_group,prev_group,TRUE);
            RAISE;
        END;

    -- we've succeeded. Now switch to old_group, or if cannot do so, switch
    -- to caller's initial consumer group

    dbms_session.switch_current_consumer_group(old_group,prev_group,TRUE);
END high_priority_task;
/
```

DBMS_SHARED_POOL

DBMS_SHARED_POOL は、共有プールへのアクセスを提供します。この共有プールは、カーソルと PL/SQL オブジェクトが格納されている共有メモリー領域です。DBMS_SHARED_POOL によって、共有プール内のオブジェクト・サイズを表示したり、メモリーの断片化を減らすためにオブジェクトを保存または非保存としてマークすることができます。

インストール時の注意

DBMS_SHARED_POOL を作成するには、DBMSPOOL.SQL スクリプトを実行します。
PRVTPOOL.PLB スクリプトは、DBMSPOOL.SQL の実行後に自動的に実行されます。これらの
スクリプトは、CATPROC.SQL では実行されません。

使用上の注意

ここで提供されるプロシージャは、大規模な PL/SQL オブジェクトのロード時に役立ちます。
大規模な PL/SQL オブジェクトのロード時には、大量の小規模オブジェクトを共有プールから
期限切れとして削除して空き領域を用意する必要があるため（メモリー断片化のため）、
ユーザーの応答時間に影響を与えます。場合によっては、大規模なオブジェクトをロードす
るためのメモリーが不足している場合があります。

DBMS_SHARED_POOL は、頻繁に実行するトリガーに対しても有効です。コンパイルしたトリ
ガーを共有プールで頻繁に使用する表に保管できます。さらに、DBMS_SHARED_POOL は
順序もサポートします。順序番号は、順序が期限切れで共有プールから削除されると失われ
ます。DBMS_SHARED_POOL は、共有プールに順序を保存し、順序番号の損失を防止するの
に役立ちます。

サブプログラムの要約

表 47-1 DBMS_SHARED_POOL パッケージのサブプログラム

サブプログラム	説明
47-2 ページの SIZES プロシージャ	共有プールにあるオブジェクトで、指定サイズより大きいオブ ジェクトを表示します。
47-3 ページの KEEP プロシージャ	共有プールにオブジェクトを保存します。
47-5 ページの UNKEEP プロシージャ	指定オブジェクトを解放します。
47-5 ページの ABORTED_REQUEST_THRESHOLD プロシージャ	共有プールについて、異常終了を要求するしきい値を設定しま す。

SIZES プロシージャ

このプロシージャは、shared_pool にあるオブジェクトが指定したサイズより大きいオブ
ジェクトを表示します。オブジェクト名を指定すると、後の KEEP または UNKEEP いずれか
のコールへの引数として使用できます。

構文

```
DBMS_SHARED_POOL.SIZES (  
    minsize NUMBER);
```

パラメータ

表 47-2 SIZES プロシージャのパラメータ

パラメータ	説明
minsize	オブジェクトを表示するために、共有プール内で占有する必要があるサイズ（単位は KB）

使用上の注意

このプロシージャの使用前に、SQL*DBA または SQL*Plus 'SET SERVEROUTPUT ON SIZE XXXXX' コマンドを発行すると、結果が表示されます。

KEEP プロシージャ

このプロシージャは、共有プールにオブジェクトを保存します。オブジェクトが一度共有プールに保存されると、期間切れ削除の対象となりません。このプロシージャは、比較的頻繁に使用する大規模なオブジェクトに対して有効です。これは、大規模なオブジェクトが共有プールに移動されるときは、連続した十分な大きさの領域を作成するために、（移動されるオブジェクトのサイズよりはるかに大きい）他の大量のオブジェクトを期間切れで削除する必要があるためです。

構文

```
DBMS_SHARED_POOL.KEEP (  
    name VARCHAR2,  
    flag CHAR      DEFAULT 'P');
```

注意： このプロシージャは、自動メカニズムが将来インプリメントされて不要になった場合は、サポートされなくなる可能性があります。

パラメータ

表 47-3 KEEP プロシージャのパラメータ

パラメータ	説明
name	保存するオブジェクトの名前。 この識別子は、アドレスと、v\$sqlarea ビューの hash_value 列を連結した値です。これは、SIZES プロシージャで表示されます。 現在、TABLE と VIEW オブジェクトは保存できません。
flag	(オプション) このパラメータを指定しないと、パッケージは、最初のパラメータをパッケージ、プロシージャまたはファンクションの名前とみなして名前を解決します。 入力内容がパッケージ、プロシージャまたはファンクションの名前であることを完全に指定するには、'P' または 'p' を設定します。 入力内容がタイプ名であることを指定するには、'T' または 't' を設定します。 入力内容がトリガー名であることを指定するには、'R' または 'r' を設定します。 入力内容が順序名であることを指定するには、'Q' または 'q' を設定します。 最初の引数がカーソル・アドレスとハッシュ値の場合、パラメータには、'P' か 'p'、'Q' か 'q'、'R' か 'r'、'T' か 't' を除く任意の文字を指定する必要があります。

例外

指定されたオブジェクトが見つからない場合は、例外が発生します。

使用上の注意

オブジェクトは 2 種類あります。

- 名前で指定する PL/SQL オブジェクト、トリガー、順序およびタイプ
- 2 つの番号（共有プール内の位置を示す）で指定する SQL カーソル・オブジェクト

例：

```
DBMS_SHARED_POOL.KEEP('scott.hispackage')
```

この例では、SCOTT が所有するパッケージ HISPACAGE を保存します。PL/SQL オブジェクトの名前は、オブジェクト命名の SQL ルールに従っています（つまり、区切られた名前やマルチバイトの名前などが可能です）。カーソルは、DBMS_SHARED_POOL.KEEP('0034CDFF,

20348871') によって保存できます。最初の 8 文字は 16 進数の完全なアドレスである必要があります。

UNKEEP プロシージャ

このプロシージャは、指定のオブジェクトを解放します。

構文

```
DBMS_SHARED_POOL.UNKEEP (  
    name VARCHAR2,  
    flag CHAR      DEFAULT 'P');
```

注意： このプロシージャは、自動メカニズムが将来インプリメントされて不要になった場合は、サポートされなくなる可能性があります。

パラメータ

表 47-4 UNKEEP プロシージャのパラメータ

パラメータ	説明
name	保存しないオブジェクトの名前。「KEEP」プロシージャの name パラメータの説明を参照してください。
flag	KEEP プロシージャの flag パラメータの説明を参照してください。

例外

指定されたオブジェクトが見つからない場合は、例外が発生します。

ABORTED_REQUEST_THRESHOLD プロシージャ

このプロシージャは、共有プールについて、異常終了を要求するしきい値を設定します。

構文

```
DBMS_SHARED_POOL.ABORTED_REQUEST_THRESHOLD (  
    threshold_size NUMBER);
```

パラメータ

表 47-5 ABORTED_REQUEST_THRESHOLD プロシージャのパラメータ

パラメータ	説明
threshold_size	共有プール内で確保解除されたメモリー（解放されたメモリーではない）を解放しないための要求サイズ（単位はバイト）。threshold_size の範囲は 5000 ～ 2GB までです。

例外

しきい値が有効な範囲内にない場合は、例外が発生します。

使用上の注意

通常、要求が空きリストで満たされない場合、RDBMS は、LRU リストからオブジェクトを解放して要求を満たすことができるかを定期的にチェックし、メモリーを再要求しようとします。このステップの完了後、RDBMS は、'ALTER SYSTEM FLUSH SHARED_POOL' とほぼ同等の内容を実行します。

これは、システム上のすべてのユーザーに影響を与えるため、このプロシージャは、その影響を thresh_hold サイズを超える共有メモリーの断片検索に失敗した処理にローカライズします。このユーザーは、LRU リストを検索しなくても、'メモリー不足'エラーとなります。

DBMS_SNAPSHOT

DBMS_SNAPSHOT によって、同じリフレッシュ・グループおよびバージ・ログの一部ではないスナップショットをリフレッシュできます。表 48-1 に、このパッケージの主なサブプログラムを示します。

注意： DBMS_MVIEW は、DBMS_SNAPSHOT のシノニムです。

DBMS_SNAPSHOT パッケージ

サブプログラムの要約

表 48-1 DBMS_SNAPSHOT パッケージのサブプログラム

サブプログラム	説明
48-3 ページの BEGIN_TABLE_REORGANIZATION プロ シージャ	リフレッシュに必要なスナップショット・データを保存 するプロセスを実行します。
48-3 ページの END_TABLE_REORGANIZATION プロシー ジャ	マスター表のスナップショット・データが有効であり、 マスター表が適切な状態であることを確認します。
48-4 ページの I_AM_A_REFRESH ファンクション	I_AM_REFRESH パッケージの状態の値を戻します。
48-4 ページの PURGE_DIRECT_LOAD_LOG プロシージャ	スナップショットで行が不要になると、ダイレクト・ ローダー・ログからその行を削除します（データ・ウェ アハウスで使用）。
48-5 ページの PURGE_LOG プロシージャ	スナップショット・ログから行をパージします。
48-6 ページの PURGE_SNAPSHOT_FROM_LOG プロシー ジャ	スナップショット・ログから行をパージします。
48-8 ページの REFRESH プロシージャ	同じリフレッシュ・グループのメンバーではない 1 つ以 上のスナップショットを継続してリフレッシュします。
48-11 ページの REFRESH_ALL_MVIEWS プロシージャ	変更がマスター表に反映されていないスナップショット をすべてリフレッシュします。
48-12 ページの REFRESH_DEPENDENT プロシージャ	指定のマスター表またはマスター表のリストに依存して いる表ベースのスナップショットをすべてリフレッシュ します。
48-14 ページの REGISTER_SNAPSHOT プロシージャ	個々のスナップショットの管理を可能にします。
48-16 ページの UNREGISTER_SNAPSHOT プロシージャ	個々のスナップショットの管理を可能にします。マス ター・サイトで起動して、スナップショットを登録解除 します。

BEGIN_TABLE_REORGANIZATION プロシージャ

このプロシージャは、リフレッシュに必要なスナップショット・データを保存するプロセスを実行します。このプロシージャは、マスター表の再編成前にコールする必要があります。

関連項目： スナップショット・ログがあるマスター表の再編成に関する情報は、『Oracle8i レプリケーション・マネージメント API リファレンス』を参照してください。

構文

```
DBMS_SNAPSHOT.BEGIN_TABLE_REORGANIZATION (  
    tabowner      IN    VARCHAR2,  
    tabname       IN    VARCHAR2);
```

パラメータ

表 48-2 BEGIN_TABLE_REORGANIZATION プロシージャのパラメータ

パラメータ	説明
tabowner	再編成する表の所有者
tabname	再編成する表の名前

END_TABLE_REORGANIZATION プロシージャ

このプロシージャは、マスター表の再編成後にコールする必要があります。マスター表のスナップショット・データが有効であり、マスター表が適切な状態であることを確認します。

関連項目： スナップショット・ログがあるマスター表の再編成に関する情報は、『Oracle8i レプリケーション・マネージメント API リファレンス』を参照してください。

構文

```
DBMS_SNAPSHOT.END_TABLE_REORGANIZATION (  
    tabowner      IN    VARCHAR2,  
    tabname       IN    VARCHAR2);
```

パラメータ

表 48-3 END_TABLE_REORGANIZATION プロシージャのパラメータ

パラメータ	説明
tabowner	再編成する表の所有者
tabname	再編成する表の名前

I_AM_A_REFRESH ファンクション

このファンクションは、I_AM_REFRESH パッケージの状態の値を戻します。戻り値が TRUE の場合は、各レプリケーション・トリガーが最初にこの状態をチェックするため、このセッションではスナップショットのすべてのローカル・レプリケーション・トリガーが完全に使用禁止になります。戻り値が FALSE の場合は、そのトリガーを使用できます。

構文

```
DBMS_SNAPSHOT.I_AM_A_REFRESH()  
RETURN BOOLEAN;
```

パラメータ

なし

PURGE_DIRECT_LOAD_LOG プロシージャ

このプロシージャは、既知のスナップショット（マテリアライズド・ビュー）でエントリが不要になると、ダイレクト・ローダー・ログからそのエントリを削除します。このプロシージャは通常、Oracle のデータ・ウェアハウス・テクノロジーを使用する環境で使用されます。

関連項目： 詳細は、『Oracle8i データ・ウェアハウス』を参照してください。

構文

```
DBMS_SNAPSHOT.PURGE_DIRECT_LOAD_LOG();
```

PURGE_LOG プロシージャ

このプロシージャは、スナップショット・ログから行を削除します。

構文

```
DBMS_SNAPSHOT.PURGE_LOG (  
    master          IN   VARCHAR2,  
    num             IN   BINARY_INTEGER := 1,  
    flag            IN   VARCHAR2      := 'NOP');
```

パラメータ

表 48-4 PURGE_LOG プロシージャのパラメータ

パラメータ	説明
master	マスター表の名前。
num	<p>スナップショット・ログから行を削除するスナップショットの中で、リフレッシュ日付が最も古いスナップショットの数。たとえば、次の文は、リフレッシュ日付が最も古い 2 つのスナップショットをリフレッシュするために必要な行を削除します。</p> <pre>dbms_snapshot.purge_log('master_table', 2);</pre> <p>スナップショット・ログにあるすべての行を削除するには、次の例のように、削除するスナップショットについて大きい数を指定します。</p> <pre>dbms_snapshot.purge_log('master_table', 9999);</pre> <p>この文は、MASTER_TABLE に基づくスナップショット数が 9999 未満の場合、MASTER_TABLE に対応しているスナップショット・ログを完全に削除します。行がスナップショット・ログからすでに削除されている単純なスナップショットは、次回リフレッシュ時に完全にリフレッシュする必要があります。</p>
flag	<p>スナップショット・ログから最低 1 つのスナップショットの行が削除されることを保証する場合は、DELETE を指定します。この引数は、引数 num の設定を上書きできます。たとえば、次の文は、スナップショット・ログに従属行が実際にあり、リフレッシュ日付が最も古いスナップショットから行を削除します。</p> <pre>dbms_snapshot.purge_log('master_table', 1, 'DELETE');</pre>

PURGE_SNAPSHOT_FROM_LOG プロシージャ

このプロシージャは、スナップショット・リフレッシュに関連したデータ・ディクショナリ表にある行を削除するためにマスター・サイトでコールされます。この表は `snapshot_id` または `snapowner`、`snapname` および `snapsite` の組合せで識別される指定スナップショットについて、マスター・サイトでメンテナンスされている表です。指定したスナップショットが、任意のマスター表からリフレッシュされた最も古いスナップショットの場合は、そのスナップショット・ログも削除されます。このプロシージャは、スナップショットの登録解除は行いません。

スナップショット・ログの 1 つを削除している間にエラーが発生した場合、それ以前に正常終了したスナップショット・ログの削除処理はロールバックされません。これは、スナップショット・ログのサイズを最小限にするためです。このプロシージャは、エラーが発生した場合でも、すべてのスナップショット・ログが削除されるまで再起動できます。

構文

```
DBMS_SNAPSHOT.PURGE_SNAPSHOT_FROM_LOG (  
    snapshot_id    IN    BINARY_INTEGER |  
    snapowner      IN    VARCHAR2,  
    snapname       IN    VARCHAR2,  
    snapsite       IN    VARCHAR2);
```

注意： このプロシージャはオーバーロードされています。`snapshot_id` には、同時に指定することのできないパラメータが 3 つあります。`snapowner`、`snapname` および `snapsite` です。

パラメータ

表 48-5 PURGE_SNAPSHOT_FROM_LOG プロシージャのパラメータ

パラメータ	説明
snapshot_id	<p>このプロシージャをターゲット・スナップショットの ID に基づいて実行する場合は、snapshot_id パラメータでスナップショット ID を指定します。スナップショット ID のリストに関しては、スナップショット・サイトで DBA_SNAPSHOT_LOGS ビューを問い合わせます。</p> <p>登録済スナップショットのリスト (DBA_REGISTERED_SNAPSHOTS) にターゲット・スナップショットがない場合は、スナップショット ID に基づいてこのプロシージャを実行すると便利です。</p>
snapowner	snapshot_id を指定しない場合は、snapowner パラメータでターゲット・スナップショットの所有者を入力します。スナップショット・ログ・サイトで DBA_REGISTERED_SNAPSHOTS ビューを問い合わせ、スナップショットの所有者を表示します。
snapname	snapshot_id を指定しない場合は、snapname パラメータでターゲット・スナップショット名を入力します。スナップショット・ログ・サイトで DBA_REGISTERED_SNAPSHOTS ビューを問い合わせ、スナップショット名を表示します。
snapsite	snapshot_id を指定しない場合は、snapsite パラメータでターゲット・スナップショットのサイトを入力します。スナップショット・ログ・サイトで DBA_REGISTERED_SNAPSHOTS ビューを問い合わせ、スナップショットのサイトを表示します。

REFRESH プロシージャ

このプロシージャは、スナップショットのリストをリフレッシュします。

構文

```
DBMS_SNAPSHOT.REFRESH (
  { list          IN          VARCHAR2,
    | tab          IN OUT DBMS_UTILITY.UNCL_ARRAY, }
  method          IN          VARCHAR2      := NULL,
  rollback_seg     IN          VARCHAR2      := NULL,
  push_deferred_rpc IN          BOOLEAN      := TRUE,
  refresh_after_errors IN        BOOLEAN      := FALSE,
  purge_option      IN          BINARY_INTEGER := 1,
  parallelism       IN          BINARY_INTEGER := 0,
  heap_size         IN          BINARY_INTEGER := 0
  atomic_refresh    IN          BOOLEAN      := TRUE);
```

注意： このプロシージャはオーバーロードされています。list パラメータと tab パラメータは、両方同時には指定できません。

パラメータ

表 48-6 REFRESH プロシージャのパラメータ

パラメータ	説明
list tab	<p>リフレッシュするスナップショットのカンマで区切られたリスト。(シノニムはサポートされていません)。これらのスナップショットは、異なるスキーマに格納したり、異なるマスター表を保有できます。ただし、リストされているすべてのスナップショットは、ローカル・データベースに存在している必要があります。</p> <p>他の方法として、DBMS_UTILITY.UNCL_ARRAY 型の PL/SQL 表を渡すことができます。この場合、各要素がスナップショットの名前です。</p>
method	<p>リストされているスナップショットのリフレッシュ方法を示す文字列。F または f は高速リフレッシュ、? は強制リフレッシュ、C または c は完全リフレッシュ、A または a は常によりリフレッシュをそれぞれ示します。スナップショットに対応するリフレッシュ方法がない場合 (つまり、リフレッシュ方法より多くのスナップショットが指定されている場合)、そのスナップショットはデフォルトのリフレッシュ方法に従ってリフレッシュされます。たとえば、SQL*Plus 内の次の EXECUTE 文を実行します。</p> <pre> dbms_snapshot.refresh ('s_emp,s_dept,scott.s_salary','CF'); </pre> <p>この文は、S_EMP スナップショットの完全リフレッシュ、S_DEPT スナップショットの高速リフレッシュ、および SCOTT.S_SALARY スナップショットのデフォルト・リフレッシュを実行します。</p>
rollback_seg	<p>スナップショットのリフレッシュ中に使用する、スナップショット・サイトのロールバック・セグメント名。</p>
push_deferred_rpc	<p>更新可能なスナップショットでのみ使用します。スナップショットをリフレッシュする前に、スナップショットから関連マスターに変更を送信する場合は、このパラメータを TRUE に設定します。そうでない場合は、変更が一時的に失われたように表示される場合があります。</p>
refresh_after_errors	<p>このパラメータを TRUE に設定すると、スナップショットのマスター表の DEFERROR ビューに未解決の競合が記録されていても、更新可能なスナップショットのリフレッシュは続行します。このパラメータが TRUE で、atomic_refresh が FALSE の場合、このプロシージャは、スナップショットのリフレッシュに失敗しても他のスナップショットのリフレッシュを続行します。</p>

表 48-6 REFRESH プロシージャのパラメータ

パラメータ	説明
purge_option	パラレル伝播メカニズムを使用する場合（つまり、パラレル化に 1 以上を設定）は、次のように指定します。0 = パージなし、1 = レイジー・パージ、2 = aggressive パージ。ほとんどの場合、レイジー・パージが最適な設定です。複数のマスター・レプリケーション・グループが別々のターゲット・サイトに送信され、1 つ以上のレプリケーション・グループへの更新や送信がまれな場合は、aggressive パージに設定してキューを減らします。すべてのレプリケーション・グループへの更新と送信がまれな場合は、このパラメータを 0 に設定し、キューを減らすためにこのパラメータを時々 2 に設定して PUSH を実行してください。
parallelism	0 = シリアル伝播、 $n > 1 = n$ 個のパラレル・サーバー・プロセスを使用したパラレル伝播、1 = 1 つのパラレル・サーバー・プロセスのみを使用したパラレル伝播。
heap_size	パラレル伝播スケジューリングで同時に検査されるトランザクションの最大数。最適なパフォーマンスのためのデフォルト設定は Oracle が自動的に計算します。オラクル社カスタマ・サポート・センターから指示がない限り、このパラメータは設定しないでください。
atomic_refresh	<p>このパラメータを TRUE に設定すると、スナップショットのリストは単一のトランザクションでリフレッシュされます。リフレッシュされたすべてのスナップショットは、単一のポイントへ同時に更新されます。スナップショットのいずれかでリフレッシュに失敗すると、すべてのスナップショットが更新されません。</p> <p>このパラメータを FALSE に設定すると、各スナップショットは個別のトランザクションでリフレッシュされます。このパラメータが FALSE の場合、ジョブ・キュー・プロセスの数は 1 以上に設定する必要があります。</p> <p>FALSE で、サマリー管理オプションが導入されていない場合、エラーが発生します。</p>

REFRESH_ALL_MVIEWS プロシージャ

このプロシージャは、次のプロパティを持つすべてのスナップショット（マテリアライズド・ビュー）をリフレッシュします。

- スナップショットが、依存しているマスター表への最新の変更以降リフレッシュされていない場合
- スナップショットとそれが依存しているすべてのマスター表がローカルな場合
- スナップショットが DBA_MVIEWS ビューにある場合

これは、データ・ウェアハウスで使用するためのプロシージャです。

構文

```
DBMS_SNAPSHOT.REFRESH_ALL_MVIEWS (  
    number_of_failures    OUT    BINARY_INTEGER,  
    method                IN     VARCHAR2          := NULL,  
    rollback_seg          IN     VARCHAR2          := NULL,  
    refresh_after_errors  IN     BOOLEAN           := FALSE,  
    atomic_refresh        IN     BOOLEAN           := TRUE);
```

パラメータ

表 48-7 REFRESH_ALL_MVIEWS プロシージャのパラメータ

パラメータ	説明
number_of_failures	処理中に発生した失敗の件数を戻します。
method	各スナップショットに対して実行するリフレッシュのタイプを示す単一のリフレッシュ方法。F または f は高速リフレッシュ、? は強制リフレッシュ、C または c は完全リフレッシュ、A または a は常にリフレッシュを示します。方法が指定されていない場合、スナップショットはデフォルトのリフレッシュ方法に従ってリフレッシュされます。
rollback_seg	スナップショットのリフレッシュ中に使用する、スナップショット・サイトのロールバック・セグメント名。
refresh_after_errors	このパラメータを TRUE に設定すると、スナップショットのマスター表の DEFERROR ビューに未解決の競合が記録されていても、更新可能なスナップショットのリフレッシュは続行されます。このパラメータが TRUE で、atomic_refresh が FALSE の場合、このプロシージャは、スナップショットのリフレッシュに失敗しても、他のスナップショットのリフレッシュを続行します。

表 48-7 REFRESH_ALL_MVIEWS プロシージャのパラメータ

パラメータ	説明
atomic_refresh	<p>このパラメータを TRUE に設定すると、リフレッシュされたスナップショットが単一のトランザクションでリフレッシュされます。リフレッシュされたすべてのスナップショットは、単一のポイントへ同時に更新されます。スナップショットのいずれかでリフレッシュに失敗すると、すべてのスナップショットが更新されません。</p> <p>このパラメータを FALSE に設定すると、リフレッシュされた各スナップショットは個別のトランザクションでリフレッシュされます。このパラメータが FALSE の場合、ジョブ・キュー・プロセスの数は 1 以上に設定する必要があります。</p>

REFRESH_DEPENDENT プロシージャ

このプロシージャは、次のプロパティを持つすべてのスナップショット（マテリアライズド・ビュー）をリフレッシュします。

- スナップショットが指定のマスター表のリストにあるマスター表に依存している場合
- スナップショットが、依存しているマスター表への最新の変更以降リフレッシュされていない場合
- スナップショットとそれが依存しているすべてのマスター表がローカルな場合
- スナップショットが DBA_MVIEWS ビューにある場合

これは、データ・ウェアハウスで使用するためのプロシージャです。

構文

```
DBMS_SNAPSHOT.REFRESH_DEPENDENT (  
  number_of_failures      OUT    BINARY_INTEGER,  
  { list                  IN     VARCHAR2,  
    | tab                 IN OUT DBMS_UTILITY.UNCL_ARRAY, }  
  method                  IN     VARCHAR2      := NULL,  
  rollback_seg            IN     VARCHAR2      := NULL,  
  refresh_after_errors    IN     BOOLEAN       := FALSE,  
  atomic_refresh          IN     BOOLEAN       := TRUE);
```

注意： このプロシージャはオーバーロードされています。list パラメータと tab パラメータは、両方同時には指定できません。

パラメータ

表 48-8 REFRESH_DEPENDENT プロシージャのパラメータ

パラメータ	説明
number_of_failures	処理中に発生した失敗の件数を戻します。
list tab	<p>スナップショットが依存できるマスター表のカンマで区切られたリスト。(シノニムはサポートされていません)。これらの表とそれらに依存するスナップショットは、別々のスキーマに配置できます。ただし、すべての表とスナップショットは、ユーザーのローカル・データベースに存在している必要があります。</p> <p>他の方法として、DBMS_UTILITY.UNCL_ARRAY 型の PL/SQL 表を渡すことができます。この場合、各要素がスナップショットの名前です。</p>
method	<p>リフレッシュ方法の文字列で、依存しているスナップショットのリフレッシュ方法を示します。特定の表に依存しているすべてのスナップショットは、その表に関連付けられたリフレッシュ方法に従ってリフレッシュされます。F または f は高速リフレッシュ、? は強制リフレッシュ、C または c は完全リフレッシュ、A または a は常にリフレッシュを示します。表に対応するリフレッシュ方法がない場合（つまり、リフレッシュ方法より多くの表が指定された場合）、その表に依存するスナップショットはデフォルトのリフレッシュ方法に従ってリフレッシュされます。たとえば、SQL*Plus 内の次の EXECUTE 文を実行します。</p> <pre> dbms_snapshot.refresh_dependent ('emp,dept,scott.salary','CF'); </pre> <p>この文は、EMP 表に依存するスナップショットの完全リフレッシュ、DEPT 表に依存するスナップショットの高速リフレッシュ、および SCOTT.SALARY 表に依存するスナップショットのデフォルト・リフレッシュを実行します。</p>
rollback_seg	スナップショットのリフレッシュ中に使用する、スナップショット・サイトのロールバック・セグメント名。
refresh_after_errors	このパラメータを TRUE に設定すると、スナップショットのマスター表の DEFERROR ビューに未解決の競合が記録されていても、更新可能なスナップショットのリフレッシュは続行されます。このパラメータが TRUE で、atomic_refresh が FALSE の場合、このプロシージャは、スナップショットのリフレッシュに失敗しても他のスナップショットのリフレッシュを続行します。

表 48-8 REFRESH_DEPENDENT プロシージャのパラメータ

パラメータ	説明
atomic_refresh	<p>このパラメータを TRUE に設定すると、リフレッシュされたスナップショットが単一のトランザクションでリフレッシュされます。リフレッシュされたすべてのスナップショットは、単一のポイントへ同時に更新されます。スナップショットのいずれかでリフレッシュに失敗すると、すべてのスナップショットが更新されません。</p> <p>このパラメータを FALSE に設定すると、リフレッシュされた各スナップショットは個別のトランザクションでリフレッシュされます。このパラメータが FALSE の場合、ジョブ・キュー・プロセスの数は 1 以上に設定する必要があります。</p> <p>FALSE で、サマリー管理オプションが導入されていない場合、エラーが発生します。</p>

REGISTER_SNAPSHOT プロシージャ

このプロシージャは、個々のスナップショットの管理を可能にします。

構文

```
DBMS_SNAPSHOT.REGISTER_SNAPSHOT (  
    snapowner    IN    VARCHAR2,  
    snapname     IN    VARCHAR2,  
    snapsite     IN    VARCHAR2,  
    {snapshot_id IN    DATE | BINARY_INTEGER,  
    flag         IN    BINARY_INTEGER,}  
    qry_txt      IN    VARCHAR2,  
    rep_type     IN    BINARY_INTEGER := DBMS_SNAPSHOT.REG_UNKNOWN);
```

注意： このプロシージャはオーバーロードされています。snapshot_id パラメータと flag パラメータは、両方同時には指定できません。

パラメータ

表 48-9 REGISTER_SNAPSHOT プロシージャのパラメータ

パラメータ	説明
snapowner	スナップショットの所有者。
snapname	スナップショット名。
snapsite	Oracle8 以降のマスター・サイトで登録したスナップショットのスナップショット・サイト名。この名前に二重引用符を含めることはできません。
snapshot_id	スナップショットの識別番号。バージョン 8 のスナップショットは BINARY_INTEGER として指定します。バージョン 8 以降のマスター・サイトで登録したバージョン 7 のスナップショットは、DATE として指定します。
flag	後続の CREATE または MOVE 文が問合せテキストに登録されているかどうかを示す PL/SQL パッケージ変数。
query_txt	スナップショット定義問合せの最初の 32,000 バイト。
rep_type	スナップショットのバージョン。代入できる有効な定数は、reg_unknown (デフォルト)、reg_v7_group、reg_v8_group および reg_repapi_group です。

使用上の注意

このプロシージャは、リモート・スナップショット・サイトでリモート・プロシージャ・コールを使用することにより、マスター・サイトで起動されます。REGISTER_SNAPSHOT が同一の snapowner、snapname および snapsite で繰り返しコールされる場合は、snapshot_id、flag および query_txt の最新の値が格納されます。問合せが VARCHAR2 の最大サイズを超える場合は、最初の 32000 文字が query_txt に格納され、残りは切り捨てられます。手動で起動するときは、プロシージャをコールするユーザーが、snapshot_id と flag の値をスナップショット・ビューで調べる必要があります。

スナップショットの問合せをマスター・サイトで登録しない場合は、オプションを FALSE に設定して SET_REGISTER_QUERY_TEXT プロシージャをコールします。最新のオプション設定を調べるためには、DDL の発行前に、スナップショット・サイトで GET_REG_QUERY_TEXT_FLAG ファンクションをコールします。

UNREGISTER_SNAPSHOT プロシージャ

このプロシージャは、個々のスナップショットの管理を可能にします。マスター・サイトで起動して、スナップショットを登録解除します。

構文

```
DBMS_SNAPSHOT.UNREGISTER_SNAPSHOT (  
    snapowner      IN   VARCHAR2,  
    snapname       IN   VARCHAR2,  
    snapsite       IN   VARCHAR2);
```

パラメータ

表 48-10 UNREGISTER_SNAPSHOT プロシージャのパラメータ

パラメータ	説明
snapowner	スナップショットの所有者
snapname	スナップショット名
snapsite	スナップショット・サイト名

DBMS_SPACE

DBMS_SPACE パッケージによって、セグメントの成長と領域要件を分析できます。

セキュリティ

このパッケージの実行には、SYS 権限が必要です。

要件

実行権限は、PUBLIC に付与されます。このパッケージのサブプログラムは、コール側のセキュリティ下で実行されます。ユーザーには、オブジェクトに関する ANALYZE 権限が必要です。

サブプログラムの要約

表 49-1 DBMS_SPACE パッケージのサブプログラム

サブプログラム	説明
49-2 ページの UNUSED_SPACE プロシージャ	オブジェクト（表、索引またはクラスタ）にある未使用領域に関する情報を戻します。
49-4 ページの FREE_BLOCKS プロシージャ	オブジェクト（表、索引またはクラスタ）にある空きブロックに関する情報を戻します。

UNUSED_SPACE プロシージャ

このプロシージャは、オブジェクト（表、索引またはクラスタ）にある未使用領域に関する情報を戻します。

構文

```
DBMS_SPACE.UNUSED_SPACE (  
    segment_owner          IN  VARCHAR2,  
    segment_name           IN  VARCHAR2,  
    segment_type           IN  VARCHAR2,  
    total_blocks           OUT NUMBER,  
    total_bytes            OUT NUMBER,  
    unused_blocks          OUT NUMBER,  
    unused_bytes           OUT NUMBER,  
    last_used_extent_file_id OUT NUMBER,  
    last_used_extent_block_id OUT NUMBER,  
    last_used_block        OUT NUMBER,  
    partition_name         IN  VARCHAR2 DEFAULT NULL);
```

パラメータ

表 49-2 UNUSED_SPACE プロシージャのパラメータ

パラメータ	説明
segment_owner	分析するセグメントのスキーマ名。
segment_name	分析するセグメントのセグメント名。
segment_type	分析するセグメントのタイプは、次の中から選択します。 TABLE TABLE PARTITION TABLE SUBPARTITION INDEX INDEX PARTITION INDEX SUBPARTITION CLUSTER LOB
total_blocks	セグメント内のブロック合計数を戻します。
total_bytes	セグメント内のブロック合計数をバイト単位で戻します。
unused_blocks	未使用のブロック数を戻します。
unused_bytes	未使用のブロック数をバイト単位で戻します。
last_used_extent_file_id	データを含んだ最新エクステントのファイル ID を戻します。
last_used_extent_block_id	データを含んだ最新エクステントのブロック ID を戻します。
last_used_block	データを含んだエクステント内の最終ブロックを戻します。
partition_name	分析するセグメントのパーティション名。 これは、パーティション表についてのみ使用します。サブパーティションの名前は、パーティションの構成時に使用します。

FREE_BLOCKS プロシージャ

このプロシージャは、オブジェクト（表、索引またはクラスタ）にある空きブロックに関する情報を戻します。

構文

```
DBMS_SPACE.FREE_BLOCKS (
    segment_owner      IN  VARCHAR2,
    segment_name       IN  VARCHAR2,
    segment_type       IN  VARCHAR2,
    freelist_group_id  IN  NUMBER,
    free_blks          OUT NUMBER,
    scan_limit         IN  NUMBER DEFAULT NULL,
    partition_name     IN  VARCHAR2 DEFAULT NULL);
```

プラグマ

```
pragma restrict_references(free_blocks,WNDS);
```

パラメータ

表 49-3 FREE_BLOCKS プロシージャのパラメータ

パラメータ	説明
segment_owner	分析するセグメントのスキーマ名。
segment_name	分析するセグメントのセグメント名。
segment_type	分析するセグメントのタイプは、次の中から選択します。 TABLE TABLE PARTITION TABLE SUBPARTITION INDEX INDEX PARTITION INDEX SUBPARTITION CLUSTER LOB
freelist_group_id	空きリスト・サイズが計算される空きリスト・グループ（インスタンス）。
free_blks	指定されたグループに関する空きブロック数を戻します。
scan_limit	読み込む空きリストのブロックの最大数（オプション）。 空きリストに X 個のブロックがある場合は、X 個の走査制限を使用します。

表 49-3 FREE_BLOCKS プロシージャのパラメータ

パラメータ	説明
partition_name	分析するセグメントのパーティション名。 これは、パーティション表についてのみ使用します。サブパーティションの名前は、パーティションの構成時に使用します。

例

例 1:

次の例では、必要なバインド変数を宣言してから実行します。

```
DBMS_SPACE.UNUSED_SPACE('SCOTT', 'EMP', 'TABLE', :total_blocks,  
:total_bytes, :unused_blocks, :unused_bytes, :lastextf,  
:last_extb, :lastusedb);
```

これにより、SCOTT スキーマの EMP 表に、バインド変数に関する未使用領域の情報が入ります。

例 2:

次の例では、4 つの空きリスト・グループを持つ SCOTT スキーマにある CLUS クラスタが使用されます。そして、CLUS の空きリスト・グループ 3 にあるブロック数が戻されます。

```
DBMS_SPACE.FREE_BLOCKS('SCOTT', 'CLUS', 'CLUSTER', 3, :free_blocks);
```

注意: scan_limit が正の数でない場合は、エラーが発生します。

DBMS_SPACE_ADMIN

DBMS_SPACE_ADMIN パッケージは、ローカルに管理される表領域に対する機能を提供します。

セキュリティ

このパッケージは、SYS 権限で実行されるため、このパッケージを実行する権限を持つユーザーは、ビットマップも操作できます。

定数

SEGMENT_VERIFY_EXTENTS	セグメントが所有する領域の使用状況がビットマップに適切に反映されていることを検証します。
SEGMENT_VERIFY_EXTENTS_GLOBAL	セグメントが所有する領域の使用状況がビットマップに適切に反映されており、この領域が他のセグメントから要求されていないことを検証します。
SEGMENT_MARK_CORRUPT	一時セグメントを破損としてマークします。これにより、ディクショナリからの排除が容易になります（領域の再生なし）。
SEGMENT_MARK_VALID	破損した一時セグメントを有効としてマークします。これは、セグメントのエクステント・マップまたは他の場所での破損が解決され、セグメントが正常に削除できる場合に便利です。
SEGMENT_DUMP_EXTENT_MAP	指定のセグメントのエクステント・マップをダンプします。
TABLESPACE_VERIFY_BITMAP	表領域のビットマップを、その表領域内のセグメントのエクステント・マップを使用して検証し、すべてが一致していることを検証します。
TABLESPACE_EXTENT_MAKE_FREE	この範囲（エクステント）の領域をビットマップ上で空き領域にします。
TABLESPACE_EXTENT_MAKE_USED	この範囲（エクステント）の領域をビットマップ上で使用領域にします。

サブプログラムの要約

表 50-1 DBMS_SPACE_ADMIN パッケージのサブプログラム

サブプログラム	説明
50-4 ページの SEGMENT_VERIFY プロシージャ	セグメントのエクステント・マップの一貫性を検証します。
50-5 ページの SEGMENT_CORRUPT プロシージャ	セグメントを破損または有効としてマークし、適切なエラーのリカバリを可能にします。
50-6 ページの SEGMENT_DROP_CORRUPT プロシージャ	現在破損としてマークされているセグメントを削除します（領域の再生なし）。
50-6 ページの SEGMENT_DUMP プロシージャ	指定セグメントのセグメント・ヘッダーとエクステント・マップをダンプします。
50-7 ページの TABLESPACE_VERIFY プロシージャ	表領域内のセグメントについて、ビットマップとエクステント・マップが同期していることを検証します。
50-8 ページの TABLESPACE_FIX_BITMAPS プロシージャ	適切な DBA 範囲（エクステント）を、ビットマップ上で空きまたは使用領域としてマークします。
50-9 ページの TABLESPACE_REBUILD_BITMAPS プロシージャ	適切なビットマップを再作成します。
50-9 ページの TABLESPACE_REBUILD_QUOTAS プロシージャ	指定の表領域について割当て制限を再作成します。
50-10 ページの TABLESPACE_MIGRATE_FROM_LOCAL プロシージャ	ローカルに管理される表領域を、ディクショナリ管理の表領域に移行します。
50-11 ページの TABLESPACE_MIGRATE_TO_LOCAL プロシージャ	ディクショナリ管理形式からローカル管理形式に、表領域を移行します。
50-12 ページの TABLESPACE_RELOCATE_BITMAPS プロシージャ	指定先にビットマップを再割当てします。
50-13 ページの TABLESPACE_FIX_SEGMENT_STATES プロシージャ	移行が異常終了した表領域内のセグメントの状態を修正します。

SEGMENT_VERIFY プロシージャ

このプロシージャは、セグメントのエクステント・マップがビットマップと一致していることを検証します。

構文

```
DBMS_SPACE_ADMIN.SEGMENT_VERIFY (  
    tablespace_name      IN    VARCHAR2,  
    header_relative_file  IN    POSITIVE,  
    header_block         IN    POSITIVE,  
    verify_option        IN    POSITIVE DEFAULT SEGMENT_VERIFY_EXTENTS);
```

パラメータ

表 50-2 SEGMENT_VERIFY プロシージャのパラメータ

パラメータ	説明
tablespace_name	セグメントが常駐している表領域名
header_relative_file	セグメント・ヘッダーの相対ファイル番号
header_block	セグメント・ヘッダーのブロック番号
verify_option	チェックの種類: SEGMENT_VERIFY_EXTENTS または SEGMENT_VERIFY_EXTENTS_GLOBAL

使用上の注意

すべての DBA 範囲で誤った領域表現として検出された異常は、DBA 範囲、ビットマップ・ブロック、ビットマップ・ブロック範囲、異常情報としてトレース・ファイルに出力されます。レポートされる問題の種類は、空きとみなされない空き領域、空きとみなされた使用領域、および複数のセグメントで使用中とみなされた同一領域です。

例

次の例では、相対ファイル番号 4、ブロック番号 33 のセグメント・ヘッダーを持つセグメントで、エクステント・マップとビットマップが同期していることを検証します。

```
EXECUTE DBMS_SPACE_ADMIN.SEGMENT_VERIFY('USERS', 4, 33, 1);
```

注意： DBMS_SPACE_ADMIN パッケージのすべての例では、SCOTT.EMP を含んだ表領域 USERS が使用されます。

SEGMENT_CORRUPT プロシージャ

このプロシージャは、セグメントを破損または有効としてマークし、適切なエラーのリカバリを可能にします。

構文

```
DBMS_SPACE_ADMIN.SEGMENT_CORRUPT (  
    tablespace_name      IN    VARCHAR2,  
    header_relative_file  IN    POSITIVE,  
    header_block          IN    POSITIVE,  
    corrupt_option        IN    POSITIVE DEFAULT SEGMENT_MARK_CORRUPT);
```

パラメータ

表 50-3 SEGMENT_CORRUPT プロシージャのパラメータ

パラメータ	説明
tablespace_name	セグメントが常駐している表領域名
header_relative_file	セグメント・ヘッダーの相対ファイル番号
header_block	セグメント・ヘッダーのブロック番号
corrupt_option	SEGMENT_MARK_CORRUPT (デフォルト) または SEGMENT_MARK_VALID

例

次の例では、セグメントを破損としてマークします。

```
EXECUTE DBMS_SPACE_ADMIN.SEGMENT_CORRUPT('USERS', 4, 33, 3);
```

次の例では、破損のセグメントを有効としてマークします。

```
EXECUTE DBMS_SPACE_ADMIN.SEGMENT_CORRUPT('USERS', 4, 33, 4);
```

SEGMENT_DROP_CORRUPT プロシージャ

このプロシージャは、現在破損としてマークされているセグメントを削除します（領域の再生なし）。これを実行するには、セグメントは一時的としてマークされている必要があります。破損のセグメントを一時的としてマークするには、セグメントで DROP コマンドを発行します。

セグメントのための領域は解放されないので、TABLESPACE_FIX_BITMAPS または TABLESPACE_REBUILD_BITMAPS プロシージャを使用して調整する必要があります。これについては、この章で後述します。

構文

```
DBMS_SPACE_ADMIN.SEGMENT_DROP_CORRUPT (
    tablespace_name      IN    VARCHAR2,
    header_relative_file  IN    POSITIVE,
    header_block          IN    POSITIVE);
```

パラメータ

表 50-4 SEGMENT_DROP_CORRUPT プロシージャのパラメータ

パラメータ	説明
tablespace_name	セグメントが常駐している表領域名
header_relative_file	セグメント・ヘッダーの相対ファイル番号
header_block	セグメント・ヘッダーのブロック番号

例

```
EXECUTE DBMS_SPACE_ADMIN.SEGMENT_DROP_CORRUPT('USERS', 4, 33);
```

SEGMENT_DUMP プロシージャ

このプロシージャは、指定のセグメントのセグメント・ヘッダーとエクステント・マップのブロックをダンプします。

構文

```
DBMS_SPACE_ADMIN.SEGMENT_DUMP (
    tablespace_name      IN    VARCHAR2,
    header_relative_file  IN    POSITIVE,
    header_block          IN    POSITIVE,
    dump_option           IN    POSITIVE DEFAULT SEGMENT_DUMP_EXTENT_MAP);
```

パラメータ

表 50-5 SEGMENT_DUMP プロシージャのパラメータ

パラメータ	説明
tablespace_name	セグメントが常駐している表領域名
header_relative_file	セグメント・ヘッダーの相対ファイル番号
header_block	セグメント・ヘッダーのブロック番号
dump_option	SEGMENT_DUMP_EXTENT_MAP

例

```
EXECUTE DBMS_SPACE_ADMIN.SEGMENT_DUMP('USERS', 4, 33);
```

TABLESPACE_VERIFY プロシージャ

このプロシージャは、表領域内のセグメントについて、ビットマップとエクステント・マップが同期していることを検証します。

構文

```
DBMS_SPACE_ADMIN.TABLESPACE_VERIFY (  
    tablespace_name      IN    VARCHAR2,  
    verify_option        IN    POSITIVE DEFAULT TABLESPACE_VERIFY_BITMAP);
```

パラメータ

表 50-6 TABLESPACE_VERIFY プロシージャのパラメータ

パラメータ	説明
tablespace_name	表領域の名前
verify_option	TABLESPACE_VERIFY_BITMAP

例

```
EXECUTE DBMS_SPACE_ADMIN.TABLESPACE_VERIFY('USERS');
```

TABLESPACE_FIX_BITMAPS プロシージャ

このプロシージャは、適切な DBA 範囲（エクステント）を、ビットマップ上で空きまたは使用領域としてマークします。

構文

```
DBMS_SPACE_ADMIN.TABLESPACE_FIX_BITMAPS (  
    tablespace_name      IN    VARCHAR2,  
    dbarange_relative_file IN    POSITIVE,  
    dbarange_begin_block IN    POSITIVE,  
    dbarange_end_block   IN    POSITIVE,  
    fix_option           IN    POSITIVE);
```

パラメータ

表 50-7 TABLESPACE_FIX_BITMAPS プロシージャのパラメータ

パラメータ	説明
tablespace_name	表領域の名前
dbarange_relative_file	DBA 範囲（エクステント）の相対ファイル番号
dbarange_begin_block	エクステントの開始ブロック番号
dbarange_end_block	エクステントの終了ブロック番号
fix_option	TABLESPACE_EXTENT_MAKE_FREE または TABLESPACE_EXTENT_MAKE_USED

例

次の例では、相対ファイル番号 4 の、ブロック番号 33 からブロック番号 83 までの 50 ブロックを、ビットマップ内のビットを USED としてマークします。

```
EXECUTE DBMS_SPACE_ADMIN.TABLESPACE_FIX_BITMAPS('USERS', 4, 33, 83, 7);
```

または、オプションに 8 を指定して、ビットマップ内 FREE としてビットをマークします。BEGIN と END ブロックは、エクステント境界内にあり、エクステントの倍数である必要があります。そうでない場合は、エラーが発生します。

TABLESPACE_REBUILD_BITMAPS プロシージャ

このプロシージャは、適切なビットマップを再作成します。ビットマップ・ブロックの DBA が指定されていない場合は、指定の表領域のすべてのビットマップが再作成されます。

構文

```
DBMS_SPACE_ADMIN.TABLESPACE_REBUILD_BITMAPS (  
    tablespace_name      IN    VARCHAR2,  
    bitmap_relative_file  IN    POSITIVE  DEFAULT NULL,  
    bitmap_block          IN    POSITIVE  DEFAULT NULL);
```

パラメータ

表 50-8 TABLESPACE_REBUILD_BITMAPS プロシージャのパラメータ

パラメータ	説明
tablespace_name	表領域の名前
bitmap_relative_file	再作成するビットマップ・ブロックの相対ファイル番号
bitmap_block	再作成するビットマップ・ブロックのブロック番号

例

次の例では、USERS 表領域にあるすべてのファイルについて、ビットマップを再作成します。

```
EXECUTE DBMS_SPACE_ADMIN.TABLESPACE_REBUILD_BITMAPS('USERS');
```

注意： 現在は、すべてのファイルの再作成のみサポートされています。

TABLESPACE_REBUILD_QUOTAS プロシージャ

このプロシージャは、指定の表領域に関する割当て制限を再作成します。

構文

```
DBMS_SPACE_ADMIN.TABLESPACE_REBUILD_QUOTAS (  
    tablespace_name      IN    VARCHAR2);
```

パラメータ

表 50-9 TABLESPACE_REBUILD_QUOTAS プロシージャのパラメータ

パラメータ	説明
tablespace_name	表領域の名前

例

```
EXECUTE DBMS_SPACE_ADMIN.TABLESPACE_REBUILD_QUOTAS('USERS');
```

TABLESPACE_MIGRATE_FROM_LOCAL プロシージャ

このプロシージャは、ローカルに管理される表領域をディクショナリ管理の表領域に移行します。

構文

```
DBMS_SPACE_ADMIN.TABLESPACE_MIGRATE_FROM_LOCAL (  
    tablespace_name          IN      VARCHAR2);
```

パラメータ

表 50-10 TABLESPACE_MIGRATE_FROM_LOCAL プロシージャのパラメータ

パラメータ	説明
tablespace_name	表領域の名前

使用上の注意

表領域はオンラインで保持し、移行中に読み込み書き込みを行う必要があります。一時表領域の移行と SYSTEM 表領域の移行はサポートされていません。

注意： SYSTEM 表領域の移行は、将来サポートされる予定です。

例

```
EXECUTE DBMS_SPACE_ADMIN.TABLESPACE_MIGRATE_FROM_LOCAL('USERS');
```


TABLESPACE_MIGRATE_TO_LOCAL プロシージャ

このプロシージャを使用して、ディクショナリ管理形式からローカル管理形式に表領域を移行します。ローカル管理形式に移行した表領域は、ユーザーが管理します。

構文

```
TABLESPACE_MIGRATE_TO_LOCAL(tablespace_name, allocation_unit, relative_fno)
```

パラメータ

表 50-11 TABLESPACE_MIGRATE_TO_LOCAL のパラメータ

パラメータ名	用途	データ型	パラメータ型
tablespace_name	移行する表領域の名前	VARCHAR	IN
allocation_unit	表領域の単位サイズ（割当て可能な領域の最小許容チャンク・サイズ）	INTEGER	IN
relative_fno	ビットマップ・ブロックが配置されるファイルの相対ファイル番号（オプション）	INTEGER	IN

割当て単位に関する注意 割当て単位はオプションで指定できます。デフォルトは、表領域の全エクステント（使用中または空き）の最大公約数に基づいてシステムが計算します。この数値は、表領域の MINIMUM EXTENT を基にさらに削減されます（MINIMUM EXTENT の指定がない場合は 5）。したがって、計算された値が表領域の MINIMUM EXTENT を超えることはありません。各ファイルの最後にある使用可能エクステントは、GCD 計算には含まれません。指定する単位サイズは、システムが計算した単位サイズの因数にしてください。それ以外の場合はエラー・メッセージが戻ります。

相対ファイル番号の注意 相対ファイル番号のパラメータは、目的のファイルにビットマップを配置するために使用されます。ファイルに領域がなかった場合は、エラーが発行されます。指定するデータ・ファイルは、移行する表領域の一部であることが必要です。データ・ファイルを指定しない場合、初期ビットマップ・ブロックを配置するデータ・ファイルはシステムが選択します。初期ビットマップに必要な領域が見つからなかった場合は、エラーになります。

使用上の注意 表領域はオンラインで保持し、移行中に読み込み書き込みを行う必要があります。一時表領域は移行できないことに注意してください。SYSTEM 表領域の移行は、このリリースでは行えません。

例

次の例は、最小エクステント・サイズ 1MB で表領域 'TS1' を移行します。

```
execute dbms_space_admin.tablespace_migrate_to_local('TS1', 512, 2);
```

ビットマップは、相対ファイル番号 2 のファイルに配置されます。

TABLESPACE_RELOCATE_BITMAPS プロシージャ

このプロシージャを使用して、ビットマップを指定先に再配置します。表領域のディクショナリ管理からローカル管理形式への移行の結果は、ビットマップ・ブロックを含んだ領域ヘッダー・セグメントの作成につながる場合があります。領域ヘッダー・セグメントは、ユーザー・データとして取り扱われます。ユーザーがファイルを領域ヘッダー・セグメント以下に明示的にサイズ変更しようとする、エラーが発行されます。制御情報を別の場所に移動してからファイルのサイズを変更する場合は、tablespace_relocate_bitmaps コマンドを使用します。

構文

```
TABLESPACE_RELOCATE_BITMAPS (tablespace_name, relative_fno, block_number)
```

パラメータ

表 50-12 TABLESPACE_RELOCATE_BITMAPS のパラメータ

パラメータ名	用途	データ型	パラメータ型
tablespace_name	表領域の名前	VARCHAR	IN
relative_fno	宛先ファイルの相対ファイル番号	NUMBER	IN
block_number	宛先データベースのブロック番号	NUMBER	IN

使用上の注意

ビットマップの再配置中、表領域はオンラインで読み込みと書き込みを維持する必要があります。移行したローカルに管理される表領域でのみ実行できます。

例

```
execute dbms_space_admin.tablespace_relocate_bitmaps('TS1', 3, 4);
```

ビットマップをファイル 3、ブロック 4 に移動します。

注意： ソース・アドレスと宛先アドレスはオーバーラップできません。宛先ブロック番号は単位の境界まで切り下げられます。宛先位置にユーザー・データが存在する場合は、エラーになります。

TABSPACE_FIX_SEGMENT_STATES プロシージャ

このプロシージャを使用して、移行が異常終了した表領域内のセグメントの状態を修正します。ローカルから、またはローカルへの表領域の移行中、セグメントは転送状態に置かれます。移行が異常終了した場合は、イベント 10906 の設定時点でセグメントの状態が SMON によって訂正されます。セグメントがこのような転送状態にあるデータベースは、ダウングレードできません。このプロシージャは、このようなセグメントの状態を修正するために使用できます。

構文

```
TABSPACE_FIX_SEGMENT_STATES(tablespace_name);
```

パラメータ

表 50-13 TABSPACE_FIX_SEGMENT_STATES のパラメータ

パラメータ名	用途	データ型	パラメータ型
tablespace_name	セグメントを修正する必要がある表領域の名前	VARCHAR	IN

使用上の注意

このプロシージャがコールされたとき、表領域はオンラインで読み込みと書き込みを維持する必要があります。

例

```
execute dbms_space_admin.tablespace_fix_segment_states('TS1');
```

詳細な例

1. TABSPACE_VERIFY で、一部のセグメントにビットマップ内空き領域としてマークされた割当てブロックがあるが、セグメント間オーバーラップについてはレポートされていないことが検出されました。この場合は、次のアクションをお勧めします。
 - SEGMENT_EXTENT_MAP_DUMP プロシージャをコールして、セグメントに割り当てられている DBA 範囲をダンプします。

- 各範囲について、TABLESPACE_MAKE_USED オプションを指定した TABLESPACE_FIX_BITMAPS をコールして、その領域を使用領域としてマークします。
 - TABLESPACE_REBUILD_QUOTAS をコールして、割当て制限を修正します。
2. ビットマップ内空き領域とマークされたセグメント・ブロックの一部があるため、セグメントを削除できません。このセグメントは破損として自動的にマークされています。この場合は、次の処理を行います。
- SEGMENT_CHECK_ALL オプションを指定した SEGMENT_VERIFY プロシージャをコールします。オーバーラップがレポートされていない場合は、次の処理を行います。
 - * SEGMENT_EXTENT_MAP_DUMP プロシージャをコールして、セグメントに割り当てられている DBA 範囲をダンプします。
 - * 各範囲について、TABLESPACE_MAKE_FREE オプションを指定した TABLESPACE_FIX_BITMAPS をコールして、その領域を空き領域としてマークします。
 - * SEGMENT_DROP_CORRUPT をコールして、SEG\$ エントリを削除します。
 - * TABLESPACE_REBUILD_QUOTAS をコールして、割当て制限を修正します。
3. TABLESPACE_VERIFY で、オーバーラップがレポートされました。この場合、以前の内部エラーに基づいて、実データの一部を破棄する必要がある場合があります。破棄するオブジェクトに表 T1 を選択し、次の処理を行います。
- T1 がオーバーラップしているすべてのオブジェクトのリストを作成します。
 - SQL を使用して、T1 を削除します。必要に応じて、SEGMENT_DROP_CORRUPT を後に続けます。
 - T1 がオーバーラップしていたすべてのオブジェクトについて、SEGMENT_VERIFY をコールします。必要に応じて、TABLESPACE_FIX_BITMAPS をコールして、適切なビットマップ・ブロックを使用領域としてマークします。
 - TABLESPACE_VERIFY を再実行します。
4. ビットマップ・ブロックの 1 セットがメディア破損です。次の処理を行います。
- すべてのビットマップ・ブロックについて、または 1 つのブロックのみが破損している場合、単一のブロックについて、TABLESPACE_REBUILD_BITMAPS をコールします。
 - TABLESPACE_REBUILD_QUOTAS をコールして、割当て制限を再作成します。
 - TABLESPACE_VERIFY をコールして、ビットマップが一貫しているかどうかをチェックします。

Oracle によって、ユーザーは動的 SQL を使用するストアド・プロシージャと無名 PL/SQL ブロックを記述できます。動的 SQL 文は、ユーザーのソース・プログラムに埋め込まれていません。実行時にプログラムに入力されるか、またはプログラムによって作成されるように、文字列に格納されています。これによって、ユーザーは用途の広いプロシージャを作成できます。たとえば、この動的 SQL によって、実行時まで名前がわからない表で動作するプロシージャを作成できます。

また、データ操作言語（DML）文やデータ定義言語（DDL）文はいずれも、DBMS_SQL パッケージを使用して解析できます。したがって、PL/SQL を使用して DDL 文を直接解析できます。たとえば、DBMS_SQL パッケージが提供する PARSE プロシージャを使用することによって、ストアド・プロシージャ内から DROP TABLE 文の入力を選択できるようになりました。

注意： Oracle8i は、DBMS_SQL パッケージのかわりにシステム固有の動的 SQL を導入しています。システム固有の動的 SQL を使用して、動的 SQL 文を PL/SQL ブロックに直接設定できます。

ほとんどの場合、DBMS_SQL のかわりにシステム固有の動的 SQL を使用できます。システム固有の動的 SQL は、DBMS_SQL と比べて使用方法が簡単でパフォーマンスが向上します。

関連項目： システム固有の動的 SQL の詳細は、『Oracle8i PL/SQL ユーザーズ・ガイドおよびリファレンス』を参照してください。

DBMS_SQL とシステム固有の動的 SQL の比較は、『Oracle8i アプリケーション開発者ガイド 基礎編』を参照してください。

DBMS_SQL の使用方法

ストアド・プロシージャ内から動的 SQL を使用する機能は一般的に、Oracle コール・インタフェース (OCI) の手順に従っています。

関連項目：『Oracle8i コール・インタフェース・プログラマーズ・ガイド』

PL/SQL は、C などの他の一般的なプログラム言語とは、多少異なります。たとえば、ユーザーはアドレス（ポインタとも呼ばれます）を PL/SQL で参照できません。そのため、Oracle コール・インタフェースと DBMS_SQL パッケージの間には、いくつか相違点があります。相違点は、次のとおりです。

- OCI はアドレスによるバインドを使用するのに対して、DBMS_SQL パッケージは値によるバインドを使用します。
- DBMS_SQL では、無名ブロックの OUT パラメータの値を検索するには、VARIABLE_VALUE をコールする必要があります。また、行をフェッチして、行内の列の値を実際にプログラムに取り出した後で、COLUMN_VALUE をコールする必要があります。
- 現行リリースの DBMS_SQL パッケージは、CANCEL カーソル・プロシージャを提供していません。
- NULL は PL/SQL 変数の値として完全にサポートされているため、標識変数は不要です。

DBMS_SQL パッケージの使用例は、次のとおりです。このコードは、Oracle コール・インタフェースのユーザーにとってはかなり簡潔です。

例

この文のテキストはコンパイル時に判明しているため、この例では、動的 SQL を実際に使用する必要はありません。ここでは、このパッケージの概念をわかりやすく説明します。

DEMO プロシージャは、DEMO の実行時に指定した給与よりも高い給与のすべての従業員を EMP 表から削除します。

```
CREATE OR REPLACE PROCEDURE demo(salary IN NUMBER) AS
    cursor_name INTEGER;
    rows_processed INTEGER;
BEGIN
    cursor_name := dbms_sql.open_cursor;
    DBMS_SQL.PARSE(cursor_name, 'DELETE FROM emp WHERE sal > :x',
        dbms_sql.native);
    DBMS_SQL.BIND_VARIABLE(cursor_name, ':x', salary);
    rows_processed := dbms_sql.execute(cursor_name);
    DBMS_SQL.close_cursor(cursor_name);
```

```
EXCEPTION
WHEN OTHERS THEN
    DBMS_SQL.CLOSE_CURSOR(cursor_name);
END;
```

定数

```
v6 constant INTEGER      := 0;
native constant INTEGER := 1;
v7 constant INTEGER      := 2;
```

型

```
TYPE varchar2s IS TABLE OF VARCHAR2(256) INDEX BY BINARY_INTEGER;
TYPE desc_rec  IS RECORD (
    col_type          BINARY_INTEGER := 0,
    col_max_len       BINARY_INTEGER := 0,
    col_name          VARCHAR2(32)   := '',
    col_name_len      BINARY_INTEGER := 0,
    col_schema_name   VARCHAR2(32)   := '',
    col_schema_name_len BINARY_INTEGER := 0,
    col_precision     BINARY_INTEGER := 0,
    col_scale         BINARY_INTEGER := 0,
    col_charsetid     BINARY_INTEGER := 0,
    col_charsetform   BINARY_INTEGER := 0,
    col_null_ok       BOOLEAN        := TRUE);
TYPE desc_tab IS TABLE OF desc_rec INDEX BY BINARY_INTEGER;
```

複数の SQL 型

```
type Number_Table IS TABLE OF NUMBER          INDEX BY BINARY_INTEGER;
type Varchar2_Table IS TABLE OF VARCHAR2(2000) INDEX BY BINARY_INTEGER;
type Date_Table IS TABLE OF DATE              INDEX BY BINARY_INTEGER;
type Blob_Table IS TABLE OF BLOB              INDEX BY BINARY_INTEGER;
type Clob_Table IS TABLE OF CLOB              INDEX BY BINARY_INTEGER;
type Bfile_Table IS TABLE OF BFILE            INDEX BY BINARY_INTEGER;
type Urowid_Table IS TABLE OF UROWID          INDEX BY BINARY_INTEGER;
```

例外

```
inconsistent_type exception;  
pragma exception_init(inconsistent_type, -6562);
```

この例外は、指定した OUT パラメータ（要求した値を設定するパラメータ）の型がその値の型と異なる場合、プロシージャ COLUMN_VALUE または VARIABLE_VALUE で発生します。

実行フロー

OPEN_CURSOR

SQL 文を処理するためには、オープン・カーソルが必要です。OPEN_CURSOR ファンクションをコールすると、ユーザーは Oracle が保持している有効なカーソルを表すデータ構造のカーソル ID 番号を受け取ります。これらのカーソルは、プリコンパイラ、OCI または PL/SQL レベルで定義されたカーソルとは異なり、DBMS_SQL パッケージでのみ使用されます。

PARSE

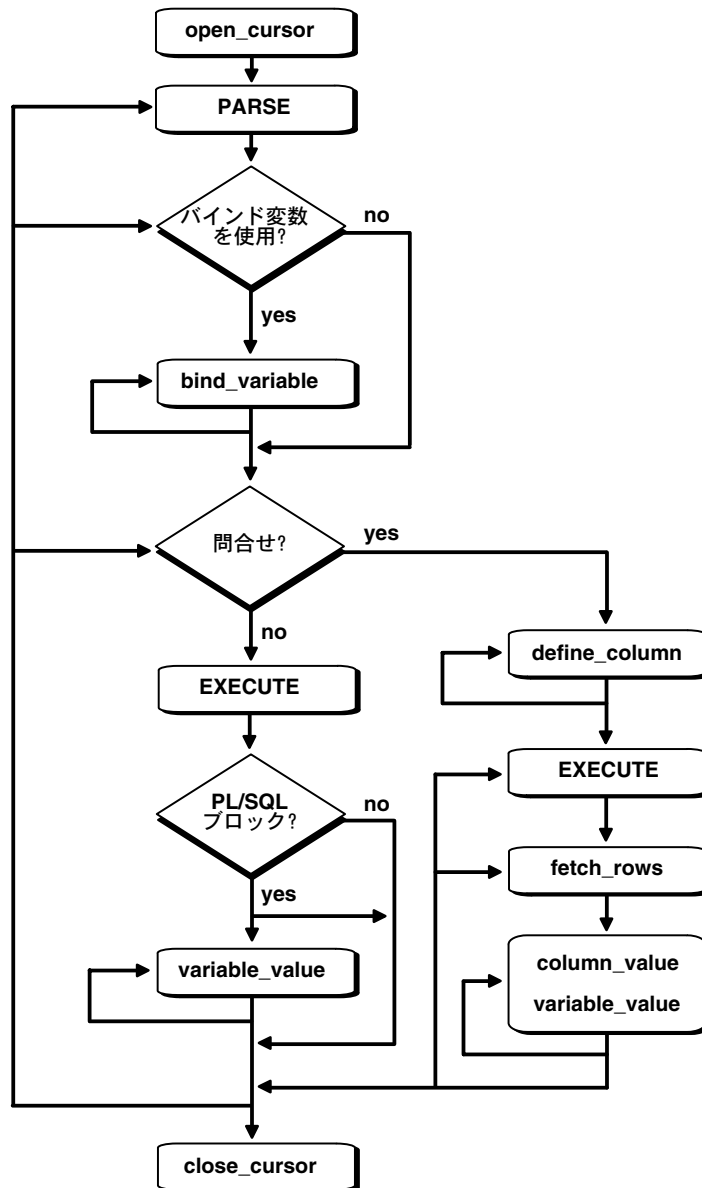
SQL 文はすべて、PARSE プロシージャをコールして解析する必要があります。文を解析することによって、その文の構文がチェックされ、プログラム内のカーソルに関連付けられます。

関連項目： SQL 文の解析方法は、『Oracle8i 概要』を参照してください。

DML 文または DDL 文はすべて解析できます。DDL 文は解析時に実行され、暗黙のコミットを実行します。

注意： パッケージやプロシージャを削除するために DDL 文を解析するときに、パッケージ内のプロシージャが使用中の場合はデッドロックが起こる可能性があります。プロシージャへのコール後は、実行がユーザー側に戻されるまで、そのプロシージャは使用中であるとみなされます。このようなデッドロックは、5 分後にタイムアウトします。

図 51-1 DBMS_SQL 実行フロー



BIND_VARIABLE または BIND_ARRAY

多くの DML 文では、プログラム内のデータを Oracle に入力することが必要です。実行時に提供する入力データを含んでいる SQL 文を定義する場合は、SQL 文内のプレースホルダを使用して、データの提供場所にマークを付ける必要があります。

SQL 文内の各プレースホルダに対してバインド・プロシージャ (BIND_VARIABLE プロシージャまたは BIND_ARRAY プロシージャ) の 1 つをコールして、プログラム内の変数の値 (または配列の値) をプレースホルダに提供する必要があります。SQL 文が引き続き実行されると、Oracle は、ユーザーのプログラムが入力変数と出力変数、またはバインド変数に設定したデータを使用します。

DBMS_SQL は、その都度異なるバインド変数を使用して DML 文を繰り返し実行できます。BIND_ARRAY プロシージャを使用すると、スカラーの集合をバインドでき、それぞれの値は 1 回の EXECUTE につき 1 度のみ入力変数として使用されます。これは、OCI がサポートする配列インタフェースに類似しています。

DEFINE_COLUMN、DEFINE_COLUMN_LONG または DEFINE_ARRAY

SELECT 文内で選択されている行の列は、選択リスト内での相対位置 (左から右) によって識別されます。問合せの場合は、定義プロシージャの 1 つ (DEFINE_COLUMN、DEFINE_COLUMN_LONG または DEFINE_ARRAY) をコールして SELECT 値を受け入れる変数を指定する必要があります。これは、INTO 句が静的問合せに対して行う方法とほとんど同じです。

DEFINE_COLUMN を使用して LONG 列以外の列を定義するのと同様に、DEFINE_COLUMN_LONG プロシージャを使用して LONG 列を定義します。COLUMN_VALUE_LONG を使用して LONG 列からフェッチする前に、DEFINE_COLUMN_LONG をコールする必要があります。

DEFINE_ARRAY プロシージャを使用して、行を単一の SELECT 文でフェッチする PL/SQL コレクションを定義します。DEFINE_ARRAY は、1 回のフェッチで複数行をフェッチするインタフェースを提供します。COLUMN_VALUE プロシージャで行をフェッチする前に、DEFINE_ARRAY をコールする必要があります。

EXECUTE

EXECUTE ファンクションをコールして、SQL 文を実行します。

FETCH_ROWS または EXECUTE_AND_FETCH

FETCH_ROWS ファンクションは、問合せを満たす行を検索します。フェッチが行を検索できなくなるまで、連続する各フェッチは別の行を検索します。1 回のみの実行に対して EXECUTE をコールしている場合は、EXECUTE の次に FETCH_ROWS をコールするより、EXECUTE_AND_FETCH をコールする方が効率的です。

VARIABLE_VALUE、COLUMN_VALUE または COLUMN_VALUE_LONG

問合せの場合は、COLUMN_VALUE をコールして、FETCH_ROWS コールで検索する列の値を判別します。PL/SQL プロシージャへのコールまたは returning 句がある DML 文を含んだ無名ブロックの場合は、VARIABLE_VALUE をコールして、文の実行時に出力変数に割り当てられた値を検索します。

LONG データベース列（サイズは最大 2GB まで可能）の一部のみをフェッチするには、COLUMN_VALUE_LONG プロシージャを使用します。列値へのオフセット（単位はバイト）とフェッチするバイト数を指定できます。

CLOSE_CURSOR

セッションでカーソルが不要な場合は、CLOSE_CURSOR をコールしてカーソルをクローズします。Oracle Open Gateway を使用している場合は、これ以外のときにもカーソルのクローズが必要になる場合があります。追加情報は、Oracle Open Gateway の関連ドキュメントを参照してください。

カーソルをクローズしないと、カーソルが不要になっても、そのカーソルが使用しているメモリは割り当てられたままになります。

セキュリティ

定義者権限モジュール

定義者権限モジュールは、モジュールの所有者の権限下で実行されます。定義者権限モジュールからコールされた DBMS_SQL サブプログラムは、モジュールで定義されたスキーマに関して実行されます。

注意： Oracle 8i より前のバージョンでは、すべての PL/SQL ストアド・プロシージャとパッケージが定義者権限モジュールでした。

実行者権限モジュール

実行者権限モジュールは、モジュールの実行者の権限下で実行されます。したがって、実行者権限モジュールからコールされた DBMS_SQL サブプログラムは、モジュールの実行者の権限下で実行されます。

モジュールに current_user に設定された AUTHID があると、未修飾の名前は、実行者のスキーマに関連付けて変換されます。

例 income は USER1 のスキーマにある実行者権限ストアド・プロシージャで、USER2 には、そのストアド・プロシージャに対する EXECUTE 権限が付与されています。

```
CREATE PROCEDURE income(amount number)
  AUTHID current_user IS
  c number;
  n number;
BEGIN
  c:= dbms_sql.open_cursor;
  dbms_sql.parse(c, 'insert into accts(''income'', :1)', dbms_sql.native);
  dbms_sql.bind_variable(c, '1', amount);
  n := dbms_sql.execute(c);
  dbms_sql.close_cursor(c);
END;
```

USER1 が USER1.income をコールする場合は、USER1 の権限が使用され、未修飾名の名前解決が USER1 のスキーマに関して行われます。

USER2 が USER1.income をコールする場合は、USER2 の権限が使用され、未修飾名の名前解決（たとえば、accts）が USER2 のスキーマに関して行われます。

関連項目：『Oracle8i PL/SQL ユーザーズ・ガイドおよびリファレンス』

無名ブロック

無名 PL/SQL ブロックからコールされたすべての DBMS_SQL サブプログラムは、カレント・ユーザーの権限を使用して実行されます。

サブプログラムの要約

表 51-1 DBMS_SQL パッケージのサブプログラム

サブプログラム	説明
51-10 ページの OPEN_CURSOR ファンクション	新規カーソルのカーソル番号を戻します。
51-10 ページの PARSE プロシージャ	指定した文を解析します。
51-13 ページの BIND_VARIABLE および BIND_ARRAY プロシージャ	指定の値を指定の変数にバインドします。
51-13 ページの BIND_VARIABLE および BIND_ARRAY プロシージャ	指定の値を指定のコレクションにバインドします。

表 51-1 DBMS_SQL パッケージのサブプログラム

サブプログラム	説明
51-17 ページの DEFINE_COLUMN プロシージャ	指定したカーソルから選択し、SELECT 文のみで使用する列を定義します。
51-19 ページの DEFINE_ARRAY プロシージャ	指定したカーソルから選択し、SELECT 文のみで使用するコレクションを定義します。
51-21 ページの DEFINE_COLUMN_LONG プロシージャ	指定したカーソルから選択し、SELECT 文のみで使用する LONG 列を定義します。
51-21 ページの EXECUTE ファンクション	指定のカーソルを実行します。
51-22 ページの EXECUTE_AND_FETCH ファンクション	指定のカーソルを実行して、行をフェッチします。
51-22 ページの FETCH_ROWS ファンクション	指定のカーソルから行をフェッチします。
51-23 ページの COLUMN_VALUE プロシージャ	カーソルの指定位置にあるカーソル要素の値を戻します。
51-25 ページの COLUMN_VALUE_LONG プロシージャ	DEFINE_COLUMN_LONG で定義した LONG 列の選択された部分を戻します。
51-26 ページの VARIABLE_VALUE プロシージャ	指定のカーソルについて指定の変数の値を戻します。
51-28 ページの IS_OPEN ファンクション	指定のカーソルがオープンの場合に TRUE を戻します。
51-29 ページの DESCRIBE_COLUMNS プロシージャ	DBMS_SQL を介してオープンして解析されたカーソルの列を記述します。
51-31 ページの CLOSE_CURSOR プロシージャ	指定したカーソルをクローズして、メモリーを解放します。
51-32 ページの LAST_ERROR_POSITION ファンクション	エラーが発生した SQL 文テキスト内のバイト・オフセットを戻します。
51-32 ページの LAST_ROW_COUNT ファンクション	フェッチされた累積行数を戻します。
51-33 ページの LAST_ROW_ID ファンクション	最後に処理された行の ROWID を戻します。
51-33 ページの LAST_SQL_FUNCTION_CODE ファンクション	文の SQL ファンクション・コードを戻します。

OPEN_CURSOR ファンクション

このプロシージャは、新規のカーソルをオープンします。このカーソルが不要になった場合は、CLOSE_CURSOR をコールして、明示的にクローズする必要があります。

カーソルを使用すると、同じ SQL 文を繰り返し実行したり、新規の SQL 文を実行することができます。カーソルを再使用すると、対応するカーソル・データ領域の内容が新規 SQL 文の解析時に再設定されるため、再使用の前にクローズして再オープンする必要はありません。

構文

```
DBMS_SQL.OPEN_CURSOR  
RETURN INTEGER;
```

パラメータ

なし

プラグマ

```
pragma restrict_references (open_cursor, RNDS, WNDS);
```

戻り値

このファンクションは、新規カーソルのカーソル ID 番号を戻します。

PARSE プロシージャ

このプロシージャは、指定したカーソル内の指定した文を解析します。すべての文が即時に解析されます。さらに、DDL 文は、解析時にただちに実行されます。

PARSE プロシージャには、引数として VARCHAR2 文を使用するバージョンと VARCHAR2S (VARCHAR2 の表) を使用するバージョンの 2 つがあります。

注意： DBMS_SQL を使用して DDL 文を動的に実行すると、プログラムがハングする場合があります。たとえば、パッケージ内のプロシージャをコールすると、実行がユーザー側に戻るまでそのパッケージがロックされます。最初のロックを解放する前に、動的にパッケージを削除するなど、ロックの競合を引き起こす操作を行うとプログラムはハングします。

前述の構文を持つ SQL 文を解析するためのサイズは、32KB に制限されています。

構文

```
DBMS_SQL.PARSE (
  c                IN INTEGER,
  statement        IN VARCHAR2,
  language_flag    IN INTEGER);
```

PARSE プロシージャは、大規模な SQL 文について次の構文もサポートしています。

注意： このプロシージャは、PL/SQL 表の文の要素を連結し、その結果の文字列を解析します。このプロシージャを使用すると、文を分割することによって、単一の VARCHAR2 変数についての制限を超えた長い文を解析できます。

```
DBMS_SQL.PARSE (
  c                IN INTEGER,
  statement        IN VARCHAR2S,
  lb               IN INTEGER,
  ub               IN INTEGER,
  lfflg            IN BOOLEAN,
  language_flag    IN INTEGER);
```

パラメータ

表 51-2 PARSE プロシージャのパラメータ

パラメータ	説明
c	文を解析するカーソルの ID 番号。
statement	解析する SQL 文。 PL/SQL 文と異なり、SQL 文の終わりにはセミコロンを含めないでください。例は次のとおりです。 DBMS_SQL.PARSE(cursor1, 'BEGIN proc; END;', 2); DBMS_SQL.PARSE(cursor1, 'INSERT INTO tab values(1)', 2);
lb	文内の要素の下限。
ub	文内の要素の上限。

表 51-2 PARSE プロシージャのパラメータ

パラメータ	説明
lfflg	TRUE の場合、連結している各要素の後に改行を挿入します。
language_flag	Oracle で SQL 文を処理する方法を決定します。次のオプションが認識されます。 <ul style="list-style-type: none">■ V6（または 0）は、バージョン 6 の動作を指定します。■ NATIVE（または 1）は、プログラムの接続先のデータベースに関する通常の動作を指定します。■ V7（または 2）は、Oracle7 の動作を指定します。

注意： クライアント側コードは、リモート・パッケージの変数または定数を参照できないため、定数の値を明示的に使用する必要があります。

たとえば、次のコードは、クライアント側でコンパイルしません。

```
DBMS_SQL.PARSE(cur_hdl, stmt_str, dbms_sql.V7); -- uses constant
dbms_sql.V7
```

次のコードは、引数が明示的に指定されているので、クライアント側で有効です。

```
DBMS_SQL.PARSE(cur_hdl, stmt_str, 2); -- compiles on the client
```

大規模な SQL 文字列を解析するための VARCHAR2S データ型 32KB を超える SQL 文を解析するために、DBMS_SQL パッケージは、PL/SQL 表を使用して文字列の表を PARSE プロシージャに渡します。これらの文字列は連結された後、Oracle Server に渡されます。

ローカル変数を VARCHAR2S 表項目型として宣言し、次に、PARSE プロシージャを使用すると、大規模な SQL 文を VARCHAR2S として解析できます。

VARCHAR2S データ型の定義は、次のとおりです。

```
TYPE varchar2s IS TABLE OF VARCHAR2(256) INDEX BY BINARY_INTEGER;
```

例外

コンパイル警告を伴った DBMS_SQL を使用して、型、プロシージャ、ファンクションまたはパッケージを作成すると、ORA-24344 例外が発生しますが、プロシージャはそのまま作成されます。

BIND_VARIABLE および BIND_ARRAY プロシージャ

この 2 つのプロシージャは、文内の変数の名前に基づいて、カーソル内の指定の変数に指定の値または値のセットをバインドします。この変数が IN 変数、IN/OUT 変数、または IN コレクションである場合は、指定したバインド値が、変数タイプまたは配列タイプに対して有効である必要があります。OUT 変数のバインド値は無視されます。

SQL 文のバインド変数またはコレクションは、名前によって識別されます。バインド変数またはバインド配列に値をバインドする場合は、次の例に示すように、文中でバインド変数を識別する文字列の先頭にコロンを付ける必要があります。

```
SELECT emp_name FROM emp WHERE SAL > :X;
```

この例では、対応するバインド・コールは次のようになります。

```
BIND_VARIABLE(cursor_name, ':X', 3500);
```

または

```
BIND_VARIABLE (cursor_name, 'X', 3500);
```

構文

```
DBMS_SQL.BIND_VARIABLE (
    c                IN INTEGER,
    name             IN VARCHAR2,
    value            IN <datatype>)
```

<datatype> は、次のいずれかの型である必要があります。

```
NUMBER
DATE
VARCHAR2 CHARACTER SET ANY_CS
BLOB
CLOB CHARACTER SET ANY_CS
BFILE
UROWID
```

BIND_VARIABLE は、異なるデータ型を受け入れるためにオーバーロードされていることに注意してください。

関連項目：『Oracle8i アプリケーション開発者ガイド ラージ・オブジェクト』

プラグマ

```
pragma restrict_references(bind_variable,WNDS);
```

使用上の注意

BIND_VARIABLE では、次の構文もサポートされています。大カッコ [] は、BIND_VARIABLE ファンクションのオプション・パラメータを示します。

```
DBMS_SQL.BIND_VARIABLE (
  c              IN INTEGER,
  name           IN VARCHAR2,
  value          IN VARCHAR2 CHARACTER SET ANY_CS [,out_value_size IN INTEGER]);
```

CHAR、RAW および ROWID データをバインドするために、次のバリエーションを構文で使用できます。

```
DBMS_SQL.BIND_VARIABLE_CHAR (
  c              IN INTEGER,
  name           IN VARCHAR2,
  value          IN CHAR CHARACTER SET ANY_CS [,out_value_size IN INTEGER]);
```

```
DBMS_SQL.BIND_VARIABLE_RAW (
  c              IN INTEGER,
  name           IN VARCHAR2,
  value          IN RAW [,out_value_size IN INTEGER]);
```

```
DBMS_SQL.BIND_VARIABLE_ROWID (
  c              IN INTEGER,
  name           IN VARCHAR2,
  value          IN ROWID);
```

パラメータ

表 51-3 BIND_VARIABLE プロシージャのパラメータ

パラメータ	説明
c	値をバインドするカーソルの ID 番号。
name	文内の変数の名前。
value	カーソル内の変数にバインドする値。 IN 変数と IN/OUT 変数の場合、この値は、このパラメータで渡される値の型と同じ型です。
out_value_size	VARCHAR2、RAW、CHAR OUT または IN/OUT 変数の最大予測 OUT 値サイズ（単位はバイト）。 サイズの指定がない場合は、現行値の長さが使用されます。このパラメータは、value パラメータが初期化されていない場合、指定する必要があります。

バルク配列バインド

バルク選択、挿入、更新および削除は、多くのコールを1つにまとめることによって、アプリケーションのパフォーマンスを改善できます。DBMS_SQL パッケージによって、ユーザーは PL/SQL 表タイプを使用しながらデータの収集に対する処理を実行できます。

表項目は、バインドされていない同種のコレクションです。表項目は、持続記憶域では他のリレーショナル表に似ており、組込みの配列を持ちません。ただし、表項目が、（問合せまたは持続データのナビゲーション・アクセスのいずれかによって）作業領域に移されたり、あるいは PL/SQL の変数またはパラメータの値として作成されると、要素の値を取得して設定するために配列形式の構文で使用する添え書きが、その表項目の要素に与えられます。

これらの要素の添え書きは詳細である必要はなく、負数を含むあらゆる数値が使用できます。たとえば、表項目には、-10、2 および 7 の位置のみにある要素を含めることができます。

表項目が一時作業領域から持続記憶域に移されると、添え書きは格納されません。つまり、表項目は、持続記憶域内では順序が付いていません。

表は、バインド実行時に、PL/SQL バッファからローカルの DBMS_SQL バッファにコピーされ（すべてのスカラー型について同様）、ローカルの DBMS_SQL バッファから操作されます。したがって、バインド・コール後に表を変更した場合でも、その変更が実行方法に影響を与えることはありません。

スカラー型と LOB コレクション型

ローカル変数を次のいずれかの表項目型として宣言できます。これらの表項目型は、DBMS_SQL ではパブリック・タイプとして定義されています。

```
type Number_Table IS TABLE OF NUMBER          INDEX BY BINARY_INTEGER;
type Varchar2_Table IS TABLE OF VARCHAR2(2000) INDEX BY BINARY_INTEGER;
type Date_Table IS TABLE OF DATE              INDEX BY BINARY_INTEGER;
type Blob_Table IS TABLE OF BLOB              INDEX BY BINARY_INTEGER;
type Clob_Table IS TABLE OF CLOB              INDEX BY BINARY_INTEGER;
type Bfile_Table IS TABLE OF BFILE            INDEX BY BINARY_INTEGER;
type Urowid_Table IS TABLE OF UROWID         INDEX BY BINARY_INTEGER;
```

構文

```
DBMS_SQL.BIND_ARRAY (
    c                IN INTEGER,
    name             IN VARCHAR2,
    <table_variable> IN <datatype>
[,index1           IN INTEGER,
  index2           IN INTEGER]) );
```

<table_variable> とそれに対応する <datatype> は、次のいずれかの組合せになります。

<num_tab>	Number_Table
<vchr2_tab>	Varchar2_Table
<date_tab>	Date_Table
<blob_tab>	Blob_Table
<clob_tab>	Clob_Table
<bfile_tab>	Bfile_Table
<urowid_tab>	Urowid_Table

BIND_ARRAY プロシージャは、異なるデータ型を受け入れるためにオーバーロードされていることに注意してください。

パラメータ

表 51-4 BIND_ARRAY プロシージャのパラメータ

パラメータ	説明
c	値をバインドするカーソルの ID 番号
name	文内のコレクションの名前
table_variable	<datatype> として宣言されたローカル変数
index1	範囲の下限を示す表要素の索引
index2	範囲の上限を示す表要素の索引

使用上の注意

範囲をバインドするためには、範囲を指定する要素（タブ（index1）とタブ（index2））が表に含まれている必要がありますが、その範囲は詳細でなくても構いません。index1 には、index2 以下の値を指定してください。タブ（index1）とタブ（index2）の間にあるすべての要素がバインドして使用されます。

バインド・コールで索引を指定しない場合で、かつ文内の 2 つの異なるバインドが異なる数の要素を含んだ表を指定している場合、実際に使用される要素の数は、すべての表の最小値となります。これは索引を指定する場合にも当てはまります。つまり、すべての表に関する 2 つの索引の間では最小範囲が選択されます。

問合せ内のすべてのバインド変数が、配列バインドである必要はありません。一部は通常のバインドの場合があり、式の評価などでは、同じ値がコレクションの各要素に使用されま

す。

関連項目： コレクションのバインド方法の例は、51-37 ページの「[例 3、4 および 5: バルク DML](#)」を参照してください。

問合せの処理

動的 SQL を使用して問合せを処理する場合は、次のステップを実行する必要があります。

1. `DEFINE_COLUMN`、`DEFINE_COLUMN_LONG` または `DEFINE_ARRAY` をコールして、`SELECT` 文が戻す値を受け入れる変数を指定します。
2. `EXECUTE` をコールして、`SELECT` 文を実行します。
3. `FETCH_ROWS`（または `EXECUTE_AND_FETCH`）をコールして、問合せに合致した行を検索します。
4. 問合せに関して `FETCH_ROWS` コールが検索した列の値を判別するために、`COLUMN_VALUE` または `COLUMN_VALUE_LONG` をコールします。`PL/SQL` プロシージャへのコールを含んだ無名ブロックを使用した場合は、`VARIABLE_VALUE` をコールして、これらのプロシージャの出力変数に割り当てられた値を検索します。

DEFINE_COLUMN プロシージャ

このプロシージャは、指定のカーソルから選択する列を定義します。このプロシージャが使用できるのは、`SELECT` カーソルのみです。

定義されている列は、指定のカーソル内にある文の `SELECT` リスト内での相対位置によって識別されます。`COLUMN` 値の型によって、定義されている列の型が決まります。

構文

```
DBMS_SQL.DEFINE_COLUMN (  
    c                IN INTEGER,  
    position         IN INTEGER,  
    column           IN <datatype>)
```

<datatype> は、次のいずれかの型である必要があります。

```
NUMBER  
DATE  
BLOB  
CLOB CHARACTER SET ANY_CS  
BFILE  
UROWID
```

`DEFINE_COLUMN` は、異なるデータ型を受け入れるためにオーバーロードされていることに注意してください。

関連項目：『Oracle8i アプリケーション開発者ガイド ラージ・オブジェクト』

プラグマ

```
pragma restrict_references(define_column,RNDS,WNDS);
```

DEFINE_COLUMN プロシージャでは、次の構文もサポートされます。

```
DBMS_SQL.DEFINE_COLUMN (
    c                IN INTEGER,
    position         IN INTEGER,
    column           IN VARCHAR2 CHARACTER SET ANY_CS,
    column_size      IN INTEGER);
```

CHAR、RAW および ROWID データを持つ列の定義には、プロシージャ構文で次のバリエーションを使用できます。

```
DBMS_SQL.DEFINE_COLUMN_CHAR (
    c                IN INTEGER,
    position         IN INTEGER,
    column           IN CHAR CHARACTER SET ANY_CS,
    column_size      IN INTEGER);
```

```
DBMS_SQL.DEFINE_COLUMN_RAW (
    c                IN INTEGER,
    position         IN INTEGER,
    column           IN RAW,
    column_size      IN INTEGER);
```

```
DBMS_SQL.DEFINE_COLUMN_ROWID (
    c                IN INTEGER,
    position         IN INTEGER,
    column           IN ROWID);
```

パラメータ

表 51-5 DEFINE_COLUMN プロシージャのパラメータ

パラメータ	説明
c	選択対象に定義されている行のカーソルの ID 番号。
position	定義している行内にある列の相対位置。 文の最初の列は位置 1 です。
column	定義している列の値。 この値の型によって、定義している列の型が決まります。
column_size	VARCHAR2 型、CHAR 型、および RAW 型の列に対する列値の最大予測サイズ（単位はバイト）。

DEFINE_ARRAY プロシージャ

このプロシージャは、(FETCH_ROWS コールで) 行をフェッチする列に対してコレクションを定義します。このプロシージャによって、ユーザーは単一の SELECT 文から、行を一括してフェッチできます。1 回のフェッチ・コールで、PL/SQL の集計オブジェクトに多数の行をフェッチできます。

行をフェッチすると、それらの行は COLUMN_VALUE コールを実行するまで DBMS_SQL バッファにコピーされ、COLUMN_VALUE コールの実行時点で、引数として渡された表にコピーされます。

コレクション用のスカラー型と LOB 型

ローカル変数を、次のいずれかの表項目型として宣言し、DBMS_SQL を使用して、任意の行数をその中にフェッチできます。(これらの表項目型は、BIND_ARRAY プロシージャに指定できる型と同じです)。

type Number_Table	IS TABLE OF NUMBER	INDEX BY BINARY_INTEGER;
type Varchar2_Table	IS TABLE OF VARCHAR2(2000)	INDEX BY BINARY_INTEGER;
type Date_Table	IS TABLE OF DATE	INDEX BY BINARY_INTEGER;
type Blob_Table	IS TABLE OF BLOB	INDEX BY BINARY_INTEGER;
type Clob_Table	IS TABLE OF CLOB	INDEX BY BINARY_INTEGER;
type Bfile_Table	IS TABLE OF BFILE	INDEX BY BINARY_INTEGER;
type Urowid_Table	IS TABLE OF UROWID	INDEX BY BINARY_INTEGER;

構文

```
DBMS_SQL.DEFINE_ARRAY (  
    c           IN INTEGER,  
    position    IN INTEGER,  
    bf_tab      IN Bfile_Table,  
    cnt         IN INTEGER,  
    lower_bound IN INTEGER);
```

プラグマ

```
pragma restrict_references (define_array, RNDS, WNDS);
```

後続の `FETCH_ROWS` コールが `cnt` 行をフェッチします。`COLUMN_VALUE` がコールされると、これらの行は位置 `lower_bound`、`lower_bound+1`、`lower_bound+2` のように配置されます。行が送られてくる間、ユーザーは `FETCH_ROWS` コールまたは `COLUMN_VALUE` コールを継続して発行します。行は、`COLUMN_VALUE` コールに引数として指定した表内に蓄積されます。

パラメータ

表 51-6 DEFINE_ARRAY プロシージャのパラメータ

パラメータ	説明
c	配列をバインドするカーソルの ID 番号。
position	定義している配列内にある列の相対位置。 文の最初の列は位置 1 です。
bf_tab	フェッチされた行を格納する表。
cnt	フェッチする行数。
lower_bound	索引。

`cnt` にはゼロより大きい整数を指定してください。それ以外の値が指定されると、例外が発生します。`lower_bound` は、正の数、負の数またはゼロでも構いません。`DEFINE_ARRAY` コールが発行された問合せに、配列バインドを含めることはできません。

関連項目： コレクションの定義方法の例は、51-39 ページの「[例 6 および 7: 配列の定義](#)」を参照してください。

DEFINE_COLUMN_LONG プロシージャ

このプロシージャは、SELECT カーソルに対して LONG 列を定義します。定義されている列は、指定のカーソルの文の SELECT リスト内での相対位置によって識別されます。COLUMN 値の型によって、定義されている列の型が決まります。

構文

```
DBMS_SQL.DEFINE_COLUMN_LONG (  
    c                IN INTEGER,  
    position         IN INTEGER);
```

パラメータ

表 51-7 DEFINE_COLUMN_LONG プロシージャのパラメータ

パラメータ	説明
c	選択対象に定義されている行のカーソルの ID 番号。
position	定義している行内にある列の相対位置。 文の最初の列は位置 1 です。

EXECUTE ファンクション

このファンクションは、指定のカーソルを実行します。このファンクションはカーソルの ID 番号を受け入れて、処理された行数を戻します。戻り値は、DDL 文を含めて、INSERT 文、UPDATE 文、および DELETE 文に対してのみ有効で、戻り値が未定義の場合は無視されます。

構文

```
DBMS_SQL.EXECUTE (  
    c    IN INTEGER)  
RETURN INTEGER;
```

パラメータ

表 51-8 EXECUTE ファンクションのパラメータ

パラメータ	説明
c	実行するカーソルのカーソル ID 番号

EXECUTE_AND_FETCH ファンクション

このファンクションは、指定のカーソルを実行して行をフェッチします。このファンクションは、EXECUTE をコールしてから FETCH_ROWS をコールするのと同じ機能を提供します。ただし、リモート・データベースに対して使用する場合は、EXECUTE_AND_FETCH をコールした方がネットワークのラウンドトリップ数を低減できます。

EXECUTE_AND_FETCH ファンクションは、実際にフェッチされた行数を返します。

構文

```
DBMS_SQL.EXECUTE_AND_FETCH (
    c                IN INTEGER,
    exact            IN BOOLEAN DEFAULT FALSE)
RETURN INTEGER;
```

プラグマ

```
pragma restrict_references(execute_and_fetch,WNDS);
```

パラメータ

表 51-9 EXECUTE_AND_FETCH ファンクションのパラメータ

パラメータ	説明
c	実行してフェッチするカーソルのカーソル ID 番号。
exact	問合せで実際に一致する行数が 1 以外の場合は、TRUE に設定すると、例外が発生します。 注意: LONG 列に対して、exact パラメータを TRUE に設定するオプションはサポートされていません。 例外が発生しても、行はフェッチされ、使用可能です。

FETCH_ROWS ファンクション

このファンクションは、指定のカーソルから行をフェッチします。FETCH_ROWS は、フェッチする行が残っている限り、繰り返しコールできます。これらの行はバッファに取り出し、FETCH_ROWS への各コール後に、COLUMN_VALUE をコールして各列ごとに読み込む必要があります。

FETCH_ROWS ファンクションは、フェッチするカーソルの ID 番号を受け入れて、実際にフェッチされた行数を返します。

構文

```
DBMS_SQL.FETCH_ROWS (  
    c                IN INTEGER)  
    RETURN INTEGER;
```

パラメータ

表 51-10 FETCH_ROWS ファンクションのパラメータ

パラメータ	説明
c	ID 番号

プラグマ

```
pragma restrict_references(fetch_rows,WNDS);
```

COLUMN_VALUE プロシージャ

このプロシージャは、指定したカーソル内の指定の位置にあるカーソル要素の値を戻します。このプロシージャは、FETCH_ROWS をコールしてフェッチしたデータへのアクセスに使用されます。

構文

```
DBMS_SQL.COLUMN_VALUE (  
    c                IN  INTEGER,  
    position         IN  INTEGER,  
    value            OUT <datatype>  
    [,column_error   OUT NUMBER]  
    [,actual_length  OUT INTEGER]);
```

<datatype> は、次のいずれかの型である必要があります。

```
NUMBER  
DATE  
VARCHAR2  
CHARACTER SET ANY_CS  
BLOB  
CLOB  
CHARACTER SET ANY_CS  
BFILE  
UROWID
```

注意： 大カッコ [] は、オプション・パラメータを示します。

関連項目：『Oracle8i アプリケーション開発者ガイド ラージ・オブジェクト』

プラグマ

```
pragma restrict_references (column_value, RNDS, WNDS);
```

COLUMN_VALUE プロシージャでは、次の構文もサポートされます。

```
DBMS_SQL.COLUMN_VALUE(  
    c                IN  INTEGER,  
    position         IN  INTEGER,  
    <table_variable> IN  <datatype>);
```

<table_variable> とそれに対応する <datatype> は、次のいずれかの組合せが可能です。

```
<num_tab>      Number_Table  
<vchr2_tab>    Varchar2_Table  
<date_tab>     Date_Table  
<blob_tab>     Blob_Table  
<clob_tab>     Clob_Table  
<bfile_tab>    Bfile_Table  
<urowid_tab>   Urowid_Table
```

CHAR、RAW および ROWID データを含んだ列では、次のバリエーションを構文で使用できません。

```
DBMS_SQL.COLUMN_VALUE_CHAR (  
    c                IN  INTEGER,  
    position         IN  INTEGER,  
    value            OUT CHAR CHARACTER SET ANY_CS  
    [,column_error   OUT NUMBER]  
    [,actual_length  OUT INTEGER]);
```

```
DBMS_SQL.COLUMN_VALUE_RAW (  
    c                IN  INTEGER,  
    position         IN  INTEGER,  
    value            OUT RAW  
    [,column_error   OUT NUMBER]  
    [,actual_length  OUT INTEGER]);
```

```
DBMS_SQL.COLUMN_VALUE_ROWID (  
    c                IN  INTEGER,  
    position         IN  INTEGER,  
    value            OUT ROWID  
    [,column_error   OUT NUMBER]  
    [,actual_length  OUT INTEGER]);
```

パラメータ

表 51-11 COLUMN_VALUE プロシージャのパラメータ

パラメータ	説明
c	値をフェッチするカーソルの ID 番号。
position	カーソル内の列の相対位置。 文の最初の列は位置 1 です。
value	指定した列と行の値を返します。 指定した行番号がフェッチされた行数の合計より大きい場合は、エラー・メッセージが発生します。 この出力パラメータの型が、DEFINE_COLUMN へのコールで定義されている値の実際の型と異なる場合、例外エラー ORA-06562、inconsistent_type が発生します。
table_variable	<datatype> として宣言されたローカル変数。
column_error	指定した列値のエラー・コードを返します。
actual_length	指定した列内の値の（切捨て前の）実際の長さ。

例外：

指定した OUT パラメータの value が、実際の値の型と異なる場合は、inconsistent_type (ORA-06562) が発生します。この型は、DEFINE_COLUMN プロシージャをコールして列を定義したときに指定した型です。

COLUMN_VALUE_LONG プロシージャ

このプロシージャは、LONG 列の値の一部を取得します。

構文

```
DBMS_SQL.COLUMN_VALUE_LONG (  
    c                IN  INTEGER,  
    position         IN  INTEGER,  
    length           IN  INTEGER,  
    offset           IN  INTEGER,  
    value            OUT VARCHAR2,  
    value_length     OUT INTEGER);
```

プラグマ

```
pragma restrict_references (column_value_long, RNDS, WNDS);
```

パラメータ

表 51-12 COLUMN_VALUE_LONG プロシージャのパラメータ

パラメータ	説明
c	値を取得するカーソルのカーソル ID 番号
position	値を取得する列の位置
length	フェッチする LONG 値のバイト数
offset	フェッチを開始するための LONG フィールドへのオフセット
value	VARCHAR2 の列の値
value_length	値に実際に戻されるバイト数

VARIABLE_VALUE プロシージャ

このプロシージャは、指定のカーソルについて指定の変数の値を戻します。このプロシージャは、returning 句を使用して PL/SQL ブロックまたは DML 文内のバインド変数の値を戻すために使用されます。

構文

```
DEMS_SQL.VARIABLE_VALUE (  
  c                IN  INTEGER,  
  name             IN  VARCHAR2,  
  value            OUT <datatype>);
```

<datatype> は、次のいずれかの型である必要があります。

- NUMBER
- DATE
- VARCHAR2 CHARACTER SET ANY_CS
- BLOB
- CLOB CHARACTER SET ANY_CS
- BFILE
- UROWID

プラグマ

```
pragma restrict_references(variable_value,RNDS,WNDS);
```

VARIABLE_VALUE プロシージャでは、次の構文もサポートされます。

```
DBMS_SQL.VARIABLE_VALUE (
    c                IN   INTEGER,
    name             IN   VARCHAR2,
    <table_variable> IN   <datatype>);
```

<table_variable> とそれに対応する <datatype> は、次のいずれかの組合せが可能です。

```
<num_tab>      Number_Table
<vchr2_tab>    Varchar2_Table
<date_tab>     Date_Table
<blob_tab>     Blob_Table
<clob_tab>     Clob_Table
<bfile_tab>    Bfile_Table
<urowid_tab>   Urowid_Table
```

CHAR、RAW および ROWID データを含んだ変数では、次のバリエーションを構文で使用できます。

```
DBMS_SQL.VARIABLE_VALUE_CHAR (
    c                IN   INTEGER,
    name             IN   VARCHAR2,
    value            OUT CHAR CHARACTER SET ANY_CS);
```

```
DBMS_SQL.VARIABLE_VALUE_RAW (
    c                IN   INTEGER,
    name             IN   VARCHAR2,
    value            OUT RAW);
```

```
DBMS_SQL.VARIABLE_VALUE_ROWID (
    c                IN   INTEGER,
    name             IN   VARCHAR2,
    value            OUT ROWID);
```

パラメータ

表 51-13 VARIABLE_VALUE プロシージャのパラメータ

パラメータ	説明
c	値を取得するカーソルの ID 番号。
name	値を検索する変数名。
value	指定した位置の変数の値を戻します。 この出力パラメータの型が、BIND_VARIABLE へのコールで定義されている値の実際の型と異なる場合、例外エラー ORA-06562、inconsistent_type が発生します。

更新、挿入、削除の処理

動的 SQL を使用して INSERT、UPDATE または DELETE を処理する場合は、次のステップを実行する必要があります。

- 最初に、EXECUTE をコールして、INSERT 文、UPDATE 文または DELETE 文を実行します。
- 文に returning 句がある場合は、VARIABLE_VALUE をコールして出力変数に割り当てられた値を取り出します。

IS_OPEN ファンクション

このファンクションは、指定のカーソルが現在オープンしているかどうかをチェックします。

構文

```
DBMS_SQL.IS_OPEN (  
    c                IN INTEGER)  
RETURN BOOLEAN;
```

プラグマ

```
pragma restrict_references (is_open,RNDS,WNDS);
```


パラメータ

表 51-14 IS_OPEN ファンクションのパラメータ

パラメータ	説明
c	チェックするカーソルのカーソル ID 番号

戻り値

表 51-15 IS_OPEN ファンクションの戻り値

戻り値	説明
TRUE	指定のカーソルは、現在オープンしています。
FALSE	指定のカーソルは、現在オープンしていません。

DESCRIBE_COLUMNS プロシージャ

このプロシージャは、DBMS_SQL によってオープンして解析されたカーソルの列を記述します。

DESC_REC 型

DBMS_SQL パッケージでは、DESC_REC レコード型を次のように宣言します。

```
type desc_rec is record (  
    col_type          BINARY_INTEGER := 0,  
    col_max_len       BINARY_INTEGER := 0,  
    col_name          VARCHAR2(32)   := '',  
    col_name_len      BINARY_INTEGER := 0,  
    col_schema_name   VARCHAR2(32)   := '',  
    col_schema_name_len BINARY_INTEGER := 0,  
    col_precision     BINARY_INTEGER := 0,  
    col_scale         BINARY_INTEGER := 0,  
    col_charsetid     BINARY_INTEGER := 0,  
    col_charsetform   BINARY_INTEGER := 0,  
    col_null_ok       BOOLEAN        := TRUE);
```

パラメータ

表 51-16 DESC_REC 型のパラメータ

パラメータ	説明
col_type	記述される列の型
col_max_len	列の最大長
col_name	列の名前
col_name_len	列名の長さ
col_schema_name	列の型が定義されたスキーマの名前（オブジェクト型の場合）
col_schema_name_len	スキーマの長さ
col_precision	数値の場合は、列の精度
col_scale	数値の場合は、列の位取り
col_charsetid	列のキャラクタ・セットの識別子
col_charsetform	列のキャラクタ・セット・フォーム
col_null_ok	列で NULL が可能な場合は、TRUE

DESC_TAB 型

DESC_TAB 型は、DESC_REC レコードの PL/SQL 表です。

type desc_tab is table of desc_rec index by BINARY_INTEGER;

ローカル変数を PL/SQL 表型である DESC_TAB として宣言し、次に、DESCRIBE_COLUMNS プロシージャをコールして、各列の説明を表に含めることができます。その際、すべての列が記述され、単一の列のみを記述することはできません。

構文

```
DBMS_SQL.DESCRIBE_COLUMNS (
  c          IN  INTEGER,
  col_cnt    OUT INTEGER,
  desc_t     OUT DESC_TAB);
```

パラメータ

表 51-17 DBMS_SQL.DESCRIBE_COLUMNS プロシージャのパラメータ

パラメータ	説明
c	記述される列のカーソルの ID 番号。
col_cnt	問合せの選択リストにある列数。
desc_t	DESC_REC の表。各 DESC_REC が問合せ内の列を説明します。

関連項目： DESCRIBE_COLUMNS の使用方法は、51-41 ページの「[例 8: 列の記述](#)」を参照してください。

CLOSE_CURSOR プロシージャ

このファンクションは、指定のカーソルをクローズします。

構文

```
DBMS_SQL.CLOSE_CURSOR (  
    c      IN OUT INTEGER);
```

プラグマ

```
pragma restrict_references(close_cursor,RNDS,WNDS);
```

パラメータ

表 51-18 CLOSE_CURSOR プロシージャのパラメータ

パラメータ	モード	説明
c	IN	クローズするカーソルの ID 番号。
c	OUT	カーソルは NULL に設定されています。 CLOSE_CURSOR をコールした後、カーソルに割り当てられたメモリーは解放され、そのカーソルからはフェッチできなくなります。

エラーの位置

DBMS_SQL パッケージには、セッションで最後に参照されたカーソルの情報を取得するための追加ファンクションがいくつかあります。これらのファンクションが戻す値は、SQL 文の実行直後にのみ意味を持ちます。また、エラーを検出するファンクションは、特定の DBMS_SQL コール後にのみ意味を持ちます。たとえば、PARSE 直後に LAST_ERROR_POSITION をコールします。

LAST_ERROR_POSITION ファンクション

このファンクションは、エラーが発生した SQL 文テキスト内のバイト・オフセットを戻します。SQL 文内の最初の文字は、位置 0 にあります。

構文

```
DBMS_SQL.LAST_ERROR_POSITION  
    RETURN INTEGER;
```

パラメータ

なし

プラグマ

```
pragma restrict_references(last_error_position,RNDS,WNDS);
```

使用上の注意

このファンクションは、別の DBMS_SQL プロシージャまたはファンクションのコール前、かつ PARSE のコール後にコールしてください。

LAST_ROW_COUNT ファンクション

このファンクションは、フェッチされた累積行数を戻します。

構文

```
DBMS_SQL.LAST_ROW_COUNT  
    RETURN INTEGER;
```

パラメータ

なし

プラグマ

```
pragma restrict_references(last_row_count,RNDS,WNDS);
```

使用上の注意

このファンクションは、FETCH_ROWS コールまたは EXECUTE_AND_FETCH コール後にコールしてください。EXECUTE コール後にコールすると、戻される値はゼロです。

LAST_ROW_ID ファンクション

このファンクションは、処理された最後の行の ROWID を戻します。

構文

```
DBMS_SQL.LAST_ROW_ID  
RETURN ROWID;
```

パラメータ

なし

プラグマ

```
pragma restrict_references(last_row_id,RNDS,WNDS);
```

使用上の注意

このファンクションは、FETCH_ROWS コールまたは EXECUTE_AND_FETCH コール後にコールしてください。

LAST_SQL_FUNCTION_CODE ファンクション

このファンクションは、文の SQL ファンクション・コードを戻します。これらのコードは、『Oracle8i コール・インタフェース・プログラマーズ・ガイド』に一覧があります。

構文

```
DBMS_SQL.LAST_SQL_FUNCTION_CODE  
RETURN INTEGER;
```

パラメータ

なし

プラグマ

```
pragma restrict_references(last_sql_function_code,RNDS,WNDS);
```

使用上の注意

このファンクションは、SQL 文の実行直後にコールする必要があります。そうでない場合は、戻り値が未定義になります。

例

このセクションには、DBMS_SQL パッケージを使用するプロシージャの例が記述されています。

例 1: 次のプロシージャの例は、プロシージャを SQL 文に渡し、その SQL 文を解析して実行します。

```
CREATE OR REPLACE PROCEDURE exec (STRING IN varchar2) AS
    cursor_name INTEGER;
    ret INTEGER;
BEGIN
    cursor_name := DBMS_SQL.OPEN_CURSOR;
```

DDL 文は PARSE をコールして実行され、暗黙のコミットが実行されます。

```
    DBMS_SQL.PARSE(cursor_name, string, DBMS_SQL.native);
    ret := DBMS_SQL.EXECUTE(cursor_name);
    DBMS_SQL.CLOSE_CURSOR(cursor_name);
END;
```

このようなプロシージャを作成すると、次の操作を実行できます。

- コール側のプログラムによって、実行時に SQL 文を動的に生成できます。
- SQL 文は、DDL 文またはバインドなしの DML で構いません。

たとえば、このプロシージャの作成後に、次のコールを行うことができます。

```
exec('create table acct(c1 integer)');
```

次の例のように、このプロシージャはリモートでコールすることもできます。これによって、リモート DDL を実行できます。

```
exec@hq.com('CREATE TABLE acct(c1 INTEGER)');
```

例 2: 次のプロシージャの例は、ソースと宛先表の名前が渡され、ソース表から宛先表に行をコピーします。このプロシージャの例は、ソース表と宛先表にはいずれも次の列があることを前提としています。

```
id          of type NUMBER
name        of type VARCHAR2(30)
birthdate of type DATE
```

このプロシージャでは、動的 SQL を使用する必要は特にありませんが、ここでは、このパッケージの概念をわかりやすく説明しています。

```
CREATE OR REPLACE PROCEDURE copy (
    source      IN VARCHAR2,
    destination IN VARCHAR2) IS
    id_var      NUMBER;
    name_var    VARCHAR2(30);
    birthdate_var DATE;
    source_cursor INTEGER;
    destination_cursor INTEGER;
    ignore      INTEGER;
BEGIN

    -- Prepare a cursor to select from the source table:
    source_cursor := dbms_sql.open_cursor;
    DBMS_SQL.PARSE(source_cursor,
        'SELECT id, name, birthdate FROM ' || source,
        DBMS_SQL.native);
    DBMS_SQL.DEFINE_COLUMN(source_cursor, 1, id_var);
    DBMS_SQL.DEFINE_COLUMN(source_cursor, 2, name_var, 30);
    DBMS_SQL.DEFINE_COLUMN(source_cursor, 3, birthdate_var);
    ignore := DBMS_SQL.EXECUTE(source_cursor);

    -- Prepare a cursor to insert into the destination table:
    destination_cursor := DBMS_SQL.OPEN_CURSOR;
    DBMS_SQL.PARSE(destination_cursor,
        'INSERT INTO ' || destination ||
        ' VALUES (:id_bind, :name_bind, :birthdate_bind)',
        DBMS_SQL.native);

    -- Fetch a row from the source table and insert it into the destination table:
    LOOP
        IF DBMS_SQL.FETCH_ROWS(source_cursor) > 0 THEN
            -- get column values of the row
            DBMS_SQL.COLUMN_VALUE(source_cursor, 1, id_var);
            DBMS_SQL.COLUMN_VALUE(source_cursor, 2, name_var);
            DBMS_SQL.COLUMN_VALUE(source_cursor, 3, birthdate_var);

            -- Bind the row into the cursor that inserts into the destination table. You
```

```
-- could alter this example to require the use of dynamic SQL by inserting an
-- if condition before the bind.
    DBMS_SQL.BIND_VARIABLE(destination_cursor, ':id_bind', id_var);
    DBMS_SQL.BIND_VARIABLE(destination_cursor, ':name_bind', name_var);
    DBMS_SQL.BIND_VARIABLE(destination_cursor, ':birthdate_bind',
birthdate_var);
    ignore := DBMS_SQL.EXECUTE(destination_cursor);
    ELSE

-- No more rows to copy:
    EXIT;
    END IF;
END LOOP;

-- Commit and close all cursors:
COMMIT;
DBMS_SQL.CLOSE_CURSOR(source_cursor);
DBMS_SQL.CLOSE_CURSOR(destination_cursor);
EXCEPTION
    WHEN OTHERS THEN
        IF DBMS_SQL.IS_OPEN(source_cursor) THEN
            DBMS_SQL.CLOSE_CURSOR(source_cursor);
        END IF;
        IF DBMS_SQL.IS_OPEN(destination_cursor) THEN
            DBMS_SQL.CLOSE_CURSOR(destination_cursor);
        END IF;
        RAISE;
END;
/
```


例 3、4 および 5: バルク DML 次の一連の例では、DELETE、INSERT および UPDATE の各 SQL DML 文でのバルク配列バインド（表項目）の使用方法を示します。

たとえば、DELETE 文では、WHERE 句に配列をバインドし、配列内の要素ごとに文を実行できます。

```
declare
    stmt varchar2(200);
    dept_no_array dbms_sql.Number_Table;
    c number;
    dummy number;
begin
    dept_no_array(1) := 10; dept_no_array(2) := 20;
    dept_no_array(3) := 30; dept_no_array(4) := 40;
    dept_no_array(5) := 30; dept_no_array(6) := 40;
    stmt := 'delete from emp where deptno = :dept_array';
    c := dbms_sql.open_cursor;
    dbms_sql.parse(c, stmt, dbms_sql.native);
    dbms_sql.bind_array(c, ':dept_array', dept_no_array, 1, 4);
    dummy := dbms_sql.execute(c);
    dbms_sql.close_cursor(c);

    exception when others then
        if dbms_sql.is_open(c) then
            dbms_sql.close_cursor(c);
        end if;
        raise;
end;
/
```

前述の例では、1 から 4 までの要素のみが、bind_array コールで指定したとおりに使用されます。配列の各要素は、大量の従業員をデータベースから削除する可能性があります。

次に、バルク INSERT 文の例を示します。

```
declare
    stmt varchar2(200);
    empno_array dbms_sql.Number_Table;
    empname_array dbms_sql.Varchar2_Table;
    c number;
    dummy number;
begin
    for i in 0..9 loop
        empno_array(i) := 1000 + i;
        empname_array(i) := get_name(i);
    end loop;
    stmt := 'insert into emp values(:num_array, :name_array)';
    c := dbms_sql.open_cursor;
```

```
dbms_sql.parse(c, stmt, dbms_sql.native);
dbms_sql.bind_array(c, ':num_array', empno_array);
dbms_sql.bind_array(c, ':name_array', empname_array);
dummy := dbms_sql.execute(c);
dbms_sql.close_cursor(c);

exception when others then
    if dbms_sql.is_open(c) then
        dbms_sql.close_cursor(c);
    end if;
    raise;
end;
/
```

実行が開始されると、10 人の従業員はすべて表に挿入されます。
最後に、バルク UPDATE 文の例を示します。

```
declare
    stmt varchar2(200);
    emp_no_array dbms_sql.Number_Table;
    emp_addr_array dbms_sql.Varchar2_Table;
    c number;
    dummy number;
begin
    for i in 0..9 loop
        emp_no_array(i) := 1000 + i;
        emp_addr_array(i) := get_new_addr(i);
    end loop;
    stmt := 'update emp set ename = :name_array
            where empno = :num_array';
    c := dbms_sql.open_cursor;
    dbms_sql.parse(c, stmt, dbms_sql.native);
    dbms_sql.bind_array(c, ':num_array', empno_array);
    dbms_sql.bind_array(c, ':name_array', empname_array);
    dummy := dbms_sql.execute(c);
    dbms_sql.close_cursor(c);

    exception when others then
        if dbms_sql.is_open(c) then
            dbms_sql.close_cursor(c);
        end if;
        raise;
end;
/
```

EXECUTE がコールされると、全従業員のアドレスが一度に更新されます。2つのコレクションの処理は、いつも同時に進行します。WHERE 句で複数の行が戻される場合、全従業員は、その時点で addr_array が参照するアドレスを取得します。

例 6 および 7: 配列の定義 次の例では、DEFINE_ARRAY プロシージャの使用方法を示します。

```
declare
  c      number;
  d      number;
  n_tab  dbms_sql.Number_Table;
  indx   number := -10;
begin
  c := dbms_sql.open_cursor;
  dbms_sql.parse(c, 'select n from t order by 1', dbms_sql);

  dbms_sql.define_array(c, 1, n_tab, 10, indx);

  d := dbms_sql.execute(c);
  loop
    d := dbms_sql.fetch_rows(c);

    dbms_sql.column_value(c, 1, n_tab);

    exit when d != 10;
  end loop;

  dbms_sql.close_cursor(c);

  exception when others then
    if dbms_sql.is_open(c) then
      dbms_sql.close_cursor(c);
    end if;
    raise;
end;
/
```

前述の例では FETCH_ROWS をコールするたびに、DBMS_SQL バッファに保持されている 10 行がフェッチされます。COLUMN_VALUE コールが実行されると、それらの行は指定した PL/SQL 表（この場合は n_tab）の DEFINE 文で指定した -10 から -1 の位置に移動します。次に、2 番目のバッチがループ内でフェッチされ、行が 0 から 9 の位置に移動し、あとは同様に続きます。

各配列への現行の索引は、自動的にメンテナンスされます。この索引は、EXECUTE 時に "indx" に初期化され、COLUMN_VALUE がコールされるたびに更新されます。任意の時点で再実行した場合、各 DEFINE の現行の索引は "indx" に再初期化されます。

このようにして、問合せのすべての結果が表内にフェッチされます。FETCH_ROWS で 10 行をフェッチできない場合は、実際にフェッチされた行数を戻して（1 行もフェッチできなかった場合はゼロを戻します）、ループを終了します。

DEFINE_ARRAY プロシージャの別の使用例を次に示します。

次のように定義された MULTI_TAB 表を想定します。

```
create table multi_tab (num number,
                        dat1 date,
                        var varchar2(24),
                        dat2 date)
```

この表からすべてを選択して 4 つの PL/SQL 表に移動するには、次の簡単なプログラムを使用できます。

```
declare
  c      number;
  d      number;
  n_tab  dbms_sql.Number_Table;
  d_tab1 dbms_sql.Date_Table;
  v_tab  dbms_sql.Varchar2_Table;
  d_tab2 dbms_sql.Date_Table;
  indx number := 10;
begin

  c := dbms_sql.open_cursor;
  dbms_sql.parse(c, 'select * from multi_tab order by 1', dbms_sql);

  dbms_sql.define_array(c, 1, n_tab, 5, indx);
  dbms_sql.define_array(c, 2, d_tab1, 5, indx);
  dbms_sql.define_array(c, 3, v_tab, 5, indx);
  dbms_sql.define_array(c, 4, d_tab2, 5, indx);

  d := dbms_sql.execute(c);

  loop
    d := dbms_sql.fetch_rows(c);

    dbms_sql.column_value(c, 1, n_tab);
    dbms_sql.column_value(c, 2, d_tab1);
    dbms_sql.column_value(c, 3, v_tab);
    dbms_sql.column_value(c, 4, d_tab2);

    exit when d != 5;
  end loop;

  dbms_sql.close_cursor(c);
/*
```

これで、4つの表はあらゆる用途に使用できます。使用方法の1つは、'INSERT into SOME_T values (:a, :b, :c, :d)'などの問合せを使用して、行を他の表に移動するために BIND_ARRAY を使用できます。

```
*/

exception when others then
    if dbms_sql.is_open(c) then
        dbms_sql.close_cursor(c);
    end if;
    raise;
end;
/
```

例 8: 列の記述 この例は、記述する表に対して SELECT * による問合せを使用し、SQL*Plus の DESCRIBE コールのかわりに使用できます。

```
declare
    c number;
    d number;
    col_cnt integer;
    f boolean;
    rec_tab dbms_sql.desc_tab;
    col_num number;
    procedure print_rec(rec in dbms_sql.desc_rec) is
    begin
        dbms_output.new_line;
        dbms_output.put_line('col_type           = '
                               || rec.col_type);
        dbms_output.put_line('col_maxlen        = '
                               || rec.col_max_len);
        dbms_output.put_line('col_name         = '
                               || rec.col_name);
        dbms_output.put_line('col_name_len     = '
                               || rec.col_name_len);
        dbms_output.put_line('col_schema_name  = '
                               || rec.col_schema_name);
        dbms_output.put_line('col_schema_name_len = '
                               || rec.col_schema_name_len);
        dbms_output.put_line('col_precision    = '
                               || rec.col_precision);
        dbms_output.put_line('col_scale        = '
                               || rec.col_scale);
        dbms_output.put('col_null_ok          = ');
        if (rec.col_null_ok) then
            dbms_output.put_line('true');
        else
```

```
        dbms_output.put_line('false');
    end if;
end;
begin
    c := dbms_sql.open_cursor;

    dbms_sql.parse(c, 'select * from scott.bonus', dbms_sql);

    d := dbms_sql.execute(c);

    dbms_sql.describe_columns(c, col_cnt, rec_tab);

/*
 * Following loop could simply be for j in 1..col_cnt loop.
 * Here we are simply illustrating some of the PL/SQL table
 * features.
 */
    col_num := rec_tab.first;
    if (col_num is not null) then
        loop
            print_rec(rec_tab(col_num));
            col_num := rec_tab.next(col_num);
            exit when (col_num is null);
        end loop;
    end if;

    dbms_sql.close_cursor(c);
end;
/
```

例 9: RETURNING 句 RETURNING 句は、Oracle 8.0.3 では DML 文に追加されていました。この句を使用して、INSERT 文、UPDATE 文および DELETE 文は、式の値を戻すことができます。この値は、バインド変数に戻されます。

単一行が挿入、更新または削除される場合は、DBMS_SQL.BIND_VARIABLE を使用して、これらのアウトバインドをバインドします。複数の行が挿入、更新または削除された場合は、DBMS_SQL.BIND_ARRAY を使用します。これらのバインド変数の値を取得するには、DBMS_SQL.VARIABLE_VALUE をコールする必要があります。

注意： これは、DBMS_SQL 内でアウトバインドを使用した PL/SQL ブロックを実行した後で、DBMS_SQL.VARIABLE_VALUE をコールする必要があります。ことに似ています。

i) 単一行の挿入

```
create or replace procedure single_Row_insert
(c1 number, c2 number, r out number) is
c number;
n number;
begin
  c := dbms_sql.open_cursor;
  dbms_sql.parse(c, 'insert into tab values (:bnd1, :bnd2) ' ||
                  'returning c1*c2 into :bnd3', 2);
  dbms_sql.bind_variable(c, 'bnd1', c1);
  dbms_sql.bind_variable(c, 'bnd2', c2);
  dbms_sql.bind_variable(c, 'bnd3', r);
  n := dbms_sql.execute(c);
  dbms_sql.variable_value(c, 'bnd3', r); -- get value of outbind variable
  dbms_sql.close_cursor(c);
end;
/
```

ii) 単一行の更新

```
create or replace procedure single_Row_update
(c1 number, c2 number, r out number) is
c number;
n number;
begin
  c := dbms_sql.open_cursor;
  dbms_sql.parse(c, 'update tab set c1 = :bnd1, c2 = :bnd2 ' ||
                  'where rownum < 2' ||
                  'returning c1*c2 into :bnd3', 2);
  dbms_sql.bind_variable(c, 'bnd1', c1);
  dbms_sql.bind_variable(c, 'bnd2', c2);
  dbms_sql.bind_variable(c, 'bnd3', r);
```

```
n := dbms_sql.execute(c);
dbms_sql.variable_value(c, 'bnd3', r);-- get value of outbind variable
dbms_sql.close_cursor(c);
end;
/
```

iii) 単一行の削除

```
create or replace procedure single_Row_Delete
(c1 number, c2 number, r out number) is
c number;
n number;
begin
c := dbms_sql.open_cursor;
dbms_sql.parse(c, 'delete from tab ' ||
                'where rownum < 2 ' ||
                'returning c1*c2 into :bnd3', 2);
dbms_sql.bind_variable(c, 'bnd1', c1);
dbms_sql.bind_variable(c, 'bnd2', c2);
dbms_sql.bind_variable(c, 'bnd3', r);
n := dbms_sql.execute(c);
dbms_sql.variable_value(c, 'bnd3', r);-- get value of outbind variable
dbms_sql.close_cursor(c);
end;
/
```

iv) 複数行の挿入

```
create or replace procedure multi_Row_insert
(c1 dbms_sql.number_table, c2 dbms_sql.number_table,
r out dbms_sql.number_table) is
c number;
n number;
begin
c := dbms_sql.open_cursor;
dbms_sql.parse(c, 'insert into tab values (:bnd1, :bnd2) ' ||
                'returning c1*c2 into :bnd3', 2);
dbms_sql.bind_array(c, 'bnd1', c1);
dbms_sql.bind_array(c, 'bnd2', c2);
dbms_sql.bind_array(c, 'bnd3', r);
n := dbms_sql.execute(c);
dbms_sql.variable_value(c, 'bnd3', r);-- get value of outbind variable
dbms_sql.close_cursor(c);
end;
/
```


v) 複数行の更新

```

create or replace procedure multi_Row_update
(c1 number, c2 number, r out dbms_Sql.number_table) is
c number;
n number;
begin
  c := dbms_sql.open_cursor;
  dbms_sql.parse(c, 'update tab set c1 = :bnd1 where c2 = :bnd2 ' ||
    'returning c1*c2 into :bnd3', 2);
  dbms_sql.bind_variable(c, 'bnd1', c1);
  dbms_sql.bind_variable(c, 'bnd2', c2);
  dbms_sql.bind_array(c, 'bnd3', r);
  n := dbms_sql.execute(c);
  dbms_sql.variable_value(c, 'bnd3', r);-- get value of outbind variable
  dbms_sql.close_Cursor(c);
end;
/

```

注意： bnd1 と bnd2 は、同様に配列にできます。更新されたすべての行に
 対する式の値は、bnd3 に入れられます。bnd1 と bnd2 の各値について、ど
 の行が更新されたかを区別する方法はありません。

vi) 複数行の削除

```

create or replace procedure multi_row_delete
(c1 dbms_Sql.number_table,
r out dbms_sql.number_table) is
c number;
n number;
begin
  c := dbms_sql.open_cursor;
  dbms_sql.parse(c, 'delete from tab where c1 = :bnd1' ||
    'returning c1*c2 into :bnd2', 2);
  dbms_sql.bind_array(c, 'bnd1', c1);
  dbms_sql.bind_array(c, 'bnd2', r);
  n := dbms_sql.execute(c);
  dbms_sql.variable_value(c, 'bnd2', r);-- get value of outbind variable
  dbms_Sql.close_Cursor(c);
end;
/

```

vii) バルク PL/SQL でのアウトバインド

```
create or replace foo (n number, square out number) is
begin square := n * n; end;/

create or replace procedure bulk_plsql
  (n dbms_sql.number_Table, square out dbms_sql.number_Table) is
c number;
r number;
begin
  c := dbms_sql.open_cursor;
  dbms_sql.parse(c, 'begin foo(:bnd1, :bnd2); end;', 2);
  dbms_sql.bind_array(c, 'bnd1', n);
  dbms_sql.bind_Array(c, 'bnd2', square);
  r := dbms_sql.execute(c);
  dbms_sql.variable_Value(c, 'bnd2', square);
end;
/
```

注意： number_Table の DBMS_SQL.BIND_ARRAY は、数値を内部的にバインドします。文を実行する回数は、インバインド配列内の要素数によって決まります。

DBMS_STATS は、データベース・オブジェクトについて収集されたオプティマイザの統計情報を表示および変更するためのメカニズムを提供します。統計情報は、次の 2 つの場所に常駐させることができます。

1. ディクショナリ
2. この目的のためにユーザーのスキーマ内に作成された表

コストベースのオプティマイザに影響を与える統計情報は、ディクショナリに格納されている統計情報のみです。

また、このパッケージによって各種の統計情報を並列的に簡単に収集できます。パッケージは、次の 3 つの主なセクションに分割されています。

- [統計情報の設定または取得](#)
- [統計情報の転送](#)
- [オプティマイザ統計情報の収集](#)

DBMS_STATS の使用方法

ほとんどの DBMS_STATS プロシージャには、statown、stattab および statid の 3 つのパラメータが含まれています。これらのパラメータによって、ユーザーはオプティマイザに影響を与えない自分自身の表（ディクショナリ外）に統計情報を格納できます。したがって、ユーザーは統計情報のセットをメンテナンスおよび試用することができます。

stattab パラメータで、統計情報を保持する表の名前を指定します。この表は、(statown パラメータが指定されていない限り) 統計情報が収集されるオブジェクトと同じスキーマ内に常駐しているとみなされます。ユーザーは、異なる stattab 識別子で複数の表を作成し、統計情報セットを別々に保持できます。

さらに、statid パラメータを使用して 1 つの stattab 内で複数の統計情報セットをメンテナンスできるため、ユーザーのスキーマが混乱するのを回避できます。

すべての SET または GET プロシージャについて、stattab が準備されていない場合（つまり NULL の場合）、操作はディクショナリにある統計情報に対して直接行われます。したがって、ディクショナリを直接変更する場合は、統計表の作成は不要です。ただし、stattab が NULL でない場合、SET または GET 操作は、指定したユーザー統計表に対して行われ、ディクショナリに対しては行われません。

型

最小 / 最大値とヒストグラム終点の型は、次のとおりです。

```
TYPE numarray IS VARRAY(256) OF NUMBER;
TYPE datearray IS VARRAY(256) OF DATE;
TYPE chararray IS VARRAY(256) OF VARCHAR2(4000);
TYPE rawarray IS VARRAY(256) OF RAW(2000);
```

```
type StatRec is record (
    epc      NUMBER,
    minval   RAW(2000),
    maxval   RAW(2000),
    bkvals   NUMARRAY,
    novals   NUMARRAY);
```

失効した表のリストの型は、次のとおりです。

```
type ObjectElem is record (
    ownname   VARCHAR2(30),      -- owner
    objtype   VARCHAR2(6),      -- 'TABLE' or 'INDEX'
    objname   VARCHAR2(30),      -- table/index
    partname   VARCHAR2(30),      -- partition
    subpartname VARCHAR2(30),      -- subpartition
    confidence NUMBER);          -- not used
type ObjectTab is TABLE of ObjectElem;
```

サブプログラムの要約

表 52-1 DBMS_STATS パッケージのサブプログラム

サブプログラム	説明
52-5 ページの PREPARE_COLUMN_VALUES プロシ ジャ	ユーザー指定の最小値、最大値およびヒストグラム終点のデータ型固有値を、 <code>SET_COLUMN_STATS</code> を介して将来格納するために Oracle の内部表記に変換します。
52-8 ページの SET_COLUMN_STATS プロシージャ	列に関連する情報を設定します。
52-9 ページの SET_INDEX_STATS プロシージャ	索引に関連する情報を設定します。
52-11 ページの SET_TABLE_STATS プロシージャ	表に関連する情報を設定します。
52-12 ページの CONVERT_RAW_VALUE プロシージャ	最大値または最小値の内部表記を、データ型固有値に変換します。
52-13 ページの GET_COLUMN_STATS プロシージャ	列に関連するすべての情報を取得します。
52-15 ページの GET_INDEX_STATS プロシージャ	索引に関連するすべての情報を取得します。
52-16 ページの GET_TABLE_STATS プロシージャ	表に関連するすべての情報を取得します。
52-17 ページの DELETE_COLUMN_STATS プロシージャ	列に関連する統計情報を削除します。
52-18 ページの DELETE_INDEX_STATS プロシージャ	索引に関連する統計情報を削除します。
52-19 ページの DELETE_TABLE_STATS プロシージャ	表に関連する統計情報を削除します。
52-21 ページの DELETE_SCHEMA_STATS プロシージャ	スキーマに関連する統計情報を削除します。
52-21 ページの DELETE_ DATABASE_STATS プロシージャ	データベース全体に関する統計情報を削除します。
52-23 ページの CREATE_STAT_TABLE プロシージャ	統計情報を保持できる <code>ownname</code> のスキーマに <code>stattab</code> の名前で表を作成します。
52-24 ページの DROP_STAT_TABLE プロシージャ	<code>CREATE_STAT_TABLE</code> で作成したユーザー統計表を削除します。

表 52-1 DBMS_STATS パッケージのサブプログラム

サブプログラム	説明
52-24 ページの EXPORT_COLUMN_STATS プロシージャ	特定の列に関する統計情報を取り出し、stattab で識別されるユーザー統計表に格納します。
52-25 ページの EXPORT_INDEX_STATS プロシージャ	特定の索引に関する統計情報を取り出し、stattab で識別されるユーザー統計表に格納します。
52-26 ページの EXPORT_TABLE_STATS プロシージャ	特定の表に関する統計情報を取り出し、ユーザー統計表に格納します。
52-27 ページの EXPORT_SCHEMA_STATS プロシージャ	ownname で識別されるスキーマ内のすべてのオブジェクトに関する統計情報を取り出し、stattab で識別されるユーザー統計表に格納します。
52-28 ページの EXPORT_DATABASE_STATS プロシージャ	データベース内のすべてのオブジェクトに関する統計情報を取り出し、statown.stattab で識別されるユーザー統計表に格納します。
52-29 ページの IMPORT_COLUMN_STATS プロシージャ	stattab で識別されるユーザー統計表から特定の列に関する統計情報を取り出し、ディクショナリに格納します。
52-30 ページの IMPORT_INDEX_STATS プロシージャ	stattab で識別されるユーザー統計表から特定の索引に関する統計情報を取り出し、ディクショナリに格納します。
52-31 ページの IMPORT_TABLE_STATS プロシージャ	stattab で識別されるユーザー統計表から特定の表に関する統計情報を取り出し、ディクショナリに格納します。
52-32 ページの IMPORT_SCHEMA_STATS プロシージャ	ownname で識別されるスキーマ内のすべてのオブジェクトに関する統計情報をユーザー統計表から取り出し、ディクショナリに格納します。
52-32 ページの IMPORT_DATABASE_STATS プロシージャ	データベース内のすべてのオブジェクトに関する統計情報をユーザー統計表から取り出し、ディクショナリに格納します。
52-34 ページの GATHER_INDEX_STATS プロシージャ	索引の統計情報を収集します。
52-35 ページの GATHER_TABLE_STATS プロシージャ	表と列（および索引）の統計情報を収集します。
52-37 ページの GATHER_SCHEMA_STATS プロシージャ	スキーマ内のすべてのオブジェクトに関する統計情報を収集します。
52-39 ページの GATHER_DATABASE_STATS プロシージャ	データベース内のすべてのオブジェクトに関する統計情報を収集します。
52-42 ページの GENERATE_STATS プロシージャ	関連するオブジェクトに関して以前に収集した統計情報から、オブジェクトの統計情報を生成します。

統計情報の設定または取得

次のプロシージャによって、列、索引および表に関連する個別の統計情報を格納したり、取り出すことができます。

```
PREPARE_COLUMN_VALUES  
SET_COLUMN_STATS  
SET_INDEX_STATS  
SET_TABLE_STATS
```

```
CONVERT_RAW_VALUE  
GET_COLUMN_STATS  
GET_INDEX_STATS  
GET_TABLE_STATS
```

```
DELETE_COLUMN_STATS  
DELETE_INDEX_STATS  
DELETE_TABLE_STATS  
DELETE_SCHEMA_STATS  
DELETE_DATABASE_STATS
```

PREPARE_COLUMN_VALUES プロシージャ

このプロシージャは、ユーザー指定の最小値、最大値およびヒストグラム終点のデータ型固有値を、SET_COLUMN_STATS を介して将来格納するために Oracle の内部表記に変換します。

構文

```
DBMS_STATS.PREPARE_COLUMN_VALUES (  
    srec      IN OUT StatRec,  
    charvals   CHARARRAY);  
  
DBMS_STATS.PREPARE_COLUMN_VALUES (  
    srec      IN OUT StatRec,  
    datevals   DATEARRAY);  
  
DBMS_STATS.PREPARE_COLUMN_VALUES (  
    srec      IN OUT StatRec,  
    numvals    NUMARRAY);  
  
DBMS_STATS.PREPARE_COLUMN_VALUES (  
    srec      IN OUT StatRec,  
    rawvals    RAWARRAY);  
  
DBMS_STATS.PREPARE_COLUMN_VALUES_NVARCHAR (  
    srec      IN OUT StatRec,
```

```
nvmin      NVARCHAR2,  
nvmax      NVARCHAR2);
```

```
DBMS_STATS.PREPARE_COLUMN_VALUES_ROWID (  
  srec  IN OUT StatRec,  
  rwmin      ROWID,  
  rwmax      ROWID);
```

プラグマ

```
pragma restrict_references(prepare_column_values, WNDS, RNDS, WNPS, RNPS);  
pragma restrict_references(prepare_column_values_nvarchar, WNDS, RNDS, WNPS, RNPS);  
pragma restrict_references(prepare_column_values_rowid, WNDS, RNDS, WNPS, RNPS);
```

パラメータ

表 52-2 PREPARE_COLUMN_VALUES プロシージャのパラメータ

パラメータ	説明
srec.epc	<p>charvals、datevals、numvals または rawvals で指定される値の数。この値は 2 ～ 256 の範囲で指定し、ヒストグラム情報を認めないプロシージャ（nvarchar と rowid）では 2 を設定する必要があります。</p> <p>最初に対応する配列のエントリは列の最小値を含み、最後のエントリは最大値を含みます。2 つを超えるエントリがある場合、その他のエントリは、残りの高さ調整ヒストグラムまたは頻度ヒストグラムのエンドポイント値（小さい値から大きい値に順序立てられた中間値を持つ）を含みます。この値は圧縮のために調整できるため、戻り値は、SET_COLUMN_STATS へのコールに対して現状のままになります。</p>
srec.bkvals	<p>頻度分布が必要な場合、この配列には、charvals、datevals、numvals または rawvals で指定されている各個別値の発生回数が含まれています。そうでない場合、この配列は単なるアウトプット・パラメータであり、このプロシージャのコール時には NULL が設定されている必要があります。</p>

データ型固有の入力パラメータ（次の中から 1 つを選択）。

charvals	列型が文字ベースの場合の値の配列。各文字列の最初の 32 バイトまでが使用されます。配列のエントリ数は、2 ～ 256 の範囲である必要があります。
datevals	列型が日付ベースの場合の値の配列。
numvals	列型が数値ベースの場合の値の配列。
rawvals	列型が RAW の場合の値の配列。各文字列の最初の 32 バイトまでが使用されます。
nvmin, nvmax	列型が各国文字キャラクタ・セット（NLS）の場合の最大値と最小値。この型の列について、ヒストグラム情報は提供できません。
rwmin, rwmax	列型が rowid の場合の最大値と最小値。この型の列について、ヒストグラム情報は提供できません。

アウトプット・パラメータ

表 52-3 PREPARE_COLUMN_VALUES プロシージャのアウトプット・パラメータ

パラメータ	説明
srec.minval	SET_COLUMN_STATS へのコールでの使用に適した最小値の内部表記
srec.maxval	SET_COLUMN_STATS へのコールでの使用に適した最大値の内部表記
srec.bkvals	SET_COLUMN_STATS へのコールでの使用に適した配列
srec.novals	SET_COLUMN_STATS へのコールでの使用に適した配列

例外

ORA-20001: 入力値が無効か、または矛盾しています。

SET_COLUMN_STATS プロシージャ

このプロシージャは、列に関連する情報を設定します。

構文

```
DBMS_STATS.SET_COLUMN_STATS (  
    ownname  VARCHAR2,  
    tabname  VARCHAR2,  
    colname  VARCHAR2,  
    partname VARCHAR2 DEFAULT NULL,  
    stattab  VARCHAR2 DEFAULT NULL,  
    statid   VARCHAR2 DEFAULT NULL,  
    distcnt  NUMBER   DEFAULT NULL,  
    density  NUMBER   DEFAULT NULL,  
    nullcnt  NUMBER   DEFAULT NULL,  
    srec     StatRec  DEFAULT NULL,  
    avgclen  NUMBER   DEFAULT NULL,  
    flags    NUMBER   DEFAULT NULL,  
    statown  VARCHAR2 DEFAULT NULL);
```

パラメータ

表 52-4 SET_COLUMN_STATS プロシージャのパラメータ

パラメータ	説明
ownname	スキーマの名前。
tabname	列が所属している表の名前。
colname	列の名前。
partname	統計情報を格納する表パーティションの名前。表がパーティション化されていて、partname が NULL の場合、統計情報はグローバルな表レベルで格納されます。
stattab	統計情報の格納場所を示すユーザー統計表の識別子。stattab が NULL の場合、統計情報はディクショナリ内に直接格納されます。
statid	stattab 内の統計情報を関連付ける識別子（オプション）（stattab が NULL でない場合のみ該当します）。
distcnt	個別値の数。
density	列密度。この値が NULL で、distcnt が NULL でない場合、密度は distcnt から導出されます。
nullcnt	NULL の数。

表 52-4 SET_COLUMN_STATS プロシージャのパラメータ

パラメータ	説明
srec	PREPARE_COLUMN_VALUES または GET_COLUMN_STATS へのコールで入力された StatRec 構造
avgclen	列の平均長（単位はバイト）
flags	Oracle 内部で使用（NULL のままにします）
statown	stattab を含んだスキーマ（ownname と異なる場合）

例外

ORA-20000: オブジェクトが存在しないか、または権限が不十分です。

ORA-20001: 入力値が無効か、または矛盾しています。

SET_INDEX_STATS プロシージャ

このプロシージャは、索引に関連する情報を設定します。

構文

```
DBMS_STATS.SET_INDEX_STATS (  
  ownname  VARCHAR2,  
  indname  VARCHAR2,  
  partname VARCHAR2  DEFAULT NULL,  
  stattab  VARCHAR2  DEFAULT NULL,  
  statid   VARCHAR2  DEFAULT NULL,  
  numrows  NUMBER    DEFAULT NULL,  
  numlblks NUMBER    DEFAULT NULL,  
  numdist  NUMBER    DEFAULT NULL,  
  avglblk  NUMBER    DEFAULT NULL,  
  avgdblk  NUMBER    DEFAULT NULL,  
  clstfct  NUMBER    DEFAULT NULL,  
  indlevel NUMBER    DEFAULT NULL,  
  flags    NUMBER    DEFAULT NULL,  
  statown  VARCHAR2  DEFAULT NULL);
```

パラメータ

表 52-5 SET_INDEX_STATS プロシージャのパラメータ

パラメータ	説明
ownname	スキーマの名前。
indname	索引名。
partname	統計情報を格納する索引パーティションの名前。索引がパーティション化されていて、partname が NULL の場合、統計情報はグローバルな索引レベルで格納されます。
stattab	統計情報の格納場所を示すユーザー統計表の識別子。stattab が NULL の場合、統計情報はディクショナリ内に直接格納されます。
statid	stattab 内の統計情報を関連付ける識別子（オプション）（stattab が NULL でない場合のみ該当します）。
numrows	索引（パーティション）内の行数。
numlblks	索引（パーティション）内のリーフ・ブロックの数。
numdist	索引（パーティション）内の個別キーの数。
avglblk	この索引（パーティション）について各個別キーが出現するリーフ・ブロックの平均整数値。この値が提供されない場合、この値は numlblks と numdist から導出されます。
avgdblk	この索引（パーティション）について個別キーが指す表内のデータ・ブロックの平均整数値。この値が提供されない場合、この値は clstfct と numdist から導出されます。
clstfct	user_indexes ビューの clustering_factor 列の説明を参照してください。
indlevel	索引（パーティション）の高さ。
flags	Oracle 内部で使用（NULL のままにします）。
statown	stattab を含んだスキーマ（ownname と異なる場合）。

例外

- ORA-20000: オブジェクトが存在しないか、または権限が不十分です。
- ORA-20001: 入力値が無効です。

SET_TABLE_STATS プロシージャ

このプロシージャは、表に関連する情報を設定します。

構文

```
DBMS_STATS.SET_TABLE_STATS (  
    ownname  VARCHAR2,  
    tabname  VARCHAR2,  
    partname VARCHAR2 DEFAULT NULL,  
    stattab  VARCHAR2 DEFAULT NULL,  
    statid   VARCHAR2 DEFAULT NULL,  
    numrows  NUMBER   DEFAULT NULL,  
    numblks  NUMBER   DEFAULT NULL,  
    avgrlen  NUMBER   DEFAULT NULL,  
    flags    NUMBER   DEFAULT NULL,  
    statown  VARCHAR2 DEFAULT NULL);
```

パラメータ

表 52-6 SET_TABLE_STATS プロシージャのパラメータ

パラメータ	説明
ownname	スキーマの名前。
tabname	表の名前。
partname	統計情報を格納する表パーティションの名前。表がパーティション化されていて、partname が NULL の場合、統計情報はグローバルな表レベルで格納されます。
stattab	統計情報の格納場所を示すユーザー統計表の識別子。stattab が NULL の場合、統計情報はディクショナリ内に直接格納されます。
statid	stattab 内の統計情報を関連付ける識別子（オプション）（stattab が NULL でない場合のみ該当します）。
numrows	表（パーティション）内の行数。
numblks	表（パーティション）が占有するブロックの数。
avgrlen	表（パーティション）の行の平均長。
flags	Oracle 内部で使用（NULL のままにします）。
statown	stattab を含んだスキーマ（ownname と異なる場合）。

例外

ORA-20000: オブジェクトが存在しないか、または権限が不十分です。

ORA-20001: 入力値が無効です。

CONVERT_RAW_VALUE プロシージャ

このプロシージャは、最大値または最小値の内部表記をデータ型固有値に変換します。
GET_COLUMN_STATS または PREPARE_COLUMN_VALUES で入力された StatRec 構造の minval フィールドと maxval フィールドの値が、有効な入力値です。

構文

```
DBMS_STATS.CONVERT_RAW_VALUE (  
    rawval      RAW,  
    resval OUT VARCHAR2);
```

```
DBMS_STATS.CONVERT_RAW_VALUE (  
    rawval      RAW,  
    resval OUT DATE);
```

```
DBMS_STATS.CONVERT_RAW_VALUE (  
    rawval      RAW,  
    resval OUT NUMBER);
```

```
DBMS_STATS.CONVERT_RAW_VALUE_NVARCHAR (  
    rawval      RAW,  
    resval OUT NVARCHAR2);
```

```
DBMS_STATS.CONVERT_RAW_VALUE_ROWID (  
    rawval      RAW,  
    resval OUT ROWID);
```

プラグマ

```
pragma restrict_references(convert_raw_value, WNDS, RNDS, WNPS, RNPS);  
pragma restrict_references(convert_raw_value_nvarchar, WNDS, RNDS, WNPS, RNPS);  
pragma restrict_references(convert_raw_value_rowid, WNDS, RNDS, WNPS, RNPS);
```

パラメータ

表 52-7 CONVERT_RAW_VALUE プロシージャのパラメータ

パラメータ	説明
rawval	列最大または列最小のデータ型固有アウトプット・パラメータの内部表記
resval	変換済の型固有値

例外

なし

GET_COLUMN_STATS プロシージャ

このプロシージャは、列に関連するすべての情報を取得します。

構文

```
DBMS_STATS.GET_COLUMN_STATS (  
  ownname      VARCHAR2,  
  tabname      VARCHAR2,  
  colname      VARCHAR2,  
  partname     VARCHAR2 DEFAULT NULL,  
  stattab      VARCHAR2 DEFAULT NULL,  
  statid       VARCHAR2 DEFAULT NULL,  
  distcnt OUT NUMBER,  
  density OUT NUMBER,  
  nullcnt OUT NUMBER,  
  srec      OUT StatRec,  
  avgclen OUT NUMBER,  
  statown   VARCHAR2 DEFAULT NULL);
```

パラメータ

表 52-8 GET_COLUMN_STATS プロシージャのパラメータ

パラメータ	説明
ownname	スキーマの名前。
tabname	列が所属している表の名前。
colname	列の名前。
partname	統計情報を取得する表パーティションの名前。表がパーティション化されていて、partname が NULL の場合、統計情報はグローバルな表レベルで取り出されます。
stattab	統計情報を取り出す場所を示すユーザー統計表の識別子。stattab が NULL の場合、統計情報はディクショナリから直接取り出されます。
statid	stattab 内の統計情報を関連付ける識別子（オプション）（stattab が NULL でない場合のみ該当します）。
distcnt	個別値の数。
density	列密度。
nullcnt	NULL の数。
srec	列最大、列最小およびヒストグラム値の内部表記を保持する構造。
avgclen	列の平均長（単位はバイト）。
statown	stattab を含んだスキーマ（ownname と異なる場合）。

例外

ORA-20000: オブジェクトが存在しないか権限が不十分、または要求したオブジェクトの統計情報が格納されていません。

GET_INDEX_STATS プロシージャ

このプロシージャは、索引に関連するすべての情報を取得します。

構文

```
DBMS_STATS.GET_INDEX_STATS (  
    ownname      VARCHAR2,  
    indname      VARCHAR2,  
    partname     VARCHAR2 DEFAULT NULL,  
    stattab      VARCHAR2 DEFAULT NULL,  
    statid       VARCHAR2 DEFAULT NULL,  
    numrows      OUT NUMBER,  
    numlblks     OUT NUMBER,  
    numdist      OUT NUMBER,  
    avglblk      OUT NUMBER,  
    avgdblk      OUT NUMBER,  
    clstfct      OUT NUMBER,  
    indlevel     OUT NUMBER,  
    statown      VARCHAR2 DEFAULT NULL);
```

パラメータ

表 52-9 GET_INDEX_STATS プロシージャのパラメータ

パラメータ	説明
ownname	スキーマの名前。
indname	索引名。
partname	統計情報を取得する索引パーティションの名前。索引がパーティション化されていて、partname が NULL の場合、統計情報はグローバルな索引レベルで取り出されます。
stattab	統計情報を取り出す場所を示すユーザー統計表の識別子。stattab が NULL の場合、統計情報はディクショナリから直接取り出されます。
statid	stattab 内の統計情報を関連付ける識別子（オプション）（stattab が NULL でない場合のみ該当します）。
numrows	索引（パーティション）内の行数。
numlblks	索引（パーティション）内のリーフ・ブロックの数。
numdist	索引（パーティション）内の個別キーの数。
avglblk	この索引（パーティション）について各個別キーが出現するリーフ・ブロックの平均整数値。

表 52-9 GET_INDEX_STATS プロシージャのパラメータ

パラメータ	説明
avgdblk	この索引（パーティション）について個別キーが指す表内のデータ・ブロックの平均整数値
clstfct	索引（パーティション）のクラスタ化要素
indlevel	索引（パーティション）の高さ
statown	stattab を含んだスキーマ（ownname と異なる場合）

例外

ORA-20000: オブジェクトが存在しないか権限が不十分、または要求したオブジェクトの統計情報が格納されていません。

GET_TABLE_STATS プロシージャ

このプロシージャは、表に関連するすべての情報を取得します。

構文

```
DEMS_STATS.GET_TABLE_STATS (  
  ownname      VARCHAR2,  
  tabname      VARCHAR2,  
  partname     VARCHAR2 DEFAULT NULL,  
  stattab      VARCHAR2 DEFAULT NULL,  
  statid       VARCHAR2 DEFAULT NULL,  
  numrows OUT  NUMBER,  
  numblks OUT  NUMBER,  
  avgrlen OUT  NUMBER,  
  statown      VARCHAR2 DEFAULT NULL);
```

パラメータ

表 52-10 GET_TABLE_STATS プロシージャのパラメータ

パラメータ	説明
ownname	スキーマの名前。
tabname	列が所属している表の名前。
partname	統計情報を取得する表パーティションの名前。表がパーティション化されていて、partname が NULL の場合、統計情報はグローバルな表レベルで取り出されます。

表 52-10 GET_TABLE_STATS プロシージャのパラメータ

パラメータ	説明
stattab	統計情報を取り出す場所を示すユーザー統計表の識別子。 stattab が NULL の場合、統計情報はディクショナリから直接取り出されます。
statid	stattab 内の統計情報を関連付ける識別子（オプション） (stattab が NULL でない場合のみ該当します)。
numrows	表（パーティション）内の行数。
numblks	表（パーティション）が占有するブロックの数。
avgrlen	表（パーティション）の行の平均長。
statown	stattab を含んだスキーマ（ownname と異なる場合）。

例外

ORA-20000: オブジェクトが存在しないか権限が不十分、または要求されたオブジェクトの統計情報が格納されていません。

DELETE_COLUMN_STATS プロシージャ

このプロシージャは、列に関連する統計情報を削除します。

構文

```
DBMS_STATS.DELETE_COLUMN_STATS (
    ownname      VARCHAR2,
    tabname      VARCHAR2,
    colname      VARCHAR2,
    partname     VARCHAR2 DEFAULT NULL,
    stattab      VARCHAR2 DEFAULT NULL,
    statid       VARCHAR2 DEFAULT NULL,
    cascade_parts BOOLEAN  DEFAULT TRUE,
    statown      VARCHAR2 DEFAULT NULL);
```

パラメータ

表 52-11 DELETE_COLUMN_STATS プロシージャのパラメータ

パラメータ	説明
ownname	スキーマの名前。
tabname	列が所属している表の名前。

表 52-11 DELETE_COLUMN_STATS プロシージャのパラメータ

パラメータ	説明
colname	列の名前。
partname	統計情報を削除する表パーティションの名前。表がパーティション化されていて、partname が NULL の場合、グローバルな列統計情報が削除されます。
stattab	統計情報を削除する場所を示すユーザー統計表の識別子。stattab が NULL の場合、統計情報はディクショナリから直接削除されます。
statid	stattab 内の統計情報を関連付ける識別子（オプション）（stattab が NULL でない場合のみ該当します）。
cascade_parts	表がパーティション化されていて、partname が NULL の場合、このパラメータを TRUE に設定すると、基礎となるすべてのパーティションについても列の統計情報が削除されます。
statown	stattab を含んだスキーマ（ownname と異なる場合）。

例外

ORA-20000: オブジェクトが存在しないか、または権限が不十分です。

DELETE_INDEX_STATS プロシージャ

このプロシージャは、索引に関連する統計情報を削除します。

構文

```
DBMS_STATS.DELETE_INDEX_STATS (  
  ownname      VARCHAR2,  
  indname      VARCHAR2,  
  partname     VARCHAR2 DEFAULT NULL,  
  stattab      VARCHAR2 DEFAULT NULL,  
  statid       VARCHAR2 DEFAULT NULL,  
  cascade_parts BOOLEAN  DEFAULT TRUE,  
  statown      VARCHAR2 DEFAULT NULL);
```

パラメータ

表 52-12 DELETE_INDEX_STATS プロシージャのパラメータ

パラメータ	説明
ownname	スキーマの名前。
indname	索引名。
partname	統計情報を削除する索引パーティションの名前。索引がパーティション化されていて、partname が NULL の場合、索引統計情報はグローバル・レベルで削除されます。
stattab	統計情報を削除する場所を示すユーザー統計表の識別子。stattab が NULL の場合、統計情報はディクショナリから直接削除されます。
statid	stattab 内の統計情報を関連付ける識別子（オプション）（stattab が NULL でない場合のみ該当します）。
cascade_parts	索引がパーティション化されていて、partname が NULL の場合、このパラメータを TRUE に設定すると、基礎となるすべてのパーティションについても索引の統計情報が削除されます。
statown	stattab を含んだスキーマ（ownname と異なる場合）。

例外

ORA-20000: オブジェクトが存在しないか、または権限が不十分です。

DELETE_TABLE_STATS プロシージャ

このプロシージャは、表に関連する統計情報を削除します。

構文

```
DBMS_STATS.DELETE_TABLE_STATS (
    ownname          VARCHAR2,
    tabname          VARCHAR2,
    partname          VARCHAR2 DEFAULT NULL,
    stattab          VARCHAR2 DEFAULT NULL,
    statid            VARCHAR2 DEFAULT NULL,
    cascade_parts     BOOLEAN   DEFAULT TRUE,
    cascade_columns   BOOLEAN   DEFAULT TRUE,
    cascade_indexes   BOOLEAN   DEFAULT TRUE,
    statown           VARCHAR2 DEFAULT NULL);
```

パラメータ

表 52-13 DELETE_TABLE_STATS プロシージャのパラメータ

パラメータ	説明
ownname	スキーマの名前。
tabname	列が所属している表の名前。
partname	統計情報を取得する表パーティションの名前。表がパーティション化されていて、partname が NULL の場合、統計情報はグローバルな表レベルで取り出されます。
stattab	統計情報を取り出す場所を示すユーザー統計表の識別子。stattab が NULL の場合、統計情報はディクショナリから直接取り出されます。
statid	stattab 内の統計情報を関連付ける識別子 (オプション) (stattab が NULL でない場合のみ該当します)。
cascade_parts	表がパーティション化されていて、partname が NULL の場合、このパラメータを TRUE に設定すると、基礎となるすべてのパーティションについても表の統計情報が削除されます。
cascade_columns	基礎となるすべての列について、DELETE_COLUMN_STATS をコールする必要があることを示します (cascade_parts パラメータを渡します)。
cascade_indexes	基礎となるすべての索引について、DELETE_INDEX_STATS をコールする必要があることを示します (cascade_parts パラメータを渡します)。
statown	stattab を含んだスキーマ (ownname と異なる場合)。

例外

ORA-20000: オブジェクトが存在しないか、または権限が不十分です。

DELETE_SCHEMA_STATS プロシージャ

このプロシージャは、スキーマ全体の統計情報を削除します。

構文

```
DBMS_STATS.DELETE_SCHEMA_STATS (  
    ownname VARCHAR2,  
    stattab VARCHAR2 DEFAULT NULL,  
    statid VARCHAR2 DEFAULT NULL,  
    statown VARCHAR2 DEFAULT NULL);
```

パラメータ

表 52-14 DELETE_SCHEMA_STATS プロシージャのパラメータ

パラメータ	説明
ownname	スキーマの名前。
stattab	統計情報を削除する場所を示すユーザー統計表の識別子。 stattab が NULL の場合、統計情報はディクショナリから直接削除されます。
statid	stattab 内の統計情報を関連付ける識別子（オプション） (stattab が NULL でない場合のみ該当します)。
statown	stattab を含んだスキーマ (ownname と異なる場合)。

例外

ORA-20000: オブジェクトが存在しないか、または権限が不十分です。

DELETE_DATABASE_STATS プロシージャ

このプロシージャは、データベース全体の統計情報を削除します。

構文

```
DBMS_STATS.DELETE_DATABASE_STATS (  
    stattab VARCHAR2 DEFAULT NULL,  
    statid VARCHAR2 DEFAULT NULL,  
    statown VARCHAR2 DEFAULT NULL);
```

パラメータ

表 52-15 DELETE_DATABASE_STATS プロシージャのパラメータ

パラメータ	説明
stattab	統計情報を削除する場所を示すユーザー統計表の識別子。 stattab が NULL の場合、統計情報はディクショナリから直接削除されます。
statid	stattab 内の統計情報を関連付ける識別子 (オプション) (stattab が NULL でない場合のみ該当します)。
statown	stattab を含むスキーマ。stattab が NULL ではなく、 statown が NULL の場合は、データベース内のすべてのスキーマ に、同じ名前の stattab を持つユーザー統計表が含まれている とみなされます。

例外

ORA-20000: オブジェクトが存在しないか、または権限が不十分です。

統計情報の転送

次のプロシージャを使用して、ディクショナリからユーザー統計表へ (export_*)、およびユーザー統計表からディクショナリへ (import_*) 統計情報を転送できます。

```
CREATE_STAT_TABLE
DROP_STAT_TABLE

EXPORT_COLUMN_STATS
EXPORT_INDEX_STATS
EXPORT_TABLE_STATS
EXPORT_SCHEMA_STATS
EXPORT_DATABASE_STATS

IMPORT_COLUMN_STATS
IMPORT_INDEX_STATS
IMPORT_TABLE_STATS
IMPORT_SCHEMA_STATS
IMPORT_DATABASE_STATS
```


CREATE_STAT_TABLE プロシージャ

このプロシージャは、統計情報を保持できる ownname のスキーマにある stattab の名前で表を作成します。この表は、このパッケージのプロシージャを介して単独にアクセスされるため、この表を構成する列と型は互いに関係がありません。

構文

```
DBMS_STATS.CREATE_STAT_TABLE (  
    ownname  VARCHAR2,  
    stattab  VARCHAR2,  
    tblspace VARCHAR2 DEFAULT NULL);
```

パラメータ

表 52-16 CREATE_STAT_TABLE プロシージャのパラメータ

パラメータ	説明
ownname	スキーマの名前。
stattab	作成する表の名前。ユーザーがディクショナリ統計情報を直接変更しない場合、この値は、stattab パラメータとして他のプロシージャに渡されます。
tblspace	統計表を作成する表領域。このパラメータを指定しないと、統計表はユーザーのデフォルトの表領域に作成されます。

例外

ORA-20000: 表がすでに存在するか、権限が不十分です。

ORA-20001: 表領域が存在しません。

DROP_STAT_TABLE プロシージャ

このプロシージャは、ユーザー統計表を削除します。

構文

```
DBMS_STATS.DROP_STAT_TABLE (  
    ownname VARCHAR2,  
    stattab VARCHAR2);
```

パラメータ

表 52-17 DROP_STAT_TABLE プロシージャのパラメータ

パラメータ	説明
ownname	スキーマの名前
stattab	ユーザー統計表の識別子

例外

ORA-20000: 表が存在しないか、または権限が不十分です。

EXPORT_COLUMN_STATS プロシージャ

このプロシージャは、特定の列に関する統計情報を取り出し、stattab で識別されるユーザー統計表に格納します。

構文

```
DBMS_STATS.EXPORT_COLUMN_STATS (  
    ownname  VARCHAR2,  
    tabname  VARCHAR2,  
    colname  VARCHAR2,  
    partname VARCHAR2 DEFAULT NULL,  
    stattab  VARCHAR2,  
    statid   VARCHAR2 DEFAULT NULL,  
    statown  VARCHAR2 DEFAULT NULL);
```

パラメータ

表 52-18 EXPORT_COLUMN_STATS プロシージャのパラメータ

パラメータ	説明
ownname	スキーマの名前。
tabname	列が所属している表の名前。
colname	列の名前。
partname	表パーティション名。表がパーティション化されていて、partname が NULL の場合、グローバルでパーティション化された列の統計情報がエクスポートされます。
stattab	統計情報の格納場所を示すユーザー統計表の識別子。
statid	stattab 内の統計情報を関連付ける識別子（オプション）。
statown	stattab を含んだスキーマ（ownname と異なる場合）。

例外

ORA-20000: オブジェクトが存在しないか、または権限が不十分です。

EXPORT_INDEX_STATS プロシージャ

このプロシージャは、特定の索引に関する統計情報を取り出し、stattab で識別されるユーザー統計表に格納します。

構文

```
DBMS_STATS.EXPORT_INDEX_STATS (  
  ownname  VARCHAR2,  
  indname  VARCHAR2,  
  partname VARCHAR2 DEFAULT NULL,  
  stattab  VARCHAR2,  
  statid   VARCHAR2 DEFAULT NULL,  
  statown  VARCHAR2 DEFAULT NULL);
```

パラメータ

表 52-19 EXPORT_INDEX_STATS プロシージャのパラメータ

パラメータ	説明
ownname	スキーマの名前。
indname	索引名。
partname	索引パーティション名。索引がパーティション化されていて、partname が NULL の場合、グローバルでパーティション化された索引統計情報がエクスポートされます。
stattab	統計情報の格納場所を示すユーザー統計表の識別子。
statid	stattab 内の統計情報を関連付ける識別子（オプション）。
statown	stattab を含んだスキーマ（ownname と異なる場合）。

例外

ORA-20000: オブジェクトが存在しないか、または権限が不十分です。

EXPORT_TABLE_STATS プロシージャ

このプロシージャは、特定の表に関する統計情報を取り出し、ユーザー統計表に格納します。cascade を使用すると、指定した表に関連付けられている索引および列統計情報もすべてエクスポートされます。

構文

```
DBMS_STATS.EXPORT_TABLE_STATS (  
  ownname  VARCHAR2,  
  tabname  VARCHAR2,  
  partname VARCHAR2 DEFAULT NULL,  
  stattab  VARCHAR2,  
  statid   VARCHAR2 DEFAULT NULL,  
  cascade  BOOLEAN  DEFAULT TRUE,  
  statown  VARCHAR2 DEFAULT NULL);
```

パラメータ

表 52-20 EXPORT_TABLE_STATS プロシージャのパラメータ

パラメータ	説明
ownname	スキーマの名前。
tabname	表の名前。
partname	表パーティション名。表がパーティション化されていて、partname が NULL の場合、グローバルでパーティション化された表統計情報がエクスポートされます。
stattab	統計情報の格納場所を示すユーザー統計表の識別子。
statid	stattab 内の統計情報を関連付ける識別子（オプション）。
cascade	TRUE の場合は、この表の列と索引の統計情報もまたエクスポートされます。
statown	stattab を含んだスキーマ（ownname と異なる場合）。

例外

ORA-20000: オブジェクトが存在しないか、または権限が不十分です。

EXPORT_SCHEMA_STATS プロシージャ

このプロシージャは、ownname で識別されるスキーマ内のすべてのオブジェクトに関する統計情報を取り出し、stattab で識別されるユーザー統計表に格納します。

構文

```
DBMS_STATS.EXPORT_SCHEMA_STATS (  
    ownname VARCHAR2,  
    stattab VARCHAR2,  
    statid  VARCHAR2 DEFAULT NULL,  
    statown VARCHAR2 DEFAULT NULL);
```

パラメータ

表 52-21 EXPORT_SCHEMA_STATS プロシージャのパラメータ

パラメータ	説明
ownname	スキーマの名前
stattab	統計情報の格納場所を示すユーザー統計表の識別子
statid	stattab 内の統計情報を関連付ける識別子（オプション）
statown	stattab を含んだスキーマ（ownname と異なる場合）

例外

ORA-20000: オブジェクトが存在しないか、または権限が不十分です。

EXPORT_DATABASE_STATS プロシージャ

このプロシージャは、データベース内のすべてのオブジェクトに関する統計情報を取り出し、statown.stattab で識別されるユーザー統計表に格納します。

構文

```
DBMS_STATS.EXPORT_DATABASE_STATS (  
    stattab VARCHAR2,  
    statid  VARCHAR2 DEFAULT NULL,  
    statown VARCHAR2 DEFAULT NULL);
```

パラメータ

表 52-22 EXPORT_DATABASE_STATS プロシージャのパラメータ

パラメータ	説明
stattab	統計情報の格納場所を示すユーザー統計表の識別子。
statid	stattab 内の統計情報を関連付ける識別子（オプション）。
statown	stattab を含むスキーマ。statown が NULL の場合、データベース内のすべてのスキーマには stattab の名前を持つユーザー統計表が含まれているとみなされます。

例外

ORA-20000: オブジェクトが存在しないか、または権限が不十分です。

IMPORT_COLUMN_STATS プロシージャ

このプロシージャは、stattab で識別されるユーザー統計表から特定の列に関する統計情報を取り出し、ディクショナリに格納します。

構文

```
DBMS_STATS.IMPORT_COLUMN_STATS (  
    ownname  VARCHAR2,  
    tabname  VARCHAR2,  
    colname  VARCHAR2,  
    partname VARCHAR2 DEFAULT NULL,  
    stattab  VARCHAR2,  
    statid   VARCHAR2 DEFAULT NULL,  
    statown  VARCHAR2 DEFAULT NULL);
```

パラメータ

表 52-23 IMPORT_COLUMN_STATS プロシージャのパラメータ

パラメータ	説明
ownname	スキーマの名前。
tabname	列が所属している表の名前。
colname	列の名前。
partname	表パーティション名。表がパーティション化されていて、partname が NULL の場合、グローバルでパーティション化された列統計情報がインポートされます。
stattab	統計情報を取り出す場所を示すユーザー統計表の識別子。
statid	stattab 内の統計情報を関連付ける識別子（オプション）。
statown	stattab を含んだスキーマ（ownname と異なる場合）。

例外

ORA-20000: オブジェクトが存在しないか、または権限が不十分です。

ORA-20001: ユーザー統計表の値が無効または矛盾しています。

IMPORT_INDEX_STATS プロシージャ

このプロシージャは、stattab で識別されるユーザー統計表から特定の索引に関する統計情報を取り出し、ディクショナリに格納します。

構文

```
DBMS_STATS.IMPORT_INDEX_STATS (
  ownname  VARCHAR2,
  indname  VARCHAR2,
  partname VARCHAR2 DEFAULT NULL,
  stattab  VARCHAR2,
  statid   VARCHAR2 DEFAULT NULL,
  statown  VARCHAR2 DEFAULT NULL);
```

パラメータ

表 52-24 IMPORT_INDEX_STATS プロシージャのパラメータ

パラメータ	説明
ownname	スキーマの名前。
indname	索引名。
partname	索引パーティション名。索引がパーティション化されていて、partname が NULL の場合、グローバルでパーティション化された索引統計情報がインポートされます。
stattab	統計情報を取り出す場所を示すユーザー統計表の識別子。
statid	stattab 内の統計情報を関連付ける識別子（オプション）。
statown	stattab を含んだスキーマ（ownname と異なる場合）。

例外

- ORA-20000: オブジェクトが存在しないか、または権限が不十分です。
- ORA-20001: ユーザー統計表の値が無効または矛盾しています。

IMPORT_TABLE_STATS プロシージャ

このプロシージャは、stattab で識別されるユーザー統計表から特定の表に関する統計情報を取り出し、ディクショナリに格納します。cascade を使用すると、指定した表に関連付けられている索引および列統計情報もすべてインポートされます。

構文

```
DBMS_STATS.IMPORT_TABLE_STATS (  
    ownname  VARCHAR2,  
    tabname  VARCHAR2,  
    partname VARCHAR2 DEFAULT NULL,  
    stattab  VARCHAR2,  
    statid   VARCHAR2 DEFAULT NULL,  
    cascade  BOOLEAN  DEFAULT TRUE,  
    statown  VARCHAR2 DEFAULT NULL);
```

パラメータ

表 52-25 IMPORT_TABLE_STATS プロシージャのパラメータ

パラメータ	説明
ownname	スキーマの名前。
tabname	表の名前。
partname	表パーティション名。表がパーティション化されていて、partname が NULL の場合、グローバルでパーティション化された表統計情報がインポートされます。
stattab	統計情報を取り出す場所を示すユーザー統計表の識別子。
statid	stattab 内の統計情報を関連付ける識別子（オプション）。
cascade	TRUE の場合は、この表の列と索引の統計情報もまたインポートされます。
statown	stattab を含んだスキーマ（ownname と異なる場合）。

例外

ORA-20000: オブジェクトが存在しないか、または権限が不十分です。

ORA-20001: ユーザー統計表の値が無効または矛盾しています。

IMPORT_SCHEMA_STATS プロシージャ

このプロシージャは、ownname で識別されるスキーマ内のすべてのオブジェクトに関する統計情報をユーザー統計表から取り出し、ディクショナリに格納します。

構文

```
DBMS_STATS.IMPORT_SCHEMA_STATS (  
    ownname VARCHAR2,  
    stattab VARCHAR2,  
    statid   VARCHAR2 DEFAULT NULL,  
    statown  VARCHAR2 DEFAULT NULL);
```

パラメータ

表 52-26 IMPORT_SCHEMA_STATS プロシージャのパラメータ

パラメータ	説明
ownname	スキーマの名前
stattab	統計情報を取り出す場所を示すユーザー統計表の識別子
statid	stattab 内の統計情報を関連付ける識別子（オプション）
statown	stattab を含んだスキーマ（ownname と異なる場合）

例外

- ORA-20000: オブジェクトが存在しないか、または権限が不十分です。
- ORA-20001: ユーザー統計表の値が無効または矛盾しています。

IMPORT_DATABASE_STATS プロシージャ

このプロシージャは、データベース内のすべてのオブジェクトに関する統計情報をユーザー統計表から取り出し、ディクショナリに格納します。

構文

```
DBMS_STATS.IMPORT_DATABASE_STATS (  
    stattab VARCHAR2,  
    statid   VARCHAR2 DEFAULT NULL,  
    statown  VARCHAR2 DEFAULT NULL);
```

パラメータ

表 52-27 IMPORT_DATABASE_STATS プロシージャのパラメータ

パラメータ	説明
stattab	統計情報を取り出す場所を示すユーザー統計表の識別子。
statid	stattab 内の統計情報を関連付ける識別子（オプション）。
statown	stattab を含むスキーマ。statown が NULL の場合、データベース内のすべてのスキーマには stattab の名前を持つユーザー統計表が含まれているとみなされます。

例外

- ORA-20000: オブジェクトが存在しないか、または権限が不十分です。
- ORA-20001: ユーザー統計表の値が無効または矛盾しています。

オブティマイザ統計情報の収集

次のプロシージャを使用すると、特定のクラスのオブティマイザ統計情報を収集でき、ANALYZE コマンドによってパフォーマンスの向上が可能になります。

GATHER_INDEX_STATS
GATHER_TABLE_STATS
GATHER_SCHEMA_STATS
GATHER_DATABASE_STATS

statown、stattab および statid パラメータは、パッケージに対して、新規の統計情報を収集する前に、指定した表内の現行の統計情報をバックアップするように指示します。

関連オブジェクトに十分な統計情報がある場合、導出オブジェクトの統計情報を生成するために、Oracle は次のプロシージャも提供します。

GENERATE_STATS

GATHER_INDEX_STATS プロシージャ

このプロシージャは、索引の統計情報を収集します。これは、`ANALYZE INDEX [ownname.] indname [PARTITION partname] COMPUTE STATISTICS | ESTIMATE STATISTICS SAMPLE estimate_percent PERCENT` を実行するのと同じです。

並列的には実行しません。

構文

```
DEMS_STATS.GATHER_INDEX_STATS (  
    ownname          VARCHAR2,  
    indname          VARCHAR2,  
    partname         VARCHAR2 DEFAULT NULL,  
    estimate_percent NUMBER  DEFAULT NULL,  
    stattab          VARCHAR2 DEFAULT NULL,  
    statid           VARCHAR2 DEFAULT NULL,  
    statown          VARCHAR2 DEFAULT NULL);
```

パラメータ

表 52-28 GATHER_INDEX_STATS プロシージャのパラメータ

パラメータ	説明
ownname	分析する索引のスキーマ。
indname	索引名。
partname	パーティション名。
estimate_percent	推定する行のパーセント（NULL は計算を意味します）。有効な範囲は、0.000001 ～ 100 です。この値は、よい結果をアーカイブするために自動的に増加できます。
stattab	現行の統計情報を保存する場所を示すユーザー統計表の識別子。
statid	stattab 内の統計情報を関連付ける識別子（オプション）。
statown	stattab を含んだスキーマ（ownname と異なる場合）。

例外

ORA-20000: 索引が存在しないか、または権限が不十分です。

ORA-20001: 入力値が無効です。

GATHER_TABLE_STATS プロシージャ

このプロシージャは、表と列（および索引）の統計情報を収集します。このプロシージャは、可能な限り多くの作業を並列化しますが、個々のパラメータで説明するように、いくつかの制限があります。分析する表に対する SELECT 権限がユーザーにない場合、この操作は並列化しません。

構文

```
DBMS_STATS.GATHER_TABLE_STATS (  
    ownname          VARCHAR2,  
    tabname           VARCHAR2,  
    partname          VARCHAR2 DEFAULT NULL,  
    estimate_percent  NUMBER   DEFAULT NULL,  
    block_sample      BOOLEAN   DEFAULT FALSE,  
    method_opt        VARCHAR2 DEFAULT 'FOR ALL COLUMNS SIZE 1',  
    degree            NUMBER   DEFAULT NULL,  
    granularity       VARCHAR2 DEFAULT 'DEFAULT',  
    cascade            BOOLEAN   DEFAULT FALSE,  
    stattab           VARCHAR2 DEFAULT NULL,  
    statid            VARCHAR2 DEFAULT NULL,  
    statown           VARCHAR2 DEFAULT NULL);
```

パラメータ

表 52-29 GATHER_TABLE_STATS プロシージャのパラメータ

パラメータ	説明
ownname	分析する表のスキーマ。
tabname	表の名前。
partname	パーティション名。
estimate_percent	推定する行のパーセント（NULL は計算を意味します）。有効な範囲は、0.000001 ～ 100 です。この値は、よい結果をアーカイブするために自動的に増加できます。
block_sample	ランダム行サンプリングのかわりにランダム・ブロック・サンプリングを使用するかどうかを示します。ランダム・ブロック・サンプリングがより効率的ですが、データがディスク上にランダムに分散していない場合、サンプル値はある程度相関があります。統計情報の推定を行った場合のみ該当します。

表 52-29 GATHER_TABLE_STATS プロシージャのパラメータ

パラメータ	説明
method_opt	次の書式の方法オプション（句 'SIZE 1' は、並列的に統計情報を収集し非表示の句を使用するために必要です）。 FOR ALL [INDEXED HIDDEN] COLUMNS [SIZE integer] FOR COLUMNS [SIZE integer] column attribute [,column attribute ...] オプティマイザに関連する表統計情報は、常に収集されます。
degree	並列度（NULL は、表のデフォルト値の使用を意味します）。
granularity	収集する統計情報の細分化（表がパーティション化されている場合のみ該当します）。 DEFAULT: グローバル・レベルでパーティション・レベルの統計情報を収集します。 SUBPARTITION: サブパーティション・レベルの統計情報を収集します。 PARTITION: パーティション・レベルの統計情報を収集します。 GLOBAL: グローバルな統計情報を収集します。 ALL: すべての統計情報（サブパーティション、パーティションおよびグローバル）を収集します。
cascade	表の索引について統計情報を収集します。索引統計情報の収集は並列化されません。このオプションを使用することは、表の各索引で gather_index_stats プロシージャを実行するのと同じです。
stattab	現行の統計情報を保存する場所を示すユーザー統計表の識別子。
statid	stattab 内の統計情報を関連付ける識別子（オプション）。
statown	stattab を含んだスキーマ（ownname と異なる場合）。

例外

- ORA-20000: 表が存在しないか、または権限が不十分です。
- ORA-20001: 入力値が無効です。

GATHER_SCHEMA_STATS プロシージャ

このプロシージャは、スキーマ内のすべてのオブジェクトに関する統計情報を収集します。

構文

```
DBMS_STATS.GATHER_SCHEMA_STATS (  
    ownname          VARCHAR2,  
    estimate_percent NUMBER DEFAULT NULL,  
    block_sample     BOOLEAN DEFAULT FALSE,  
    method_opt       VARCHAR2 DEFAULT 'FOR ALL COLUMNS SIZE 1',  
    degree           NUMBER DEFAULT NULL,  
    granularity      VARCHAR2 DEFAULT 'DEFAULT',  
    cascade          BOOLEAN DEFAULT FALSE);
```

```
DBMS_STATS.GATHER_SCHEMA_STATS (  
    ownname          VARCHAR2,  
    estimate_percent NUMBER DEFAULT NULL,  
    block_sample     BOOLEAN DEFAULT FALSE,  
    method_opt       VARCHAR2 DEFAULT 'FOR ALL COLUMNS SIZE 1',  
    degree           NUMBER DEFAULT NULL,  
    granularity      VARCHAR2 DEFAULT 'DEFAULT',  
    cascade          BOOLEAN DEFAULT FALSE,  
    stattab          VARCHAR2 DEFAULT NULL,  
    statid           VARCHAR2 DEFAULT NULL,  
    options          VARCHAR2 DEFAULT 'GATHER',  
    objlist          OUT  ObjectTab,  
    statown          VARCHAR2 DEFAULT NULL);
```

パラメータ

表 52-30 GATHER_SCHEMA_STATS プロシージャのパラメータ

パラメータ	説明
ownname	分析するスキーマ (NULL は現行のスキーマを意味します)。
estimate_percent	推定する行のパーセント (NULL は計算を意味します)。有効な範囲は、0.000001 ～ 100 です。
block_sample	ランダム行サンプリングのかわりにランダム・ブロック・サンプリングを使用するかどうかを示します。ランダム・ブロック・サンプリングがより効率的ですが、データがディスク上にランダムに分散していない場合、サンプル値はある程度相関があります。統計情報の推定を行った場合のみ該当します。

表 52-30 GATHER_SCHEMA_STATS プロシージャのパラメータ

パラメータ	説明
method_opt	次の書式の方法オプション（句 'SIZE 1' は、並列的に統計情報を収集し非表示の句を使用するために必要です）。 FOR ALL [INDEXED HIDDEN] COLUMNS [SIZE integer] この値は、すべての個別表に渡されます。
degree	並列度（NULL は、表のデフォルト値の使用を意味します）。
granularity	収集する統計情報の細分化（表がパーティション化されている場合のみ該当します）。 DEFAULT: グローバル・レベルでパーティション・レベルの統計情報を収集します。 SUBPARTITION: サブパーティション・レベルの統計情報を収集します。 PARTITION: パーティション・レベルの統計情報を収集します。 GLOBAL: グローバルな統計情報を収集します。 ALL: すべての統計情報（サブパーティション、パーティションおよびグローバル）を収集します。
cascade	索引についても統計情報を収集します。 索引統計情報の収集は並列化されません。このオプションを使用することは、表と列の統計情報の収集に加えて、スキーマ内の各索引で gather_index_stats プロシージャを実行するのと同じです。
stattab	現行の統計情報を保存する場所を示すユーザー統計表の識別子。
statid	stattab 内の統計情報を関連付ける識別子（オプション）。

表 52-30 GATHER_SCHEMA_STATS プロシージャのパラメータ

パラメータ	説明
options	統計情報を収集するオブジェクトの詳細は、次のように指定します。 GATHER: スキーマ内のすべてのオブジェクトに関する統計情報を収集します。 GATHER STALE: *_tab_modifications ビューを調べて判別した失効オブジェクトについて、統計情報を収集します。また、失効と判別されたオブジェクトのリストも戻します。 GATHER EMPTY: 現在統計情報がないオブジェクトについて統計情報を収集し、統計情報なしと判別されたオブジェクトのリストも戻します。 LIST STALE: *_tab_modifications ビューを調べて判別した失効オブジェクトのリストを戻します。 LIST EMPTY: 現在統計情報がないオブジェクトのリストを戻します。
objlist	失効または空と判別されたオブジェクトのリスト。
stattown	stattab を含んだスキーマ (ownname と異なる場合)。

例外

- ORA-20000: スキーマが存在しないか、または権限が不十分です。
- ORA-20001: 入力値が無効です。

GATHER_DATABASE_STATS プロシージャ

このプロシージャは、データベース内のすべてのオブジェクトに関する統計情報を収集します。

構文

```
DBMS_STATS.GATHER_DATABASE_STATS (  
  estimate_percent NUMBER   DEFAULT NULL,  
  block_sample      BOOLEAN  DEFAULT FALSE,  
  method_opt        VARCHAR2 DEFAULT 'FOR ALL COLUMNS SIZE 1',  
  degree            NUMBER   DEFAULT NULL,  
  granularity        VARCHAR2 DEFAULT 'DEFAULT',  
  cascade           BOOLEAN  DEFAULT FALSE);  
  
DBMS_STATS.GATHER_DATABASE_STATS (  
  estimate_percent NUMBER   DEFAULT NULL,  
  block_sample      BOOLEAN  DEFAULT FALSE,
```

```
method_opt      VARCHAR2 DEFAULT 'FOR ALL COLUMNS SIZE 1',
degree          NUMBER   DEFAULT NULL,
granularity     VARCHAR2 DEFAULT 'DEFAULT',
cascade         BOOLEAN  DEFAULT FALSE,
stattab         VARCHAR2 DEFAULT NULL,
statid          VARCHAR2 DEFAULT NULL,
options         VARCHAR2 DEFAULT 'GATHER',
objlist         OUT      ObjectTab,
statown         VARCHAR2 DEFAULT NULL);
```

パラメータ

表 52-31 GATHER_DATABASE_STATS プロシージャのパラメータ

パラメータ	説明
estimate_percent	推定する行のパーセント (NULL は計算を意味します)。有効な範囲は、0.000001 ~ 100 です。
block_sample	ランダム行サンプリングのかわりにランダム・ブロック・サンプリングを使用するかどうかを示します。ランダム・ブロック・サンプリングがより効率的ですが、データがディスク上にランダムに分散していない場合、サンプル値はある程度相関があります。統計情報の推定を行った場合のみ該当します。
method_opt	次の書式の方法オプション (句 'SIZE 1' は、並列的に統計情報を収集し非表示の句を使用するために必要です)。 FOR ALL [INDEXED HIDDEN] COLUMNS [SIZE integer] この値は、すべての個別表に渡されます。
degree	並列度 (NULL は、表のデフォルト値の使用を意味します)。
granularity	収集する統計情報の細分化 (表がパーティション化されている場合のみ該当します)。 DEFAULT: グローバル・レベルでパーティション・レベルの統計情報を収集します。 SUBPARTITION: サブパーティション・レベルの統計情報を収集します。 PARTITION: パーティション・レベルの統計情報を収集します。 GLOBAL: グローバルな統計情報を収集します。 ALL: すべての統計情報 (サブパーティション、パーティションおよびグローバル) を収集します。

表 52-31 GATHER_DATABASE_STATS プロシージャのパラメータ

パラメータ	説明
cascade	索引についても統計情報を収集します。索引統計情報の収集は並列化されません。このオプションを使用することは、表と列の統計情報の収集に加えて、データベース内の各索引で gather_index_stats プロシージャを実行するのと同じです。
stattab	現行の統計情報を保存する場所を示すユーザー統計表の識別子。 統計表は、分析するオブジェクトと同じスキーマに常駐するとみなされるので、各スキーマにこのオプションを使用するための表が 1 つ必要です。
statid	stattab 内の統計情報を関連付ける識別子（オプション）。
options	統計情報を収集するオブジェクトの詳細は、次のように指定します。 GATHER STALE: *_tab_modifications ビューを調べて判別した失効オブジェクトについて、統計情報を収集します。また、失効と判別されたオブジェクトのリストも戻します。 GATHER EMPTY: 現在統計情報がないオブジェクトについて統計情報を収集し、統計情報なしと判別されたオブジェクトのリストも戻します。 LIST STALE: *_tab_modifications ビューを調べて判別した失効オブジェクトのリストを戻します。 LIST EMPTY: 現在統計情報がないオブジェクトのリストを戻します。
objlist	失効または空と判別されたオブジェクトのリスト。
statown	stattab を含んだスキーマ (ownname と異なる場合)。

例外

- ORA-20000: 権限が不十分です。
- ORA-20001: 入力値が無効です。

GENERATE_STATS プロシージャ

このプロシージャは、関連するオブジェクトで以前に収集した統計情報から、オブジェクトの統計情報を生成します。完全に移入されたスキーマについては、より正確な統計情報が必要な場合は、GATHER プロシージャをかわりに使用する必要があります。現在サポートされているオブジェクトは、B-tree 索引とビットマップ索引です。

構文

```
DEMS_STATS.GENERATE_STATS (  
    ownname    VARCHAR2,  
    objname    VARCHAR2,  
    organized  NUMBER DEFAULT 7);
```

パラメータ

表 52-32 GENERATE_STATS プロシージャのパラメータ

パラメータ	説明
ownname	オブジェクトのスキーマ。
objname	オブジェクト名。
organized	<p>索引とその基礎となる表の間で関連付けられた順位付けの量。複雑に編成されている索引には、ディスク上の連続行を参照する連続索引キーがその表（同じブロック）に対してあります。複雑に編成されていない索引には、ディスク上の異なる表ブロックを参照する連続キーがあります。</p> <p>このパラメータは、B-tree 索引に対してのみ使用します。数値は、0 ～ 10 の範囲で使用でき、0 は完全に編成された索引、10 は完全に編成解除された索引を示します。</p>

例外

- ORA-20000: サポートされていないオブジェクト型で、オブジェクトは存在しません。
- ORA-20001: 無効なオプションまたは無効な統計情報です。

例

使用例 前回統計情報を収集してから、emp 表に対して大量の変更がありました。コストベースのオプティマイザが最適プランを選択するために、統計情報を再度収集する必要があります。しかし、ユーザーは、現行のプランが許容されると、新規の統計情報によってオプティマイザが誤ったプランを選択することを懸念しています。ユーザーは、次のように指定できます。

```
BEGIN
  DBMS_STATS.CREATE_STAT_TABLE ('scott', 'savestats');
  DBMS_STATS.GATHER_TABLE_STATS ('scott', 'emp', 5, stattab => 'savestats');
END;
```

この操作は、新規の統計情報を emp 表に収集しますが、最初に、元の統計情報をユーザー統計表 emp.savestats に保存します。

ユーザーが、新規統計情報によってオプティマイザが誤ったプランを生成していると考えている場合は、元の統計情報を次のようにリストアできます。

```
BEGIN
  DBMS_STATS.DELETE_TABLE_STATS ('scott', 'emp');
  DBMS_STATS.IMPORT_TABLE_STATS ('scott', 'emp', stattab => 'savestats');
END;
```

DBMS_TRACE

Oracle8i PL/SQL は、サーバー上での PL/SQL プログラムの実行をトレースするために、API を提供します。サーバー上に DBMS_TRACE パッケージとしてインプリメントされているトレース API を使用すると、PL/SQL ファンクション、プロシージャおよび例外をトレースできます。

DBMS_TRACE は、セッションで PL/SQL トレースを開始および停止するサブプログラムを提供します。トレース・データはプログラムの実行時に収集され、データベース表に書き出されます。

一般的なセッションには、次の処理が含まれます。

- セッションで PL/SQL トレースを開始します (DBMS_TRACE.SET_PLSQL_TRACE)。
- トレースするアプリケーションを実行します。
- セッションでの PL/SQL トレースを停止します (DBMS_TRACE.CLEAR_PLSQL_TRACE)。

要件

このパッケージは、SYS の下で作成する必要があります。

制限事項

マルチスレッド・サーバー (MTS) では、PL/SQL トレースを使用できません。

定数

DBMS_TRACE では、次の定数を使用します。

trace_all_calls	constant INTEGER := 1;
trace_enabled_calls	constant INTEGER := 2;
trace_all_exceptions	constant INTEGER := 4;
trace_enabled_exceptions	constant INTEGER := 8;
trace_all_sql	constant INTEGER := 32;
trace_enabled_sql	constant INTEGER := 64;
trace_all_lines	constant INTEGER := 128;
trace_enabled_lines	constant INTEGER := 256;
trace_stop	constant INTEGER := 16384;
trace_pause	constant INTEGER := 4096;
trace_resume	constant INTEGER := 8192;
trace_limit	constant INTEGER := 16;
trace_major_version	constant BINARY_INTEGER := 1;
trace_minor_version	constant BINARY_INTEGER := 0;

これらの定数についてはすべて記号形式（定数名）を使用することをお薦めします。

DBMS_TRACE の使用方法

データ量の制御

大規模なアプリケーションをプロファイルすると、データ量が膨大になる可能性があります。トレース・データの収集に関する特定のプログラム・ユニットを使用可能にして、収集するデータ量を制御できます。

プログラム・ユニットは、コンパイルとデバッグを行って使用可能にできます。次のいずれかの方法で行います。

```
alter session set plsql_debug=true;
create or replace ... /* create the library units - debug information will be
generated */
```

または

```
/* recompile specific library unit with debug option */  
alter [PROCEDURE | FUNCTION | PACKAGE BODY] <libunit-name> compile debug;
```

注意： 2 番目の方法は、無名ブロックに対しては使用できません。

SET PLSQL TRACE プロシージャの TRACE_LEVEL パラメータに TRACE_LIMIT を指定することにより、最新の 8,192 レコード（概算）のみを保持して、データベースで使用する記憶域量を制限できます。

DBMS_TRACE 出力を収集するデータベース表の作成

DBMS_TRACE パッケージで出力を書き込むためのデータベース表を作成する必要があります。作成しないと、データが収集されません。これらの表を作成するには、スクリプト TRACETAB.SQL を実行します。このスクリプトで作成された表はシステムが所有します。

トレース・データの収集

トレースできる PL/SQL 機能は、スクリプト DBMSPT.SQL に記述されています。主なトレース機能には、次のトレースがあります。

- [コールのトレース](#)
- [例外のトレース](#)
- [SQL のトレース](#)
- [行のトレース](#)

DBMS_TRACE の追加機能として、トレースの解析と再開および出力の制限もできます。

コールのトレース 使用可能なコールのトレースには、次の 2 つのレベルがあります。

- レベル 1: すべてのコールをトレースします。これは、定数 trace_all_calls に対応します。
- レベル 2: 使用可能なプログラム・ユニットのみ、コールをトレースします。これは、定数 trace_enabled_calls に対応します。

リモート・プロシージャ・コール（RPC）については、使用可能かどうかを検出できないため、RPC ではレベル 1 でのみトレースできます。

例外のトレース 使用可能な例外のトレースには、次の 2 つのレベルがあります。

- レベル 1: すべての例外をトレースします。これは、定数 trace_all_exceptions に対応します。

-
- レベル 2: 使用可能なプログラム・ユニットのみ、発生した例外をトレースします。これは、定数 `trace_enabled_exceptions` に対応します。

SQL のトレース 使用可能な SQL のトレースには、次の 2 つのレベルがあります。

- レベル 1: すべての SQL をトレースします。これは、定数 `trace_all_sql` に対応します。
- レベル 2: 使用可能なプログラム・ユニットでのみ、SQL をトレースします。これは、定数 `trace_enabled_sql` に対応します。

行のトレース 使用可能な行のトレースには、次の 2 つのレベルがあります。

- レベル 1: すべての行をトレースします。これは、定数 `trace_all_lines` に対応します。
- レベル 2: 使用可能なプログラム・ユニットでのみ、行をトレースします。これは、定数 `trace_enabled_lines` に対応します。

行のトレース時には、行番号が変わるたびにレコードがデータベースに追加されます。プロシージャのコールおよびそのリターンによって行番号が変わる場合も、トレースの対象に含まれます。

注意： すべてのタイプのトレースでは、レベル 1 がレベル 2 を上書きします。たとえば、レベル 1 とレベル 2 の両方が使用可能な場合は、レベル 1 が優先されます。

収集されたデータ

使用可能なプログラム・ユニットについてのみトレースが要求されていて、現行のプログラム・ユニットが使用可能ではない場合、トレース・データは書き込まれません。

コールをトレースする場合は、コールとリターンの両方がトレースされます。トレースが使用可能であるかどうかのチェックは、コールされるルーチンまたはコールするルーチンのいずれか一方が使用可能な場合に、トレース実施と判断されます。

コールのトレースでは、プログラム・ユニットの型、名前および行番号が常に出力されます。さらに、コール側のスタックの深さが書き込まれます。コール側のユニットが使用可能な場合は、コール側プロシージャ名も出力されます。コールされる側のユニットが使用可能な場合は、コールされるプロシージャ名が出力されます。

例外のトレースでは、行番号が書き込まれます。例外が発生すると、その例外がユーザー定義か事前定義のいずれであるかが示されます。事前定義の例外の場合は例外番号も出力されます。例外が発生した場所とそのハンドラの両方がトレースされます。トレースが使用可能であるかどうかのチェックは、例外が発生した場所と例外がハンドルされた場所で個別に行われます。

DBMS_TRACE.SET_PLSQL_TRACE および DBMS_TRACE.CLEAR_PLSQL_TRACE へのコールはすべて、データベースの特別なトレース・レコードに配置されます。したがって、トレース設定が変更された時点を常に確認できます。

トレース管理 収集した項目を確認するのみではなく、トレース処理を一時停止したり、再開することもできます。トレースが一時停止されてから再開されるまでの間、情報は収集されません。一時停止と再開には、定数 TRACE_PAUSE と TRACE_RESUME を使用します。トレースが一時停止または再開したことを示すためにトレース・レコードが生成されます。

定数 TRACE_LIMIT を使用すると、最新の 8,192 トレース・イベントのみを保持することができます。これにより、データベースをいっぱいにならずにトレースを繰り返すことができます。トレースの停止時には、最新の 8,192 レコードが保存されています。各トレース・レコードごとにはチェックされないため、この制限は概算です。少なくとも要求されたトレース・レコードの数は生成され、1,000 レコードまでは追加生成できます。

サブプログラムの要約

表 53-1 DBMS_TRACE パッケージのサブプログラム

サブプログラム	説明
53-6 ページの SET_PLSQL_TRACE プロシージャ	現行のセッションでトレースを開始します。
53-6 ページの CLEAR_PLSQL_TRACE プロシージャ	セッションでのトレース・データのダンプを停止します。
53-7 ページの PLSQL_TRACE_VERSION プロシージャ	トレース・パッケージのバージョン番号を取得します。

SET_PLSQL_TRACE プロシージャ

このプロシージャは、PL/SQL のトレース・データ収集を可能にします。

構文

```
DBMS_TRACE.SET_PLSQL_TRACE (  
    trace_level INTEGER);
```

パラメータ

[表 53-2](#) に、DBMS_TRACE.SET_PLSQL_TRACE 構文のパラメータを示します。

表 53-2 SET_PLSQL_TRACE プロシージャのパラメータ

パラメータ	説明
trace_level	53-2 ページ にリストされている 1 つ以上の定数を指定する必要があります。定数を合計することにより、複数の PL/SQL 言語機能のトレースを同時に使用可能にできます。制御定数 trace_pause、trace_resume および trace_stop は、他の定数と組み合わせて使用することはできません。 詳細は、53-3 ページの「 トレース・データの収集 」を参照してください。

CLEAR_PLSQL_TRACE プロシージャ

このプロシージャは、トレース・データ収集を使用禁止にします。

構文

```
DBMS_TRACE.CLEAR_PLSQL_TRACE;
```

パラメータ

なし

PLSQL_TRACE_VERSION プロシージャ

このプロシージャは、トレース・パッケージのバージョン番号を取得します。DBMS_TRACE パッケージのバージョン番号とリリース番号を戻します。

構文

```
DBMS_TRACE.PLSQL_TRACE_VERSION (  
    major OUT BINARY_INTEGER,  
    minor OUT BINARY_INTEGER);
```

パラメータ

表 53-3 に、DBMS_TRACE.PLSQL_TRACE_VERSION 構文のパラメータを示します。

表 53-3 PLSQL_TRACE_VERSION プロシージャのパラメータ

パラメータ	説明
major	DBMS_TRACE のバージョン番号
minor	DBMS_TRACE のリリース番号

DBMS_TRANSACTION

このパッケージは、ストアド・プロシージャから SQL トランザクション文へのアクセスを提供します。

関連項目：『Oracle8i SQL リファレンス』

要件

このパッケージは、パッケージ所有者 SYS ではなく、コール・ユーザーの権限で実行されま
す。

サブプログラムの要約

表 54-1 DBMS_TRANSACTION パッケージのサブプログラム

サブプログラム
54-3 ページの READ_ONLY プロシージャ
54-3 ページの READ_WRITE プロシージャ
54-3 ページの ADVISE_ROLLBACK プロシージャ
54-4 ページの ADVISE_NOTHING プロシージャ
54-4 ページの ADVISE_COMMIT プロシージャ
54-4 ページの USE_ROLLBACK_SEGMENT プロシージャ
54-5 ページの COMMIT_COMMENT プロシージャ
54-5 ページの COMMIT_FORCE プロシージャ
54-6 ページの COMMIT プロシージャ
54-6 ページの SAVEPOINT プロシージャ
54-7 ページの ROLLBACK プロシージャ
54-7 ページの ROLLBACK_SAVEPOINT プロシージャ
54-8 ページの ROLLBACK_FORCE プロシージャ
54-8 ページの BEGIN_DISCRETE_TRANSACTION プロシージャ
54-9 ページの PURGE_MIXED プロシージャ
54-10 ページの PURGE_LOST_DB_ENTRY プロシージャ
54-12 ページの LOCAL_TRANSACTION_ID ファンクション
54-12 ページの STEP_ID ファンクション

READ_ONLY プロシージャ

このプロシージャは、次の SQL 文と同じです。

```
SET TRANSACTION READ ONLY
```

構文

```
DBMS_TRANSACTION.READ_ONLY;
```

パラメータ

なし

READ_WRITE プロシージャ

このプロシージャは、次の SQL 文と同じです。

```
SET TRANSACTION READ WRITE
```

構文

```
DBMS_TRANSACTION.READ_WRITE;
```

パラメータ

なし

ADVISE_ROLLBACK プロシージャ

このプロシージャは、次の SQL 文と同じです。

```
ALTER SESSION ADVISE ROLLBACK
```

構文

```
DBMS_TRANSACTION.ADVISE_ROLLBACK;
```

パラメータ

なし

ADVISE_NOTHING プロシージャ

このプロシージャは、次の SQL 文と同じです。

```
ALTER SESSION ADVISE NOTHING
```

構文

```
DBMS_TRANSACTION.ADVISE_NOTHING;
```

パラメータ

なし

ADVISE_COMMIT プロシージャ

このプロシージャは、次の SQL 文と同じです。

```
ALTER SESSION ADVISE COMMIT
```

構文

```
DBMS_TRANSACTION.ADVISE_COMMIT;
```

パラメータ

なし

USE_ROLLBACK_SEGMENT プロシージャ

このプロシージャは、次の SQL 文と同じです。

```
SET TRANSACTION USE ROLLBACK SEGMENT <rb_seg_name>
```

構文

```
DBMS_TRANSACTION.USE_ROLLBACK_SEGMENT (
    rb_name VARCHAR2);
```

パラメータ

表 54-2 USE_ROLLBACK_SEGMENT プロシージャのパラメータ

パラメータ	説明
rb_name	使用するロールバック・セグメントの名前

COMMIT_COMMENT プロシージャ

このプロシージャは、次の SQL 文と同じです。

```
COMMIT COMMENT <text>
```

構文

```
DBMS_TRANSACTION.COMMIT_COMMENT (  
    cmnt VARCHAR2);
```

パラメータ

表 54-3 COMMIT_COMMENT プロシージャのパラメータ

パラメータ	説明
cmnt	このコミットに関連するコメント

COMMIT_FORCE プロシージャ

このプロシージャは、次の SQL 文と同じです。

```
COMMIT FORCE <text>, <number>"
```

構文

```
DBMS_TRANSACTION.COMMIT_FORCE (  
    xid VARCHAR2,  
    scn VARCHAR2 DEFAULT NULL);
```

パラメータ

表 54-4 COMMIT_FORCE プロシージャのパラメータ

パラメータ	説明
xid	ローカルまたはグローバルなトランザクション ID
scn	システム変更番号

COMMIT プロシージャ

このプロシージャは、次の SQL 文と同じです。

```
COMMIT
```

確認のためにここで記述します。このプロシージャは、PL/SQL の一部としてすでにインプリメントされています。

構文

```
DBMS_TRANSACTION.COMMIT;
```

パラメータ

なし

SAVEPOINT プロシージャ

このプロシージャは、次の SQL 文と同じです。

```
SAVEPOINT <savepoint_name>
```

確認のためにここで記述します。このプロシージャは、PL/SQL の一部としてすでにインプリメントされています。

構文

```
DBMS_TRANSACTION.SAVEPOINT (  
    savept VARCHAR2);
```

パラメータ

表 54-5 SAVEPOINT プロシージャのパラメータ

パラメータ	説明
savept	セーブポイントの識別子

ROLLBACK プロシージャ

このプロシージャは、次の SQL 文と同じです。

```
ROLLBACK
```

確認のためにここで記述します。このプロシージャは、PL/SQL の一部としてすでにインプリメントされています。

構文

```
DBMS_TRANSACTION.ROLLBACK;
```

パラメータ

なし

ROLLBACK_SAVEPOINT プロシージャ

このプロシージャは、次の SQL 文と同じです。

```
ROLLBACK TO SAVEPOINT <savepoint_name>
```

確認のためにここで記述します。このプロシージャは、PL/SQL の一部としてすでにインプリメントされています。

構文

```
DBMS_TRANSACTION.ROLLBACK_SAVEPOINT (  
    savept VARCHAR2);
```

パラメータ

表 54-6 ROLLBACK_SAVEPOINT プロシージャのパラメータ

パラメータ	説明
savept	セーブポイントの識別子

ROLLBACK_FORCE プロシージャ

このプロシージャは、次の SQL 文と同じです。

```
ROLLBACK FORCE <text>
```

構文

```
DBMS_TRANSACTION.ROLLBACK_FORCE (  
    xid VARCHAR2);
```

パラメータ

表 54-7 ROLLBACK_FORCE プロシージャのパラメータ

パラメータ	説明
xid	ローカルまたはグローバルなトランザクション ID

BEGIN_DISCRETE_TRANSACTION プロシージャ

このプロシージャは、このトランザクションについてディスクリート・トランザクション・モードを設定します。

構文

```
DBMS_TRANSACTION.BEGIN_DISCRETE_TRANSACTION;
```

パラメータ

なし

例外

表 54-8 BEGIN_DISCRETE_TRANSACTION プロシージャの例外

例外	説明
ORA-08175	トランザクションが、ディスクリート・トランザクションとして実行できない操作を行おうとしました。 この例外が発生した場合は、ロールバックしてトランザクションを再試行します。
ORA-08176	トランザクションが、ロールバック・データを生成しない操作（索引の作成、ダイレクト・ロードまたはディスクリート・トランザクション）によって変更されたデータを検出しました。 この例外が発生した場合は、例外を受け取った操作を再試行します。

例

```
DISCRETE_TRANSACTION_FAILED exception;
  pragma exception_init(DISCRETE_TRANSACTION_FAILED, -8175);
CONSISTENT_READ_FAILURE exception;
  pragma exception_init(CONSISTENT_READ_FAILURE, -8176);
```

PURGE_MIXED プロシージャ

インダウト・トランザクションにコミットまたはロールバックを強制実行すると（自動リカバリで結果を解決するのではなく）、トランザクションで混合した結果が得られる可能性があります。一部のサイトはコミットして、それ以外はロールバックします。このような矛盾は Oracle で自動的に解決できませんが、Oracle は、MIXED 列を値 'yes' に設定して、DBA_2PC_PENDING にあるエントリにフラグ付けを行います。

Oracle では、混合出力トランザクションの情報が自動的に削除されることはありません。混合トランザクションの結果として発生したすべての不整合が解決されたことを、アプリケーションまたは DBA で確認した後、このプロシージャを使用して、指定した混合出力トランザクションの情報を削除できます。

構文

```
DBMS_TRANSACTION.PURGE_MIXED (
  xid VARCHAR2);
```

パラメータ

表 54-9 PURGE_MIXED プロシージャのパラメータ

パラメータ	説明
xid	DBA_2PC_PENDING 表にある LOCAL_TRAN_ID 列の値に設定する必要があります。

PURGE_LOST_DB_ENTRY プロシージャ

コミット処理中に障害が発生すると、自動リカバリ機能によって、そのトランザクションに関連しているすべてのサイトでの結果が一貫して解決されます。ただし、リカバリの完了前にリモート・データベースが破損または再作成されると、DBA_2PC_PENDING 内のリカバリを制御するために使用するエントリと、それに関連する表が削除されずに、リカバリ処理が定期的に再試行されます。プロシージャ PURGE_LOST_DB_ENTRY では、このようなトランザクションをローカル・サイトから削除できます。

構文

```
DBMS_TRANSACTION.PURGE_LOST_DB_ENTRY (  
    xid VARCHAR2);
```

警告： PURGE_LOST_DB_ENTRY は、他のデータベースが失われたり再作成された場合のみ使用してください。他の目的で使用すると、他のデータベースをリカバリ不能にしたり一貫性のない状態にする可能性があります。

自動リカバリの実行前に、トランザクションは、DBA_2PC_PENDING を "collecting"、"committed" または "prepared" の状態で表示できます。DBA が "commit force" または "rollback force" を使用して、インダウト・トランザクションに特定の結果を強制した場合は、"forced commit" または "forced rollback" の状態も表示できます。自動リカバリは通常、このような状態のエントリを削除します。唯一の例外は、トランザクション内の他のサイトとの間で整合性がとられていない状態の強制トランザクションがリカバリ処理で見つかった場合です。この場合、エントリは表内に残り、MIXED 列の値が 'yes' になります。

ただし、状態によっては、自動リカバリの実行が不可能な場合があります。たとえば、リモート・データベースが完全に破損した場合です。たとえ再作成しても、新規のデータベース ID を取得するので、リカバリ処理では識別できません（起こり得るエラーは、ORA-02062）。このような場合、DBA はプロシージャ PURGE_LOST_DB_ENTRY を使用して、"prepared" 以外の状態にあるエントリをクリーン・アップできます。これらのエントリはデータベース・リソースを保持していないため、DBA は特に急いでこれらのエントリを解決する必要はありません。

次の表は、トランザクションについての様々な状態の内容および DBA に必要とされるアクションを示します。

表 54-10 PURGE_LOST_DB_ENTRY プロシージャの状態

列の状態	グローバル・ トランザク ションの状態	ローカル・ トランザク ションの状態	通常の DBA アクション	代替の DBA アクション
Collecting	ロールバック	ロールバック	なし	PURGE_LOST_DB_ENTRY (1)
Committed	コミット済	コミット済	なし	PURGE_LOST_DB_ENTRY (1)
Prepared	不明	準備済	なし	FORCE COMMIT または ROLLBACK
Forced commit	不明	コミット済	なし	PURGE_LOST_DB_ENTRY (1)
Forced rollback	不明	ロールバック	なし	PURGE_LOST_DB_ENTRY (1)
Forced commit (mixed)	混合	コミット済	(2)	
Forced rollback (mixed)	混合	ロールバック	(2)	

注意 1: 重要な再構成が発生して自動リカバリではトランザクションを解決できない場合のみ使用します。例としては、リモート・データベース全体が破損し、ソフトウェアでの再構成によって 2 フェーズ・コミット機能を失う結果になった場合や TP モニターのような外部トランザクション・コーディネータからの情報を失った場合などがあります。

注意 2: 不整合を取り除くために調査したり手動で作業を行い、プロシージャ PURGE_MIXED を使用します。

パラメータ

表 54-11 PURGE_LOST_DB_ENTRY プロシージャのパラメータ

パラメータ	説明
xid	DBA_2PC_PENDING 表にある LOCAL_TRAN_ID 列の値に設定する必要があります。

LOCAL_TRANSACTION_ID ファンクション

このファンクションは、現行のトランザクションについてローカルな（インスタンスに対して）一意の識別子を戻します。現行のトランザクションがない場合は、NULL を戻します。

構文

```
DEMS_TRANSACTION.LOCAL_TRANSACTION_ID (  
    create_transaction BOOLEAN := FALSE)  
RETURN VARCHAR2;
```

パラメータ

表 54-12 LOCAL_TRANSACTION_ID ファンクションのパラメータ

パラメータ	説明
create_transaction	TRUE に設定すると、トランザクションが現在アクティブでない場合は、トランザクションを開始します。

STEP_ID ファンクション

このファンクションは、トランザクションの DML 操作を順序付ける、ローカルで（ローカル・トランザクションに対して）一意の正の整数を戻します。

構文

```
DEMS_TRANSACTION.STEP_ID  
RETURN NUMBER;
```

パラメータ

なし

このパッケージは、トランスポート可能なセットが自己完結型かどうかをチェックします。すべての違反が、ビュー `TRANSPORT_SET_VIOLATIONS` から選択できる一時表に挿入されます。

このプロシージャは、`execute_catalog_role` を与えられているユーザーのみ実行できます。このロールは、初期段階ではユーザー `SYS` にのみ割り当てられています。

関連項目：『Oracle8i 管理者ガイド』および『Oracle8i 移行ガイド』

例外

```
ts_not_found EXCEPTION;
PRAGMA exception_init(ts_not_found, -29304);
ts_not_found_num NUMBER := -29304;

invalid_ts_list EXCEPTION;
PRAGMA exception_init(invalid_ts_list, -29346);
invalid_ts_list_num NUMBER := -29346;

sys_or_tmp_ts EXCEPTION;
PRAGMA exception_init(sys_or_tmp_ts, -29351);
sys_or_tmp_ts_num NUMBER := -29351;
```

サブプログラムの要約

この2つのプロシージャは、データベース管理者がコールするように設計されています。

表 55-1 DBMS_TTS パッケージのサブプログラム

サブプログラム	説明
55-2 ページの TRANSPORT_SET_CHECK プロシージャ	表領域（トランスポート可能）のセットが自己完結型かどうかをチェックします。
55-3 ページの DOWNGRADE プロシージャ	データに関連するトランスポート可能な表領域をダウングレードします。

TRANSPORT_SET_CHECK プロシージャ

このプロシージャは、表領域（トランスポート可能）のセットが自己完結型かどうかをチェックします。このプロシージャのコール後、ユーザーはビューから選択して、違反があればそのリストを調べることができます。ビューで行が戻らない場合、表領域のセットは自己完結型です。次に例を示します。

```
SQLPLUS> EXECUTE TRANSPORT_SET_CHECK('foo,bar', TRUE);
SQLPLUS> SELECT * FROM TRANSPORT_SET_VIOLATIONS;
```

構文

```
DBMS_TTS.TRANSPORT_SET_CHECK (
    ts_list          IN VARCHAR2,
    incl_constraints IN BOOLEAN);
```

パラメータ

表 55-2 TRANSPORT_SET_CHECK プロシージャのパラメータ

パラメータ	説明
ts_list	表領域のカンマで区切られたリスト。
incl_constraints	表領域が自己完結型かどうかを調べるときに、参照整合性制約を考慮する場合は TRUE を設定します。

DOWNGRADE プロシージャ

このプロシージャは、データに関連するトランスポート可能な表領域をダウングレードします。

構文

```
DBMS_TTS.DOWNGRADE;
```

パラメータ

なし

DBMS_UTILITY

このパッケージは、各種のユーティリティ・サブプログラムを提供します。

DBMS_UTILITY は、各パーティションに対してジョブを実行します。INIT.ORA パラメータの JOB_QUEUE_PROCESSES を正しく設定して、同時実行ジョブの数を制御するのは、ユーザーの責任です。正しい構文についての最小限度のエラー・チェックがあります。すべてのエラーは、SNP トレース・ファイルにレポートされます。

要件

DBMS_UTILITY は、NAME_RESOLVE、COMPILE_SCHEMA および ANALYZE_SCHEMA プロシージャに対するコール・ユーザーの権限で実行されます。これは、SQL の正しい動作のために必要です。

このパッケージは、SYS として実行されません。権限は、DBMS_DDL を介してチェックされます。

型

type uncl_array IS TABLE OF VARCHAR2(227) INDEX BY BINARY_INTEGER;
"USER"."NAME"."COLUMN"@LINK のリストは、ここに格納されます。

type name_array IS TABLE OF VARCHAR2(30) INDEX BY BINARY_INTEGER;
NAME のリストは、ここに格納されます。

type dblink_array IS TABLE OF VARCHAR2(128) INDEX BY BINARY_INTEGER;
データベース・リンクのリストは、ここに格納されます。

TYPE index_table_type IS TABLE OF BINARY_INTEGER INDEX BY BINARY_INTEGER;
オブジェクトの生成順序は、ここに戻されます。

TYPE number_array IS TABLE OF NUMBER INDEX BY BINARY_INTEGER;
ユーザーのためのオブジェクトの生成順序は、ここに戻されます。

TYPE instance_record IS RECORD (
 inst_number NUMBER,
 inst_name VARCHAR2(60));
TYPE instance_table IS TABLE OF instance_record INDEX BY BINARY_INTEGER;
アクティブなインスタンス番号とインスタンス名のリスト。

instance_table の索引の開始は 1 で、instance_table は稠密です。

サブプログラムの要約

表 56-1 DBMS_UTILITY パッケージのサブプログラム

サブプログラム	説明
56-4 ページの COMPILE_SCHEMA プロシージャ	指定したスキーマ内にあるすべてのプロシージャ、ファンクション、パッケージおよびトリガーをコンパイルします。
56-5 ページの ANALYZE_SCHEMA プロシージャ	スキーマ内にあるすべての表、クラスタおよび索引を分析します。
56-6 ページの ANALYZE_DATABASE プロシージャ	データベース内にあるすべての表、クラスタおよび索引を分析します。

表 56-1 DBMS_UTILITY パッケージのサブプログラム

サブプログラム	説明
56-7 ページの FORMAT_ERROR_STACK ファンクション	現行のエラー・スタックをフォーマットします。
56-7 ページの FORMAT_CALL_STACK ファンクション	現行のコール・スタックをフォーマットします。
56-8 ページの IS_PARALLEL_SERVER ファンクション	このデータベースがパラレル・サーバー・モードで実行しているかどうかを検出します。
56-8 ページの GET_TIME ファンクション	現在の時間を 100 分の 1 秒単位で検出します。
56-9 ページの GET_PARAMETER_VALUE ファンクション	指定した init.ora パラメータの値を取得します。
56-10 ページの NAME_RESOLVE プロシージャ	指定した名前を解決します。
56-12 ページの NAME_TOKENIZE プロシージャ	指定した名前を解析するために解析機能をコールします。
56-12 ページの COMMA_TO_TABLE プロシージャ	名前のカンマで区切られたリストを、PL/SQL 名前表に変換します。
56-13 ページの TABLE_TO_COMMA プロシージャ	PL/SQL 名前表を、名前のカンマで区切られたリストに変換します。
56-14 ページの PORT_STRING ファンクション	Oracle とオペレーティング・システムのバージョンを一意に識別する文字列を戻します。
56-14 ページの DB_VERSION プロシージャ	データベースに関するバージョン情報を戻します。
56-15 ページの MAKE_DATA_BLOCK_ADDRESS ファンクション	ファイル番号とブロック番号を指定して、データ・ブロック・アドレスを作成します。
56-16 ページの DATA_BLOCK_ADDRESS_FILE ファンクション	データ・ブロック・アドレスのファイル番号部分を取得します。
56-16 ページの DATA_BLOCK_ADDRESS_BLOCK ファンクション	データ・ブロック・アドレスのブロック番号部分を取得します。
56-17 ページの GET_HASH_VALUE ファンクション	指定した文字列についてハッシュ値を計算します。
56-18 ページの ANALYZE_PART_OBJECT プロシージャ	

表 56-1 DBMS_UTILITY パッケージのサブプログラム

サブプログラム	説明
56-19 ページの EXEC_DDL_STATEMENT プロシージャ	parse_string で DDL 文を実行します。
56-19 ページの CURRENT_INSTANCE ファンクション	現在接続しているインスタンス番号を戻します。
56-20 ページの ACTIVE_INSTANCES プロシージャ	

COMPILE_SCHEMA プロシージャ

このプロシージャは、指定したスキーマ内にあるすべてのプロシージャ、ファンクション、パッケージおよびトリガーをコンパイルします。このプロシージャのコール後、ステータスが INVALID の項目をビュー ALL_OBJECTS から選択して、すべてのオブジェクトが正常にコンパイルされたかどうかを調べます。

Enterprise Manager のコマンドを使用すると、INVALID のオブジェクトに関連するエラーを表示できます。

```
SHOW ERRORS <type> <schema>.<name>
```

構文

```
DBMS_UTILITY.COMPILE_SCHEMA (  
    schema VARCHAR2);
```

パラメータ

表 56-2 COMPILE_SCHEMA プロシージャのパラメータ

パラメータ	説明
schema	スキーマの名前

例外

表 56-3 COMPILE_SCHEMA プロシージャの例外

例外	説明
ORA-20000	このスキーマ内のいずれかのオブジェクトに対して権限が不十分です。

ANALYZE_SCHEMA プロシージャ

このプロシージャは、スキーマ内にあるすべての表、クラスタおよび索引を分析します。

構文

```
DBMS_UTILITY.ANALYZE_SCHEMA (  
  schema          VARCHAR2,  
  method          VARCHAR2,  
  estimate_rows   NUMBER   DEFAULT NULL,  
  estimate_percent NUMBER   DEFAULT NULL,  
  method_opt      VARCHAR2 DEFAULT NULL);
```

パラメータ

表 56-4 ANALYZE_SCHEMA プロシージャのパラメータ

パラメータ	説明
schema	スキーマの名前。
method	ESTIMATE、COMPUTE または DELETE のいずれかを設定します。 ESTIMATE を設定した場合は、estimate_rows または estimate_percent のいずれかを 0（ゼロ）以外に設定する必要があります。
estimate_rows	推定する行数。
estimate_percent	推定する行のパーセント。 estimate_rows が指定されている場合、このパラメータは無視されます。
method_opt	次の書式の分析方法オプション。 [FOR TABLE] [FOR ALL [INDEXED] COLUMNS] [SIZE n] [FOR ALL INDEXES]

例外

表 56-5 ANALYZE_SCHEMA プロシージャの例外

例外	説明
ORA-20000	このスキーマ内のいずれかのオブジェクトに対して権限が不十分です。

ANALYZE_DATABASE プロシージャ

このプロシージャは、データベース内にあるすべての表、クラスタおよび索引を分析します。

構文

```
DBMS_UTILITY.ANALYZE_DATABASE (
    method          VARCHAR2,
    estimate_rows    NUMBER    DEFAULT NULL,
    estimate_percent NUMBER    DEFAULT NULL,
    method_opt       VARCHAR2 DEFAULT NULL);
```

パラメータ

表 56-6 ANALYZE_DATABASE プロシージャのパラメータ

パラメータ	説明
method	ESTIMATE、COMPUTE または DELETE のいずれかを設定します。 ESTIMATE を設定した場合は、estimate_rows または estimate_percent のいずれかを 0（ゼロ）以外に設定する必要があります。
estimate_rows	推定する行数。
estimate_percent	推定する行のパーセント。 estimate_rows が指定されている場合、このパラメータは無視されます。
method_opt	次の書式の分析方法オプション。 [FOR TABLE] [FOR ALL [INDEXED] COLUMNS] [SIZE n] [FOR ALL INDEXES]

例外

表 56-7 ANALYZE_DATABASE プロシージャの例外

例外	説明
ORA-20000	このデータベース内のいずれかのオブジェクトに対して権限が不十分です。

FORMAT_ERROR_STACK ファンクション

このファンクションは、現行のエラー・スタックをフォーマットします。このファンクションは、全エラー・スタックを表示するための例外ハンドラで使用できます。

構文

```
DBMS_UTILITY.FORMAT_ERROR_STACK  
RETURN VARCHAR2;
```

パラメータ

なし

戻り値

最大 2000 バイトまでのエラー・スタックを戻します。

FORMAT_CALL_STACK ファンクション

このファンクションは、現行のコール・スタックをフォーマットします。このファンクションは、コール・スタックにアクセスするための任意ストアド・プロシージャまたはトリガーで使用できます。このファンクションは、デバッグ時に役立ちます。

構文

```
DBMS_UTILITY.FORMAT_CALL_STACK  
RETURN VARCHAR2;
```

パラメータ

なし

プラグマ

```
pragma restrict_references(format_call_stack,WNDS);
```

戻り値

最大 2000 バイトまでのコール・スタックを戻します。

IS_PARALLEL_SERVER ファンクション

このファンクションは、このデータベースがパラレル・サーバー・モードで実行しているかどうかを検出します。

構文

```
DBMS_UTILITY.IS_PARALLEL_SERVER  
    RETURN BOOLEAN;
```

パラメータ

なし

戻り値

インスタンスがパラレル・サーバー・モードで起動されている場合は TRUE を戻し、そうでない場合は FALSE を戻します。

GET_TIME ファンクション

このファンクションは、現在の時間を 100 分の 1 秒単位で検出します。これは、主に経過時間を判断するのに役立ちます。

構文

```
DBMS_UTILITY.GET_TIME  
    RETURN NUMBER;
```

パラメータ

なし

戻り値

任意の時点からの時間を 100 分の 1 秒の数で戻します。

GET_PARAMETER_VALUE ファンクション

このファンクションは、指定した `init.ora` パラメータの値を取得します。

構文

```
DBMS_UTILITY.GET_PARAMETER_VALUE (  
    parnam IN      VARCHAR2,  
    intval IN OUT  BINARY_INTEGER,  
    strval IN OUT  VARCHAR2)  
RETURN BINARY_INTEGER;
```

パラメータ

表 56-8 GET_PARAMETER_VALUE ファンクションのパラメータ

パラメータ	説明
parnam	パラメータ名
intval	整数パラメータの値、または文字列パラメータの値の長さ
strval	文字列パラメータの値

戻り値

表 56-9 GET_PARAMETER_VALUE ファンクションの戻り値

戻り値	説明
partyp	パラメータ型 パラメータが整数またはブール・パラメータの場合は 0 パラメータが文字列またはファイル・パラメータの場合は 1

例

```
DECLARE  
    parnam VARCHAR2(256);  
    intval BINARY_INTEGER;  
    strval VARCHAR2(256);  
    partyp BINARY_INTEGER;  
BEGIN  
    partyp := dbms_utility.get_parameter_value('max_dump_file_size',  
                                              intval, strval);  
    dbms_output.put('parameter value is: ');
```

```
IF partyp = 1 THEN
    dbms_output.put_line(strval);
ELSE
    dbms_output.put_line(intval);
END IF;
IF partyp = 1 THEN
    dbms_output.put('parameter value length is: ');
    dbms_output.put_line(intval);
END IF;
dbms_output.put('parameter type is: ');
IF partyp = 1 THEN
    dbms_output.put_line('string');
ELSE
    dbms_output.put_line('integer');
END IF;
END;
```

NAME_RESOLVE プロシージャ

このプロシージャは、指定した名前を解決し、必要に応じてシノニム変換と認可チェックが含まれます。

構文

```
DBMS_UTILITY.NAME_RESOLVE (
    name          IN  VARCHAR2,
    context       IN  NUMBER,
    schema        OUT VARCHAR2,
    part1         OUT VARCHAR2,
    part2         OUT VARCHAR2,
    dblink        OUT VARCHAR2,
    part1_type    OUT NUMBER,
    object_number OUT NUMBER);
```


パラメータ

表 56-10 NAME_RESOLVE プロシージャのパラメータ

パラメータ	説明
name	<p>オブジェクト名。</p> <p>この名前はフォーム [[a.]b.]c[@d] で指定します。a、b、c は SQL 識別子、d は DB リンクです。DB リンクでは、構文チェックは実行されません。DB リンクが指定されている場合や名前が DB リンクの一部に変換される場合、オブジェクトは解決されませんが、schema、part1、part2 および dblink OUT の各パラメータは入力されます。</p> <p>a、b および c は、デリミタ付き識別子の場合があり、NLS 文字（シングルおよびマルチバイト）を含んでいる場合があります。</p>
context	0 ～ 8 の範囲の整数を指定します。
schema	オブジェクト C のスキーマ。name パラメータにスキーマが指定されていない場合、schema は、名前を解決することによって決定されます。
part1	名前の最初の部分。この名前の型は、part1_type に指定されます（シノニム、プロシージャまたはパッケージ）。
part2	このパラメータが NULL 以外の場合は、part1 で示されるパッケージ内のプロシージャ名です。
dblink	このパラメータが NULL 以外の場合、データベース・リンクは、name の一部として指定されたか、または name がデータベース・リンクの一部に変換されるシノニムとして指定されたかのいずれかです。後者の場合、part1_type はシノニムを示します。
part1_type	<p>part1 の型は、次のとおりです。</p> <p>5 シノニム</p> <p>7 プロシージャ（最上位レベル）</p> <p>8 ファンクション（最上位レベル）</p> <p>9 パッケージ</p> <p>シノニムの場合、name は、データベース・リンクの一部に変換するシノニムです。名前変換がさらに必要な場合は、このリモート・ノードで DBMS_UTILITY.NAME_RESOLVE プロシージャをコールする必要があります。</p>
object_name	このパラメータが NULL でなければ name の解決に成功しました。このパラメータの値は解決したオブジェクト数です。

例外

すべてのエラーは、例外を呼び出すことによって処理されます。オブジェクト名の指定時に起こり得る各種の構文エラーに基づいて、広範囲にわたる例外が用意されています。

NAME_TOKENIZE プロシージャ

このプロシージャは、解析機能をコールして、"a [. b [. c]][@ dblink]" と指定した名前を解析します。二重引用符を削除するか、または引用符がない場合は大文字に変換します。ソートに関するすべてのコメントは無視し、意味的な分析は行いません。不明な値は NULL のまま残ります。

構文

```
DBMS_UTILITY.NAME_TOKENIZE (  
    name      IN  VARCHAR2,  
    a         OUT VARCHAR2,  
    b         OUT VARCHAR2,  
    c         OUT VARCHAR2,  
    dblink    OUT VARCHAR2,  
    nextpos   OUT BINARY_INTEGER);
```

パラメータ

各 a、b、c および dblink は、それぞれの anext、bnext、cnext、dnext の次のトークンが開始される場所を示します。

COMMA_TO_TABLE プロシージャ

このプロシージャは、名前のカンマで区切られたリストを、PL/SQL 名前表に変換します。このプロシージャは、NAME_TOKENIZE を使用して、名前とカンマを区別します。

構文

```
DBMS_UTILITY.COMMA_TO_TABLE (  
    list      IN  VARCHAR2,  
    tablen    OUT BINARY_INTEGER,  
    tab       OUT UNCL_ARRAY);
```

パラメータ

表 56-11 COMMA_TO_TABLE プロシージャのパラメータ

パラメータ	説明
list	表のカンマで区切られたリスト
tablen	PL/SQL 表にある表の数
tab	表名のリストを含んだ PL/SQL 表

戻り値

PL/SQL 表が、値 1..n と n+1 is null とともに戻されます。

使用上の注意

list は、空ではないカンマで区切られたリストであることが必要です。カンマで区切られたリスト以外の場合は拒否されます。二重引用符内のカンマは無視されます。

カンマで区切られたリストに、ハイフン (-) などのマルチバイト・キャラクタを含めることはできません。

tab にある値は、変換されずに元のリストからカットされます。

TABLE_TO_COMMA プロシージャ

このプロシージャは、PL/SQL 名前表を、名前のカンマで区切られたリストに変換します。PL/SQL 表を 1..n に変換して n+1 null で終了します。

構文

```
DBMS_UTILITY.TABLE_TO_COMMA (  
    tab    IN  UNCL_ARRAY,  
    tablen OUT BINARY_INTEGER,  
    list   OUT VARCHAR2);
```

パラメータ

表 56-12 TABLE_TO_COMMA プロシージャのパラメータ

パラメータ	説明
tab	表名のリストを含んだ PL/SQL 表
tablen	PL/SQL 表にある表の数

表 56-12 TABLE_TO_COMMA プロシージャのパラメータ

パラメータ	説明
list	表のカンマで区切られたリスト

戻り値

カンマで区切られたリストと表内で検出された要素の数が戻されます。

PORT_STRING ファンクション

このファンクションは、オペレーティング・システムと、TWO TASK PROTOCOL バージョンのデータベースを識別する文字列を戻します。たとえば、"VAX/VMX-7.1.0.0" が戻ります。

最大長はポート固有の長さです。

構文

```
DBMS_UTILITY.PORT_STRING
RETURN VARCHAR2;
```

パラメータ

なし

プラグマ

```
pragma restrict_references(port_string, WNDS, RNDS, WNPS, RNPS);
```

DB_VERSION プロシージャ

このプロシージャは、データベースに関するバージョン情報を戻します。

構文

```
DBMS_UTILITY.DB_VERSION (
    version          OUT VARCHAR2,
    compatibility OUT VARCHAR2);
```

パラメータ

表 56-13 DB_VERSION プロシージャのパラメータ

パラメータ	説明
version	データベースの内部ソフトウェア・バージョンを表す文字列（例：7.1.0.0.0）。 この文字列の長さは可変なので、データベース・バージョンによって決定されます。
compatibility	互換性がある init.ora パラメータによって決定する、データベースの互換性設定。 このパラメータが init.ora ファイルで指定されていない場合は、NULL が戻されます。

MAKE_DATA_BLOCK_ADDRESS ファンクション

このファンクションは、ファイル番号とブロック番号を指定したデータ・ブロック・アドレスを作成します。データ・ブロック・アドレスは、データベース内のブロックを識別するために使用する内部構造です。このファンクションは、データ・ブロック・アドレスを含んだ特定の固定表にアクセスするとき役立ちます。

構文

```
DBMS_UTILITY.MAKE_DATA_BLOCK_ADDRESS (  
    file NUMBER,  
    block NUMBER)  
RETURN NUMBER;
```

パラメータ

表 56-14 MAKE_DATA_BLOCK_ADDRESS ファンクションのパラメータ

パラメータ	説明
file	ブロックを含むファイル
block	ブロック増分値に基づくファイル内でのブロックのオフセット

プラグマ

```
pragma restrict_references (make_data_block_address, WNDS, RNDS, WNPS, RNPS);
```

戻り値

表 56-15 MAKE_DATA_BLOCK_ADDRESS ファンクションの戻り値

戻り値	説明
dba	データ・ブロック・アドレス

DATA_BLOCK_ADDRESS_FILE ファンクション

このファンクションは、データ・ブロック・アドレスのファイル番号部分を取得します。

構文

```
DBMS_UTILITY.DATA_BLOCK_ADDRESS_FILE (  
    dba NUMBER)  
RETURN NUMBER;
```

パラメータ

表 56-16 DATA_BLOCK_ADDRESS_FILE ファンクションのパラメータ

パラメータ	説明
dba	データ・ブロック・アドレス

プラグマ

```
pragma restrict_references(data_block_address_file, WNDS, RNDS, WNPS, RNPS);
```

戻り値

表 56-17 DATA_BLOCK_ADDRESS_FILE ファンクションの戻り値

戻り値	説明
file	ブロックを含むファイル

DATA_BLOCK_ADDRESS_BLOCK ファンクション

このファンクションは、データ・ブロック・アドレスのブロック番号部分を取得します。

構文

```
DBMS_UTILITY.DATA_BLOCK_ADDRESS_BLOCK (  
    dba NUMBER)  
RETURN NUMBER;
```

パラメータ

表 56-18 DATA_BLOCK_ADDRESS_BLOCK ファンクションのパラメータ

パラメータ	説明
dba	データ・ブロック・アドレス

プラグマ

```
pragma restrict_references(data_block_address_block, WNDS, RNDS, WNPS, RNPS);
```

戻り値

表 56-19 DATA_BLOCK_ADDRESS_BLOCK ファンクションの戻り値

戻り値	説明
block	ブロックのブロック・オフセット

GET_HASH_VALUE ファンクション

このファンクションは、指定した文字列についてハッシュ値を計算します。

構文

```
DBMS_UTILITY.GET_HASH_VALUE (  
    name      VARCHAR2,  
    base      NUMBER,  
    hash_size NUMBER)  
RETURN NUMBER;
```

パラメータ

表 56-20 GET_HASH_VALUE ファンクションのパラメータ

パラメータ	説明
name	ハッシュする文字列
base	戻されるハッシュ値が始まるベース値
hash_size	必要とするハッシュ表のサイズ

プラグマ

```
pragma restrict_references(get_hash_value, WNDS, RNDS, WNPS, RNPS);
```

戻り値

ハッシュ値は、入力文字列に基づいています。たとえば、ハッシュ値が 1000 ～ 3047 の範囲にある文字列についてハッシュ値を取得するには、ベース値として 1000、hash_size 値として 2048 を使用します。hash_size パラメータは、2 の累乗を使用すると動作が最適になります。

ANALYZE_PART_OBJECT プロシージャ

このプロシージャは、次の SQL と同じです。

```
"ANALYZE TABLE|INDEX [<schema>.<object_name> PARTITION <pname> [<command_type>]
[<command_opt>] [<sample_clause>]
```

オブジェクトの各パーティションについて、ジョブ・キューを使用して並列に実行します。

構文

```
DBMS_UTILITY.ANALYZE_PART_OBJECT (
    schema          IN VARCHAR2 DEFAULT NULL,
    object_name     IN VARCHAR2 DEFAULT NULL,
    object_type     IN CHAR      DEFAULT 'T',
    command_type    IN CHAR      DEFAULT 'E',
    command_opt     IN VARCHAR2  DEFAULT NULL,
    sample_clause   IN VARCHAR2  DEFAULT 'SAMPLE 5 PERCENT');
```

パラメータ

表 56-21 ANALYZE_PART_OBJECT プロシージャのパラメータ

パラメータ	説明
schema	object_name のスキーマ。
object_name	分析するオブジェクトの名前。パーティション化されている必要があります。
object_type	オブジェクトの型。T（表）または I（索引）である必要があります。
command_type	次のいずれかになります。 C（統計情報の計算） E（統計情報の推定） D（統計情報の削除） V（構造の検証）

表 56-21 ANALYZE_PART_OBJECT プロシージャのパラメータ

パラメータ	説明
command_opt	command_type のその他のオプション。 C と E については、FOR 表、FOR のすべての LOCAL 索引、FOR のすべての列、または分析統計（表）の 'FOR' オプションをいくつか組み合わせることが可能です。V については、object_type が T のときに CASCADE が可能です。
sample_clause	command_type が 'E' の場合に使用するサンプル句。

EXEC_DDL_STATEMENT プロシージャ

このプロシージャは、parse_string で DDL 文を実行します。

構文

```
DBMS_UTILITY.EXEC_DDL_STATEMENT (  
    parse_string IN VARCHAR2);
```

パラメータ

表 56-22 EXEC_DDL_STATEMENT プロシージャのパラメータ

パラメータ	説明
parse_string	実行する DDL 文

CURRENT_INSTANCE ファンクション

このファンクションは、現在接続しているインスタンス番号を戻します。接続しているインスタンスが切断されると、NULL を戻します。

構文

```
DBMS_UTILITY.CURRENT_INSTANCE  
    RETURN NUMBER;
```

パラメータ

なし

ACTIVE_INSTANCES プロシージャ

構文

```
DEMS_UTILITY.ACTIVE_INSTANCE (  
    instance_table    OUT INSTANCE_TABLE,  
    instance_count    OUT NUMBER);
```

パラメータ

表 56-23 ACTIVE_INSTANCES プロシージャのパラメータ

プロシージャ	説明
instance_table	アクティブなインスタンス番号と名前のリストが含まれます。進行中のインスタンスがない場合（または OPS 以外の設定の場合）、リストは空になります。
instance_count	アクティブなインスタンスの数。OPS 以外の設定の場合は 0 になります。

DEBUG_EXTPROC

DEBUG_EXTPROC パッケージによって、セッション内の extproc エージェントを起動できます。このユーティリティ・パッケージは、外部プロシージャをデバッグするのに役立ちます。

要件

Oracle アカウントに、パッケージに対する EXECUTE 権限と、CREATE LIBRARY 権限が必要です。

注意： DEBUG_EXTPROC は、実行プロセスに連結できるデバッガを使用して、プラットフォームでのみ稼動します。

インストール時の注意

パッケージをインストールするためには、スクリプト DBGEXTP.SQL を実行します。

- 'extproc' プロセスをデバッグする Oracle USER に、このパッケージをインストールまたはロードします。
- DEBUG_EXTPROC パッケージに対する EXECUTE 権限があることを確認します。

```
SELECT SUBSTR(OBJECT_NAME, 1, 20)
FROM USER_OBJECTS
WHERE OBJECT_NAME = 'DEBUG_EXTPROC';
```
- パッケージに対して EXECUTE 権限がある場合は、他のユーザーとしてこのパッケージをインストールできます。

DEBUG_EXTPROC の使用方法

使用の前提条件

外部プロシージャ 'extproc' エージェントを起動するためには、リスナーが適切に構成されていることが前提です。

また、デバッグ処理を支援するために、デバッグ記号が付いた共有ライブラリを作成していることも前提です。C コンパイラのマニュアルをチェックして、適切な C コンパイラ・スイッチで、デバッグ記号が付いた共有ライブラリを作成してください。

使用上の注意

- Oracle に接続して、SQL*Plus または OCI プログラムから新規の Oracle セッションを起動します。
- プロシージャ DEBUG_EXTPROC.STARTUP_EXTPROC_AGENT を実行して、このセッションで extproc エージェントを起動し、たとえば、DEBUG_EXTPROC.STARTUP_EXTPROC_AGENT を実行します。extproc エージェントが終了してしまうので、このセッションは終了しないでください。
- このセッションで起動した extproc エージェントの PID を判別します。

- デバッガ（たとえば、gdb、dbx またはシステム固有なデバッガ）を使用して、extproc 実行ファイルをロードし、実行プロセスに連結します。
- ファンクション 'pextproc' にブレーク・ポイントを設定し、デバッガが実行を継続できるようにします。
- 最初に `DEBUG_EXTPROC.STARTUP_EXTPROC_AGENT` を実行した同じセッションで、外部プロシーダを実行します。
- デバッガがファンクション 'pextproc' でブレークします。この時点で、PL/SQL 外部ファンクションで参照する共有ライブラリがロードされ、ファンクションが解決されます。C ファンクションにブレーク・ポイントを設定し、デバッガが実行を継続できるようにします。

PL/SQL は、実行時に共有ライブラリをロードするため、使用するデバッガは、共有ライブラリから新規の記号を自動的に追跡管理できる場合とできない場合があります。デバッガ・コマンドをいくつか発行して、記号をロードできます（たとえば、gdb 内の 'share'）。

- デバッガは、C ファンクションでブレークします。デバッグ記号が付いた共有ライブラリを作成しておくことが前提です。
- デバッグを続行します。

サブプログラムの要約

`DEBUG_EXTPROC` には、サブプログラムの 1 つである `STARTUP_EXTPROC_AGENT` プロシーダが含まれています。これにより、セッション内で extproc エージェント・プロセスを起動します。

STARTUP_EXTPROC_AGENT プロシーダ

このプロシーダは、セッションで extproc エージェント・プロセスを起動します。これにより、実行プロセスの PID を取得できます。この PID は、デバッガを使用して実行プロセスに連結するために必要です。

構文

```
DEBUG_EXTPROC.STARTUP_EXTPROC_AGENT;
```

パラメータ

なし

OUTLN_PKG パッケージには、ストアド・アウトラインの管理に関連するサブプログラムのための機能インタフェースが含まれています。

ストアド・アウトラインは、指定した SQL 文についての実行プランに関連するストアド・データです。これによって、オプティマイザは、最初にアウトラインとともに生成されたプランと同じ実行プランを繰り返し再作成できます。アウトラインに格納されたデータの一部分は、プランの安定性を保つために使用するヒントで構成されています。

要件

OUTLN_PKG には、適切なユーザーのみ使用できる管理手順が含まれています。EXECUTE 権限は、DBA が明示的に定めない限り、一般ユーザー・コミュニティにまで拡張されません。

セキュリティ

アウトラインの管理目的に使用可能な PL/SQL ファンクションは、そのプロシージャ（またはパッケージ）に対する EXECUTE 権限があるユーザーのみが実行できます。

サブプログラムの要約

表 58-1 OUTLN_PKG パッケージのサブプログラム

サブプログラム	説明
58-2 ページの DROP_UNUSED プロシージャ	作成後に使用されなくなったアウトラインをすべて削除します。
58-3 ページの DROP_BY_CAT プロシージャ	特定のカテゴリに属しているすべてのアウトラインを削除します。
58-3 ページの UPDATE_BY_CAT プロシージャ	あるカテゴリにあるすべてのアウトラインのカテゴリを、新規のカテゴリに変更します。

DROP_UNUSED プロシージャ

このプロシージャは、SQL 文のコンパイルで適用されなくなったアウトラインを削除します。

構文

```
OUTLN_PKG.DROP_UNUSED;
```

パラメータ

なし

使用上の注意

動的 SQL のかわりに作成され、1 回のみ使用するためにアプリケーションで生成されたアウトラインに DROP_UNUSED を使用することがあります。そのような文では、アウトラインは使用されず、貴重なディスク領域を単に占有します。

DROP_BY_CAT プロシージャ

このプロシージャは、特定のカテゴリに属しているすべてのアウトラインを削除します。

構文

```
OUTLN_PKG.DROP_BY_CAT (  
    cat VARCHAR2);
```

パラメータ

表 58-2 DROP_BY_CAT プロシージャのパラメータ

パラメータ	説明
cat	削除するアウトラインのカテゴリ

使用上の注意

アウトラインのカテゴリで、時々パージが必要になることがあります。このプロシージャは、それを 1 回のコールで実行します。

例

この例では、DEFAULT カテゴリ内にあるすべてのアウトラインを削除します。

```
OUTLN_PKG.DROP_BY_CAT('DEFAULT');
```

UPDATE_BY_CAT プロシージャ

このプロシージャは、あるカテゴリにあるすべてのアウトラインのカテゴリを、新規のカテゴリに変更します。アウトライン内の SQL テキストがターゲット・カテゴリ内ですでにアウトラインを持っている場合は、新規カテゴリにマージされません。

構文

```
OUTLN_PKG.UPDATE_BY_CAT (  
    oldcat VARCHAR2 DEFAULT 'DEFAULT',  
    newcat VARCHAR2 DEFAULT 'DEFAULT');
```

パラメータ

表 58-3 UPDATE_BY_CAT プロシージャのパラメータ

パラメータ	説明
oldcat	変更する現行のカテゴリ
newcat	アウトラインを変更するターゲット・カテゴリ

使用上の注意

アウトラインのセットが希望通りならば、アウトラインを実験的なカテゴリから本番カテゴリに移動することが選択できます。同様に、あるカテゴリから別の既存のカテゴリに、アウトラインのセットをマージすることもできます。

例

次の例では、DEFAULT カテゴリ内のすべてのアウトラインを CAT1 カテゴリに変更します。

```
OUTLN_PKG.UPDATE_BY_CAT('DEFAULT', 'CAT1');
```

UTL_COLL パッケージによって、問合せや更新を行うためのコレクション・ロケータを PL/SQL プログラムで使用できます。

サブプログラムの要約

現在、このパッケージでサポートしているファンクションは、IS_LOCATOR のみです。

IS_LOCATOR ファンクション

このファンクションは、コレクション項目が実際にロケータかどうかを判別します。

構文

```
UTL_COLL.IS_LOCATOR (  
    collection IN ANY)  
RETURNS BOOLEAN;
```

パラメータ

表 59-1 IS_LOCATOR ファンクションのパラメータ

パラメータ	説明
collection	ネストした表または VARRAY 項目

戻り値

表 59-2 IS_LOCATOR ファンクションの戻り値

戻り値	説明
1	コレクション項目はロケータです。
0	コレクション項目はロケータではありません。

プラグマ

WNDS、WNPS および RNPS の各プラグマの断言。

例外

なし

例

```
CREATE OR REPLACE TYPE list_t as TABLE OF VARCHAR2(20);
/

CREATE OR REPLACE TYPE phone_book_t AS OBJECT (
    pno number,
    ph list_t );
/

CREATE TABLE phone_book OF phone_book_t
    NESTED TABLE ph STORE AS nt_ph;
CREATE TABLE phone_book1 OF phone_book_t
    NESTED TABLE ph STORE AS nt_ph_1 RETURN LOCATOR;

INSERT INTO phone_book VALUES(1, list_t('650-633-5707','650-323-0953'));
INSERT INTO phone_book1 VALUES(1, list_t('415-555-1212'));

CREATE OR REPLACE PROCEDURE chk_coll IS
    plist list_t;
    plist1 list_t;
BEGIN
    SELECT ph INTO plist FROM phone_book WHERE pno=1;

    SELECT ph INTO plist1 FROM phone_book1 WHERE pno=1;

    IF (UTL_COLL.IS_LOCATOR(plist)) THEN
        DBMS_OUTPUT.PUT_LINE('plist is a locator');
    ELSE
        DBMS_OUTPUT.PUT_LINE('plist is not a locator');
    END IF;

    IF (UTL_COLL.IS_LOCATOR(plist1)) THEN
        DBMS_OUTPUT.PUT_LINE('plist1 is a locator');
    ELSE
        DBMS_OUTPUT.PUT_LINE('plist1 is not a locator');
    END IF;

END chk_coll;

SET SERVEROUTPUT ON
EXECUTE chk_coll;
```


UTL_FILE パッケージによって、ユーザーは PL/SQL プログラムでオペレーティング・システム (OS) のテキスト・ファイルの読み込みと書込みができます。このパッケージは、標準 OS のストリーム・ファイル入出力 (I/O) の制限付きバージョンを提供します。

このファイル I/O 機能は、標準オペレーティング・システムのストリーム・ファイル I/O (OPEN、GET、PUT、CLOSE) と同様ですが、いくつかの制限があります。たとえば、FOPEN ファンクションをコールすると、ファイル・ハンドルが戻されます。後続の GET_LINE または PUT のコールでこのファイル・ハンドルを使用し、ファイルへのストリーム I/O を実行します。ファイルの I/O が終了した場合は、FCLOSE をコールして出力を完了し、そのファイルに関連しているリソースを解放します。

セキュリティ

PL/SQL のファイル I/O 機能は、PL/SQL のクライアント側とサーバー側の両方で使用可能です。クライアント・インプリメンテーション（テキスト I/O）は、オペレーティング・システムによる通常のファイル許可検査の対象であるため、追加セキュリティ制約は不要です。ただし、サーバー・インプリメンテーションは権限モードで稼働する場合があります、この機能の範囲を制限するための追加セキュリティ制約を必要とします。

注意： UTL_FILE パッケージは、Oracle Procedure Builder が提供しているクライアント側の TEXT_IO パッケージと類似しています。サーバー・インプリメンテーションに対する制約事項には、UTL_FILE と TEXT_IO の間である程度の API 差異が必要です。PL/SQL のファイル I/O では、PL/SQL 例外を使用して、ユーザーにエラーが戻されます。

サーバーのセキュリティ

PL/SQL のファイル I/O に対するサーバー・セキュリティは、アクセス可能なディレクトリへの制限で構成されています。アクセス可能なディレクトリは、インスタンス・パラメータの初期設定ファイル（INIT.ORA）で指定する必要があります。

次のように UTL_FILE_DIR パラメータを使用して、初期設定ファイル内の UTL_FILE ファンクションにアクセス可能なディレクトリを指定します。

```
UTL_FILE_DIR = <directory name>
```

注意： ディレクトリの仕様部は、プラットフォームによって異なります。

インスタンスの初期設定ファイルに行 `UTL_FILE_DIR = /usr/jsmith/my_app` が含まれている場合、ディレクトリ `/usr/jsmith/my_app` は、FOPEN ファンクションからアクセス可能です。大小文字を区別するオペレーティング・システムでは、`/usr/jsmith/My_App` という名前のディレクトリはアクセス不可になることに注意してください。

パラメータ仕様部の `UTL_FILE_DIR = *` には特別な意味があります。このエントリは、ディレクトリのアクセス検査をオフにして、UTL_FILE ファンクションからすべてのディレクトリにアクセスできるようにします。

注意：

'*' オプションの使用には、細心の注意を払ってください。このオプションは、本番のシステムでは使用しないことをお勧めします。また、アクセス可能ディレクトリのリストに '!' (UNIX の現行ディレクトリ) を含めないでください。

シンボリック・リンクが使用可能なファイル・システム上でセキュリティを保証するために、PL/SQL ファイル I/O ファンクションでアクセスできるディレクトリへの書き込み許可をユーザーに与えないでください。シンボリック・リンクと PL/SQL ファイル I/O は、通常のオペレーティング・システムの許可検査を迂回するために使用できます。また、他の方法ではアクセスできないディレクトリにユーザーが読み込みまたは書き込みアクセスをできるようにします。

ファイルの所有権と保護

UNIX システムでは、FOPEN ファンクションで作成したファイルにはそのファイルの所有者、つまりインスタンスを実行するシャドウ・プロセスの所有者がいます。通常の場合、この所有者は oracle です。FOPEN で作成したファイルは、UTL_FILE サブプログラムを使用して常に読み込みと書き込みができますが、非権限ユーザーがこれらのファイルを PL/SQL 以外で読み込む場合は、システム管理者によるアクセス権の設定が必要です。

例 (UNIX 固有)

パラメータ初期設定ファイルに次の行のみ含まれている場合は、

```
UTL_FILE_DIR=/appl/gl/log
UTL_FILE_DIR=/appl/gl/out
```

次のファイルの場所とファイル名が有効です。

FILE LOCATION	FILENAME
/appl/gl/log	L10324.log
/appl/gl/out	O10324.out

ただし、次のファイルの場所とファイル名は無効です。

FILE LOCATION	FILENAME	
/appl/gl/log/backup	L10324.log	# subdirectory
/APPL/gl/log	L10324.log	# uppercase
/appl/gl/log	backup/L10324.log	# dir in name
/usr/tmp	T10324.tmp	# not in INIT.ORA

注意： ユーザー・レベルのファイル許可はありません。UTL_FILE_DIR パラメータで指定されたすべてのファイルの場所は、ファイル I/O プロシージャのすべてのユーザーが、読み込みと書き込みのために使用できます。これによって、オペレーティング・システムのファイル許可を上書きできます。

型

```
TYPE file_type IS RECORD (id BINARY_INTEGER);
```

FILE_TYPE の内容は、UTL_FILE パッケージ専用です。このパッケージのユーザーは、このレコードのコンポーネントの参照または変更は行わないでください。

例外

表 60-1 UTL_FILE パッケージの例外

例外名	説明
INVALID_PATH	ファイルの場所またはファイル名が無効です。
INVALID_MODE	FOPEN の open_mode パラメータが無効です。
INVALID_FILEHANDLE	ファイル・ハンドルが無効です。
INVALID_OPERATION	要求どおりにファイルをオープンできないか、または操作できません。
READ_ERROR	読み込み操作中にオペレーティング・システムのエラーが発生しました。
WRITE_ERROR	書き込み操作中にオペレーティング・システムのエラーが発生しました。
INTERNAL_ERROR	PL/SQL 内の未指定エラー。

これらのパッケージ例外に加えて、UTL_FILE パッケージ内のプロシージャが、NO_DATA_FOUND や VALUE_ERROR などの事前定義済の PL/SQL 例外を発生させることもあります。

サブプログラムの要約

表 60-2 UTL_FILE サブプログラム

サブプログラム	説明
60-6 ページの FOPEN ファンクション	デフォルトの行サイズで入力用または出力用ファイルをオープンします。
60-7 ページの IS_OPEN ファンクション	ファイル・ハンドルが、オープンしているファイルを参照しているかどうかを判別します。
60-8 ページの FCLOSE プロシージャ	ファイルをクローズします。
60-8 ページの FCLOSE_ALL プロシージャ	オープンしているファイル・ハンドルをすべてクローズします。
60-9 ページの GET_LINE プロシージャ	オープンしているファイルから、テキストを 1 行読み込みます。
60-10 ページの PUT プロシージャ	ファイルに 1 行を書き込みます。行終了記号は追加されません。
60-11 ページの NEW_LINE プロシージャ	1 つ以上の OS 固有の行終了記号をファイルに書き込みます。
60-12 ページの PUT_LINE プロシージャ	ファイルに 1 行を書き込みます。OS 固有の行終了記号が追加されます。
60-12 ページの PUTF プロシージャ	フォーマット付きの PUT プロシージャです。
60-14 ページの FFLUSH プロシージャ	保留中のすべての出力データを物理的にファイルへ書き込みます。
60-14 ページの FOPEN ファンクション	指定した最大行サイズでファイルをオープンします。

FOPEN ファンクション

このファンクションは、入力用または出力用にファイルをオープンします。ファイルの場所は、インスタンスの初期化パラメータ UTL_FILE_DIR で定義されているアクセス可能なディレクトリである必要があります。ディレクトリのパスは事前に存在している必要があり、FOPEN では作成されません。

FOPEN は、ファイル・ハンドルを戻します。このファイル・ハンドルは、ファイル上のすべての後続 I/O 操作で使用する必要があります。

このバージョンの FOPEN は、最大行サイズのパラメータを使用しません。したがって、デフォルト（ほとんどのシステムで 1023）が使用されます。異なる最大行サイズを指定するには、60-14 ページの「FOPEN ファンクション」のオーバーロード・バージョンを使用します。

最大 50 ファイルまで同時にオープンできます。

構文

```
UTL_FILE.FOPEN (  
    location  IN VARCHAR2,  
    filename  IN VARCHAR2,  
    open_mode IN VARCHAR2)  
RETURN UTL_FILE.FILE_TYPE;
```

パラメータ

表 60-3 FOPEN ファンクションのパラメータ

パラメータ	説明
location	ファイルをオープンするディレクトリを指定する、オペレーティング・システム固有の文字列。
filename	ファイルの名前。拡張子（ファイル・タイプ）が含まれ、ディレクトリ・パス情報はありません。（UNIX オペレーティング・システムでは、ファイル名を '/' で終了させることはできません）。
open_mode	ファイルのオープン方法を指定する文字列（大文字も小文字も使用できます）。 サポートされている値、およびその値を指定して使用できる UTL_FILE プロシージャは次のとおりです。 'r' テキストの読み込み（GET_LINE） 'w' テキストの書き込み（PUT、PUT_LINE、NEW_LINE、PUTF、FFLUSH） 'a' テキストの追加（PUT、PUT_LINE、NEW_LINE、PUTF、FFLUSH）

注意： `open_mode` の値 'a' を使用して、存在しないファイルをオープンすると、そのファイルは書込み ('w') モードで作成されます。

戻り値

FOPEN は、そのファイルを操作する後続プロシージャすべてに渡す必要のあるファイル・ハンドルを戻します。ファイル・ハンドルの特定の内容は、UTL_FILE パッケージ専用であり、UTL_FILE のユーザーは、個々のコンポーネントの参照または変更を行わないでください。

注意： ファイルの場所とファイル名の各パラメータは、別々の文字列として FOPEN ファンクションに指定されるため、初期化ファイルで指定したとおりに、アクセス可能なディレクトリのリストと照合してファイルの場所をチェックできます。ファイルの場所と名前前でシステム上の正しいファイル名を示し、そのディレクトリがアクセス可能であることが必要です。アクセス可能なディレクトリのサブディレクトリは、必ずしもアクセス可能である必要はありません。サブディレクトリも初期設定ファイルの完全パス名を使用して指定する必要があります。

UNIX での C シェル環境変数などのオペレーティング・システム固有のパラメータは、ファイルの場所またはファイル名のパラメータでは使用できません。

例外

INVALID_PATH
INVALID_MODE
INVALID_OPERATION

IS_OPEN ファンクション

このファンクションは、オープン・ファイルをファイル・ハンドルが識別しているかどうかをテストします。IS_OPEN は、ファイル・ハンドルが、オープン状態でクローズしていないファイルを示しているかどうかを通知するのみです。このファンクションは、ユーザーがファイル・ハンドルを使用しようとしたときに、オペレーティング・システムのエラーが発生しないことを保証するものではありません。

構文

```
UTL_FILE.IS_OPEN (  
    file IN FILE_TYPE)  
RETURN BOOLEAN;
```

パラメータ

表 60-4 IS_OPEN ファンクションのパラメータ

パラメータ	説明
file	FOPEN コールが戻すアクティブなファイル・ハンドル

戻り値

TRUE または FALSE。

例外

なし

FCLOSE プロシージャ

このプロシージャは、ファイル・ハンドルが示すオープン・ファイルをクローズします。FCLOSE の実行時に、まだ書き込んでいないデータがバッファに残っていると、ファイルのクローズ時に WRITE_ERROR 例外を受け取る場合があります。

構文

```
UTL_FILE.FCLOSE (  
    file IN OUT FILE_TYPE);
```

パラメータ

表 60-5 FCLOSE プロシージャのパラメータ

パラメータ	説明
file	FOPEN コールが戻すアクティブなファイル・ハンドル

例外

```
WRITE_ERROR  
INVALID_FILEHANDLE
```

FCLOSE_ALL プロシージャ

このプロシージャは、セッションでオープンしているすべてのファイル・ハンドルをクローズします。これは、PL/SQL プログラムの例外終了などの非常時のクリーン・アップ・プロシージャとして使用します。

注意： FCLOSE_ALL は、ユーザーが保持しているオープン・ファイル・ハンドルの状態は変更しません。つまり、ファイルはクローズされていても、FCLOSE_ALL コール後のファイル・ハンドルの IS_OPEN テストでは TRUE が戻されます。FCLOSE_ALL の前にオープンされたファイルには、以降の読み込み操作や書き込み操作を行うことができません。

構文

```
UTL_FILE.FCLOSE_ALL;
```

パラメータ

なし

例外

```
WRITE_ERROR
```

GET_LINE プロシージャ

このプロシージャは、ファイル・ハンドルが示すオープン・ファイルからテキストを 1 行読み込んで、出力バッファ・パラメータに配置します。テキストは、ファイルの終わりまで読み込まれますが、行の終了記号は含まれません。

行がバッファに収まらない場合は、VALUE_ERROR 例外が発生します。" ファイルの終わり " に到達したためにテキストが読み込まれなかった場合は、NO_DATA_FOUND 例外が発生します。

行終了記号の文字はバッファに読み込まれないため、ブランク行を読み込むと空の文字列が戻されます。

オーバーロードされたバージョンの FOPEN で大きいサイズを指定しない限り、入力レコードの最大サイズは 1023 バイトです。

構文

```
UTL_FILE.GET_LINE (  
    file          IN  FILE_TYPE,  
    buffer        OUT VARCHAR2);
```

パラメータ

表 60-6 GET_LINE プロシージャのパラメータ

パラメータ	説明
file	FOPEN コールが戻すアクティブなファイル・ハンドル。 ファイルは読みみ用（モード 'r'）としてオープンする必要があります。そうでない場合は、INVALID_OPERATION 例外が発生します。
buffer	ファイルから読み込まれた行を受け取るデータ・バッファ。

例外

INVALID_FILEHANDLE
INVALID_OPERATION
READ_ERROR
NO_DATA_FOUND
VALUE_ERROR

PUT プロシージャ

PUT プロシージャは、ファイル・ハンドルが示すオープン・ファイルに、バッファ・パラメータ内に格納されているテキスト文字列を書き込みます。このファイルは書き込み操作用にオープンされる必要があります。PUT は、行終了記号を追加しません。行の終了には NEW_LINE を使用するか、または PUT_LINE を使用して行終了記号付きの完全な 1 行を書き込んでください。

オーバーロードされたバージョンの FOPEN で大きいサイズを指定しない限り、入力レコードの最大サイズは 1023 バイトです。

構文

```
UTL_FILE.PUT (  
    file      IN FILE_TYPE,  
    buffer    IN VARCHAR2);
```


パラメータ

表 60-7 PUT プロシージャのパラメータ

パラメータ	説明
file	FOPEN コールが戻すアクティブなファイル・ハンドル。
buffer	ファイルに書き込むテキストを含んだバッファ。 ファイルはモード 'w' またはモード 'a' を使用してオープンする必要があります。これ以外のモードを使用すると、INVALID_OPERATION 例外が発生します。

例外

INVALID_FILEHANDLE
INVALID_OPERATION
WRITE_ERROR

NEW_LINE プロシージャ

このプロシージャは、入力ファイル・ハンドルが示すファイルに、1 つ以上の行終了記号を書き込みます。行終了記号はプラットフォーム固有の文字や文字列であるため、このプロシージャは PUT とは異なります。

構文

```
UTL_FILE.NEW_LINE (  
    file      IN FILE_TYPE,  
    lines     IN NATURAL := 1);
```

パラメータ

表 60-8 NEW_LINE プロシージャのパラメータ

パラメータ	説明
file	FOPEN コールが戻すアクティブなファイル・ハンドル
lines	ファイルに書き込む行終了記号の数

例外

INVALID_FILEHANDLE
INVALID_OPERATION
WRITE_ERROR

PUT_LINE プロシージャ

このプロシージャは、ファイル・ハンドルが示すオープン・ファイルに、バッファ・パラメータ内に格納されているテキスト文字列を書き込みます。このファイルは書き込み操作にオープンされる必要があります。PUT_LINE は、プラットフォーム固有の行終了文字または文字列で行を終了します。

オーバーロードされたバージョンの FOPEN で大きい値を指定しない限り、出力レコードの最大サイズは 1023 バイトです。

構文

```
UTL_FILE.PUT_LINE (  
    file      IN FILE_TYPE,  
    buffer    IN VARCHAR2);
```

パラメータ

表 60-9 PUT_LINE プロシージャのパラメータ

パラメータ	説明
file	FOPEN コールが戻すアクティブなファイル・ハンドル
buffer	ファイルに書き込む行を含んだテキスト・バッファ

例外

```
INVALID_FILEHANDLE  
INVALID_OPERATION  
WRITE_ERROR
```

PUTF プロシージャ

このプロシージャは、フォーマットされた PUT プロシージャです。これは、制限付きの printf() のように動作します。フォーマット文字列には任意のテキストを指定できますが、文字 '%s' と '%n' には次のような特別な意味があります。

- %s この文字列を引数リスト内の次の引数の文字列値に置き換えます。
- %n 適切なプラットフォーム固有の行終了記号に置き換えます。

構文

```
UTL_FILE.PUTF (
    file      IN FILE_TYPE,
    format    IN VARCHAR2,
    [arg1     IN VARCHAR2  DEFAULT NULL,
    . . .
    arg5      IN VARCHAR2  DEFAULT NULL]);
```

パラメータ

表 60-10 PUTF プロシージャのパラメータ

パラメータ	説明
file	FOPEN コールが戻すアクティブなファイル・ハンドル。
format	テキストやフォーマット文字 '¥n' と '%s' を含むことができるフォーマット文字列。
arg1..arg5	1 ～ 5 個までのオプションの引数文字列。 引数文字列は、フォーマット文字列内の '%s' フォーマットに、順序正しく置き換えられます。 引数より多いフォーマットがフォーマット・パラメータ文字列内にある場合は、引数のない各 '%s' は空の文字列に置き換えられます。

例

次に、行を書き込む例を示します。

```
Hello, world!
I come from Zork with greetings for all earthlings.

my_world varchar2(4) := 'Zork';
...
PUTF(my_handle, 'Hello, world!¥nI come from %s with %s.¥n',
      my_world,
      'greetings for all earthlings');
```

引数より多い %s フォーマットがフォーマット・パラメータ内にある場合は、対応する引数のない %s は空の文字列に置き換えられます。

例外

```
INVALID_FILEHANDLE
INVALID_OPERATION
WRITE_ERROR
```

FFLUSH プロシージャ

FFLUSH は、ファイル・ハンドルが示すファイルに、保留中のデータを物理的に書き込みます。ファイルに書き込むデータは通常バッファリングされます。FFLUSH プロシージャは、バッファリングされているデータを強制的にファイルに書き込みます。データは改行文字で終了する必要があります。

フラッシュは、まだオープンしているファイルを読み込む必要がある場合に役立ちます。たとえば、デバッグ・メッセージをファイルにフラッシュして、即時に読み込むことができます。

構文

```
UTL_FILE.FFLUSH (  
    file IN FILE_TYPE);  
invalid_maxlinesize EXCEPTION;
```

パラメータ

表 60-11 FFLUSH プロシージャのパラメータ

パラメータ	説明
file	FOPEN コールが戻すアクティブなファイル・ハンドル

例外

```
INVALID_FILEHANDLE  
INVALID_OPERATION  
WRITE_ERROR
```

FOPEN ファンクション

このファンクションは、ファイルをオープンします。最大 50 ファイルまで同時にオープンできます。

注意： このバージョンの FOPEN では、最大行サイズを任意に指定できます。60-6 ページの「[FOPEN ファンクション](#)」の別のバージョンでは、デフォルトの行サイズが使用されます。

構文

```
UTL_FILE.FOPEN (
    location      IN VARCHAR2,
    filename      IN VARCHAR2,
    open_mode     IN VARCHAR2,
    max_linesize  IN BINARY_INTEGER)
RETURN file_type;
```

パラメータ

表 60-12 FOPEN ファンクションのパラメータ

パラメータ	説明
location	ファイルのディレクトリ位置
filename	ファイル名（拡張子を含む）
open_mode	オープン・モード（'r'、'w'、'a'）
max_linesize	改行文字を含むこのファイルの 1 行あたりの最大文字数（最小値は 1、最大値は 32767）

戻り値

表 60-13 FOPEN ファンクションの戻り値

戻り値	説明
file_type	オープン・ファイルのハンドル

例外

- INVALID_PATH: ファイルの場所またはファイル名が無効です。
- INVALID_MODE: open_mode の文字列が無効です。
- INVALID_OPERATION: ファイルを要求どおりにオープンできません。
- INVALID_MAXLINESIZE: 指定した max_linesize が大きすぎるか、または小さすぎます。

UTL_HTTP は、PL/SQL と SQL からハイパー・テキスト転送プロトコル (HTTP) のコールアウトを行います。ユーザーはこのファンクションを使用して、インターネット上のデータにアクセスしたり、Oracle Web Server カートリッジをコールできます。

UTL_HTTP には、REQUEST と REQUEST_PIECES の 2 つの類似したエントリポイントが含まれています。各ファンクションは、ユニバーサル・リソース・ロケータ (URL) の文字列を取得し、サイトに接続して、そのサイトで取得したデータ (一般的には HTML、ハイパー・テキスト・マークアップ言語) を戻します。

Oracle Wallet Manager でセットアップする有効な Oracle Wallet は、HTTPS (保護 HTTP) を利用している Web サイトから UTL_HTTP を使用してページをフェッチする場合にのみ必要であることに注意してください。より一般的な通常の HTTP フェッチに Wallet は不要です。

関連項目： Wallet Manager の詳細は、『Oracle8i Advanced Security 管理者ガイド』を参照してください。

例外

表 61-1 UTL_HTTP パッケージの例外

例外	説明
INIT_FAILED	HTTP コールアウト・サブシステムの初期化に失敗しました（使用可能メモリーの不足などの環境的な理由による）。
REQUEST_FAILED	HTTP コールに失敗しました（たとえば、HTTP デーモンに失敗した場合、REQUEST や REQUEST_PIECES への引数が NULL か HTTP 以外の構文であるために、URL として解釈できない場合など）。

前述の 2 つの例外は、例外ハンドラで明示的に取得しない限り、次のメッセージでレポートされます。ORA-06510: PL/SQL: ユーザー定義の例外が処理されませんでした。これらの例外はこのシステム・パッケージで定義されますが、ユーザー定義としてレポートされます。

HTTP 要求の処理中にその他の例外状況が発生した場合（メモリー不足エラーなど）、ファンクション REQUEST または REQUEST_PIECES は、その例外を再発生させます。

指定した URL への要求から応答がない場合（たとえば、URL に対応するサイトに接続できないため）は、フォーマットされた HTML エラー・メッセージが戻されます。例は次のとおりです。

```
<HTML>
<HEAD>
<TITLE>Error Message</TITLE>
</HEAD>
<BODY>
<H1>Fatal Error 500</H1>
Can't Access Document: http://home.nothing.comm.
<P>
<B>Reason:</B> Can't locate remote host: home.nothing.comm.
<P>

<P><HR>
<ADDRESS><A HREF="http://www.w3.org">
CERN-HITPD3.0A</A></ADDRESS>
</BODY>
</HTML>
```


使用上の注意

同一マシン（同一の権限や環境変数など）でブラウザを使用して URL に接続できない場合、REQUEST または REQUEST_PIECES で、その URL への接続が成功することは期待できません。

REQUEST または REQUEST_PIECES に失敗した場合（たとえば、例外が発生した場合や HTML フォーマットのエラー・メッセージが戻された場合で、URL 引数は正しいと考えられる場合）は、ブラウザで同じ URL に接続を試みて、ユーザーのマシンからネットワークの可用性を検証します。ユーザーのブラウザにプロキシ・サーバー・セットがある場合、そのセットは、オプションの proxy パラメータを使用して REQUEST または REQUEST_PIECES の各コールごとに設定する必要があります。

注意： UTL_HTTP では、環境変数を使用してそのプロキシ動作を指定することもできます。たとえば、UNIX では、環境変数 http_proxy を URL に設定することによって、そのサービスを HTTP 要求のプロキシ・サーバーとして使用するように指定します。また、環境変数 no_proxy をドメイン名に設定することによって、そのドメインの URL では HTTP プロキシ・サーバーを使用しないように指定します。

サブプログラムの要約

表 61-2 UTL_HTTP パッケージのサブプログラム

サブプログラム	説明
61-3 ページの REQUEST ファンクション	指定した URL から取り出したデータの最初の 2000 バイトまでを戻します。
61-5 ページの REQUEST_PIECES ファンクション	指定した URL から取り出したデータについて、2000 バイト・ピースの PL/SQL 表を戻します。

REQUEST ファンクション

このファンクションは、指定した URL から取り出したデータの最初の 2000 バイトまでを戻します。

構文

```
UTL_HTTP.REQUEST (  
    url      IN VARCHAR2,  
    proxy    IN VARCHAR2 DEFAULT NULL,  
    wallet_path IN VARCHAR2 DEFAULT NULL,  
    wallet_password IN VARCHAR2 DEFAULT NULL)  
RETURN VARCHAR2;
```

プラグマ

```
pragma restrict_references (request, wnds, rnds, wnps, rmps);
```

パラメータ

表 61-3 に、REQUEST ファンクションのパラメータを示します。

表 61-3 REQUEST ファンクションのパラメータ

パラメータ	説明
url	ユニバーサル・リソース・ロケータ。
proxy	(オプション) HTTP 要求時に使用するプロキシ・サーバーを指定します。
wallet_path	(オプション) クライアント側の Wallet を指定します。クライアント側の Wallet には、HTTPS 要求に必要な信頼されている認証局のリストが含まれています。wallet_path の書式は、'file:/<local-dir-for-client-side-wallet>' です。
wallet_password	(オプション) Wallet のオープンに必要なパスワードを指定します。

戻り値

戻り型は長さ 2000 以下の文字列で、引数 URL への HTTP 要求から戻された HTML 結果から最初の 2000 バイトまでが含まれます。

例外

```
INIT_FAILED
REQUEST_FAILED
```

例

```
SQLPLUS> SELECT utl_http.request('http://www.oracle.com/') FROM dual;
UTL_HTTP.REQUEST('HTTP://WWW.ORACLE.COM/')
<html>
<head><title>Oracle Corporation Home Page</title>
<!--changed Jan. 16, 19
1 row selected.
```

firewall の内側にいる場合は、proxy パラメータを含めます。たとえば、Oracle firewall 内には、www-proxy.us.oracle.com という名前のプロキシ・サーバーがあります。

```
SQLPLUS> SELECT
utl_http.request('http://www.oracle.com', 'www-proxy.us.oracle.com') FROM dual;
```

REQUEST_PIECES ファンクション

このファンクションは、指定した URL から取り出したデータについて、2000 バイト・ピースの PL/SQL 表を戻します。

構文

type html_pieces is table of varchar2(2000) index by binary_integer;

```
UTL_HTTP.REQUEST_PIECES (  
    url          IN VARCHAR2,  
    max_pieces   NATURAL      DEFAULT 32767,  
    proxy        IN VARCHAR2  DEFAULT NULL,  
    wallet_path  IN VARCHAR2  DEFAULT NULL,  
    wallet_password IN VARCHAR2  DEFAULT NULL)  
RETURN HTML_PIECES;
```

プラグマ

pragma restrict_references (request_pieces, wnds, rnds, wnps, rnps);

パラメータ

表 61-4 に、REQUEST ファンクションのパラメータを示します。

表 61-4 REQUEST_PIECES ファンクションのパラメータ

パラメータ	説明
url	ユニバーサル・リソース・ロケータ。
max_pieces	(オプション) REQUEST_PIECES が戻すピースの最大数（長さは各 2000 文字、最後のピースはこれより短くなる場合があります）。指定する場合、引数は正の整数を指定します。
proxy	(オプション) HTTP 要求時に使用するプロキシ・サーバーを指定します。
wallet_path	(オプション) クライアント側の Wallet を指定します。クライアント側の Wallet には、HTTPS 要求に必要な信頼されている認証局のリストが含まれています。wallet_path の書式は、'file:/<local-dir-for-client-side-wallet>' です。
wallet_password	(オプション) Wallet のオープンに必要なパスワードを指定します。

戻り値

REQUEST_PIECES は、UTL_HTTPHTML_PIECES 型の PL/SQL 表を戻します。PL/SQL 表の各要素は、長さ 2000 の文字列です。最後の要素は、2000 文字より短くなる場合があります。

REQUEST_PIECES で戻される PL/SQL 表の要素は、その URL への HTTP 要求から取得されたデータの連続したピースです。

例外

```
INIT_FAILED
REQUEST_FAILED
```

例

REQUEST_PIECES のコールの例は、次のようになります。戻されるピースの数 (0 以上) を調べるための PL/SQL 表メソッド COUNT の使用方法に注意してください。

```
DECLARE pieces utl_http.html_pieces;
BEGIN
    pieces := utl_http.request_pieces('http://www.oracle.com/');
    FOR i in 1 .. pieces.count loop
        .... -- process each piece
    END LOOP;
END;
```

例

次のブロックは、URL から最大 100 ピースまでのデータ（最後のピースを除いて各 2000 バイト）を取り出します。取り出すピースの数と、取り出すデータの合計バイト長を出力します。

```
SET SERVEROUTPUT ON
/
DECLARE
    x utl_http.html_pieces;
BEGIN
    x := utl_http.request_pieces('http://www.oracle.com/', 100);
    dbms_output.put_line(x.count || ' pieces were retrieved. ');
    dbms_output.put_line('with total length ');
    IF x.count < 1
    THEN dbms_output.put_line('0');
    ELSE dbms_output.put_line
        ((2000 * (x.count - 1)) + length(x(x.count)));
    END IF;
END;
/
-- Output
```

Statement processed.
4 pieces were retrieved.
with total length
7687

UTL_INADDR

UTL_INADDR パッケージは、インターネット・アドレッシングをサポートするための PL/SQL プロシージャを提供します。ホスト名を取り出す API およびローカル・ホストとリモート・ホストの IP アドレスを提供します。

注意： このパッケージには、JServer オプションのインストールが必要です。

インターネット・アドレス・パッケージ

UTL_INADDR パッケージには、2つのファンクションがあります。

- [get_host_name\(\)](#)
- [get_host_address\(\)](#)

get_host_name()

用途

ローカル・ホストの名前を取り出します。

構文

```
FUNCTION get_host_name RETURN VARCHAR2;
```

コメント

なし

get_host_address()

用途

ホストの IP アドレスを取り出します。

構文

```
FUNCTION get_host_address (host IN VARCHAR2 DEFAULT NULL) RETURN VARCHAR2;
```

パラメータ

host (IN) IP アドレスを取り出すホストの名前。ホストが NULL の場合、このファンクションはローカル・ホストの IP アドレスを戻します。

例外

表 62-1 に、インターネット・アドレス・パッケージで発生する例外を示します。

表 62-1 インターネット・アドレス・パッケージからの例外

例外	説明
UNKNOWN_HOST	ホスト名が不明です。

UTL_RAW パッケージは、RAW データ型を操作するための SQL ファンクションを提供します。通常の SQL ファンクションは複数の RAW で作動せず、PL/SQL は RAW データ型と CHAR データ型の間でのオーバーロードができないため、このパッケージが必要になります。UTL_RAW には、各種の COBOL 数値書式を複数の RAW の間で変換するサブプログラムも含まれています。

UTL_RAW は、データベース環境に固有ではなく、他の環境でもデータベース環境と同じように実際に使用できます。このため、DBMS のかわりに、UTL という接頭辞がパッケージに付けられます。

使用上の注意

RAW ファンクションには、多くの使用方法があります。UTL_RAW によって、RAW レコードは多くの要素で構成できます。RAW データ型を使用すると、キャラクタ・セット変換は実行されず、RAW は、リモート・プロシージャ・コール（RPC）を介して転送されるときに元の書式で保持されます。

また、RAW ファンクションによって、以前は hextoraw ファンクションと rawtohex ファンクションに限定されていたバイナリ・データを操作できます。

サブプログラムの要約

表 63-1 UTL_RAW パッケージのサブプログラム

サブプログラム	説明
63-3 ページの CONCAT ファンクション	最大 12 までの RAW を単一の RAW に連結します。
63-4 ページの CAST_TO_RAW ファンクション	n データ・バイトを使用して表した VARCHAR2 を、n データ・バイトを持つ RAW に変換します。
63-5 ページの CAST_TO_VARCHAR2 ファンクション	n データ・バイトを使用して表した RAW を、n データ・バイトを持つ VARCHAR2 に変換します。
63-6 ページの LENGTH ファンクション	RAW r の長さをバイトで戻します。
63-7 ページの SUBSTR ファンクション	pos から開始して、RAW r から len バイトを戻します。
63-8 ページの TRANSLATE ファンクション	RAW from_set と to_set の変換によるバイトに従って、入力 RAW r 内のバイトを変換します。
63-10 ページの TRANSLITERATE ファンクション	RAW from_set と to_set の文字変換によるバイトに従って、入力 RAW r 内のバイトを変換します。
63-11 ページの OVERLAY ファンクション	ターゲット RAW の指定部分を overlay RAW でオーバーレイし、ターゲットのバイト位置 pos から始まる len バイト分を処理します。
63-13 ページの COPIES ファンクション	r を n 回のコピーで連結したものを戻します。
63-14 ページの XRANGE ファンクション	値 start_byte で始まり値 end_byte で終わる、連続した有効な 1 バイト・コードをすべて含む RAW を戻します。
63-15 ページの REVERSE ファンクション	RAW r のバイトの順序を、最後から最初に逆転させます。

表 63-1 UTL_RAW パッケージのサブプログラム

サブプログラム	説明
63-16 ページの COMPARE ファンクション	RAW r1 と RAW r2 を比較します。
63-17 ページの CONVERT ファンクション	RAW r をキャラクタ・セット from_charset からキャラクタ・セット to_charset に変換し、結果の RAW を返します。
63-19 ページの BIT_AND ファンクション	RAW r1 と RAW r2 の値でビット単位の論理演算 "AND" を実行し、AND 演算後の結果 RAW を返します。
63-20 ページの BIT_OR ファンクション	RAW r1 と RAW r2 の値でビット単位の論理演算 "OR" を実行し、OR 演算後の結果 RAW を返します。
63-21 ページの BIT_XOR ファンクション	RAW r1 と RAW r2 の値でビット単位の論理演算 "排他 OR" を実行し、排他 OR 演算後の結果 RAW を返します。
63-22 ページの BIT_COMPLEMENT ファンクション	RAW r の値でビット単位の論理演算 "補数" を実行し、補数演算後の結果 RAW を返します。

CONCAT ファンクション

このファンクションは、最大 12 までの RAW を単一の RAW に連結します。連結したサイズが 32K を超える場合は、エラーが戻ります。

構文

```
UTL_RAW.CONCAT (
  r1  IN RAW DEFAULT NULL,
  r2  IN RAW DEFAULT NULL,
  r3  IN RAW DEFAULT NULL,
  r4  IN RAW DEFAULT NULL,
  r5  IN RAW DEFAULT NULL,
  r6  IN RAW DEFAULT NULL,
  r7  IN RAW DEFAULT NULL,
  r8  IN RAW DEFAULT NULL,
  r9  IN RAW DEFAULT NULL,
  r10 IN RAW DEFAULT NULL,
  r11 IN RAW DEFAULT NULL,
  r12 IN RAW DEFAULT NULL)
RETURN RAW;
```

プラグマ

```
pragma restrict_references(concat, WNDS, RNDS, WNPS, RNPS);
```

パラメータ

r1....r12 は、連結する RAW 項目です。

戻り値

表 63-2 CONCAT ファンクションの戻り値

戻り値	説明
RAW	連結された項目

エラー

入力値の合計の長さが RAW の最大許容長である 32767 バイトを超えると、エラーが発生します。

CAST_TO_RAW ファンクション

このファンクションは、n データ・バイトを使用して表した VARCHAR2 を、n データ・バイトの RAW に変換します。データは変更されませんが、データ型のみ RAW データ型に変換されます。

構文

```
UTL_RAW.CAST_TO_RAW (  
    c IN VARCHAR2)  
    RETURN RAW;
```

プラグマ

```
pragma restrict_references(cast_to_raw, WNDS, RNDS, WNPS, RNPS);
```

パラメータ

表 63-3 CAST_TO_RAW ファンクションのパラメータ

パラメータ	説明
c	RAW に変換される VARCHAR2

戻り値

表 63-4 CAST_TO_RAW ファンクションの戻り値

戻り値	説明
RAW	先行する長さのフィールドのない、入力 VARCHAR2 と同じバイト長の同じデータ
NULL	入力パラメータ c が NULL の場合

エラー

なし

CAST_TO_VARCHAR2 ファンクション

このファンクションは、n データ・バイトを使用して表した RAW を、n データ・バイトの VARCHAR2 に変換します。

注意： VARCHAR2 への変換時、その VARCHAR2 内の文字に対して現行の NLS キャラクタ・セットが使用されます。

構文

```
UTL_RAW.CAST_TO_VARCHAR2 (  
    r IN RAW)  
RETURN VARCHAR2;
```

プラグマ

```
pragma restrict_references (cast_to_varchar2, WNDS, RNDS, WNPS, RNPS);
```

パラメータ

表 63-5 CAST_TO_VARCHAR2 ファンクションのパラメータ

パラメータ	説明
r	VARCHAR2 に変更する RAW（先行する長さのフィールドなし）

戻り値

表 63-6 CAST_TO_VARCHAR2 ファンクションの戻り値

戻り値	説明
VARCHAR2	入力 RAW と同じデータ
NULL	入力パラメータ <code>r</code> が NULL の場合

エラー
なし

LENGTH ファンクション

このファンクションは、RAW `r` の長さをバイトで戻します。

構文

```
UTL_RAW.LENGTH (  
  r IN RAW)  
  RETURN NUMBER;
```

プラグマ

```
pragma restrict_references(length, WNDS, RNDS, WNPS, RNPS);
```

パラメータ

表 63-7 LENGTH ファンクションのパラメータ

パラメータ	説明
<code>r</code>	長さを測定する RAW バイト・ストリーム

戻り値

表 63-8 LENGTH ファンクションの戻り値

戻り値	説明
NUMBER	RAW の現行の長さと同じ長さ

エラー
なし

SUBSTR ファンクション

このファンクションは、RAW *r* から *pos* で始まる *len* バイトを戻します。

構文

```
UTL_RAW.SUBSTR (  
    r      IN RAW,  
    pos    IN BINARY_INTEGER,  
    len    IN BINARY_INTEGER DEFAULT NULL)  
RETURN RAW;
```

プラグマ

```
pragma restrict_references (substr, WNDS, RNDS, WNPS, RNPS);
```

パラメータ

pos が正の値の場合、SUBSTR は *r* の初めからカウントして最初のバイトを検索します。
pos が負の値の場合、SUBSTR は *r* の最後から逆方向にカウントします。値 *pos* は 0 に指定できません。

len を省略すると、SUBSTR は *r* の最後までバイトをすべて戻します。値 *len* は 1 未満に指定できません。

表 63-9 SUBSTR ファンクションのパラメータ

パラメータ	説明
<i>r</i>	一部分を抽出する RAW バイト列
<i>pos</i>	<i>r</i> 内で抽出を開始するバイト位置
<i>len</i>	<i>r</i> から抽出する、 <i>pos</i> からのバイト数（オプション）

デフォルトとオプション・パラメータ

表 63-10 SUBSTR ファンクションの例外

オプション・パラメータ	説明
<i>len</i>	位置 <i>pos</i> から <i>r</i> の終わりまでの長さ

戻り値

表 63-11 SUBSTR ファンクションの戻り値

戻り値	説明
portion of r	pos から始まる len バイト長
NULL	入力パラメータ r が NULL の場合

エラー

表 63-12 SUBSTR ファンクションのエラー

エラー	説明
VALUE_ERROR	pos = 0 または len < 0 のいずれかです。

TRANSLATE ファンクション

このファンクションは、RAW from_set と to_set の変換によるバイトに従って、入力 RAW r 内のバイトを変換します。r 内のバイトが from_set 内のバイトと一致すると、to_set 内の対応する位置にあるバイトに置換され、一致しないと削除されます。

r 内のバイトが from_set で未定義の場合は、結果にコピーされます。from_set にある最初（最左端）のバイトのみ使用されます。後続の複製部分はスキャンされずに無視されます。to_set が from_set より短い場合は、from_set の余分なバイトに対応する変換はなく、r 内に一致するバイトはありません。

注意： TRANSLITERATE とは、次の点で異なります。

- 変換 RAW にデフォルトはありません。
- to_set の変換 RAW で未定義の r バイトは削除されます。
- 結果 RAW は、入力 RAW r より短い場合があります。

構文

```
UTL_RAW.TRANSLATE (  
    r          IN RAW,  
    from_set  IN RAW,  
    to_set    IN RAW)  
RETURN RAW;
```

プラグマ

```
pragma restrict_references(translate, WNDS, RNDS, WNPS, RNPS);
```

パラメータ

表 63-13 TRANSLATE ファンクションのパラメータ

パラメータ	説明
r	変換する RAW ソース・バイト列
from_set	変換する RAW バイト・コード (r にある場合)
to_set	対応する from_str バイトが変換される RAW バイト・コード

戻り値

表 63-14 TRANSLATE ファンクションの戻り値

戻り値	説明
RAW	変換されたバイト列

エラー

表 63-15 TRANSLATE ファンクションのエラー

エラー	説明
VALUE_ERROR	次のいずれかです。 - r が NULL または長さ 0 (あるいはその両方) です。 - from_set が NULL または長さ 0 (あるいはその両方) です。 - to_set が NULL または長さ 0 (あるいはその両方) です。

TRANSLITERATE ファンクション

このファンクションは、RAW from_set と to_set の文字変換によるバイトに従って、入力 RAW r 内のバイトを変換します。r 内の連続するバイトが from_set 内で検索され、見つからない場合は、変更しないまま結果 RAW にコピーされます。見つかった場合、そのバイトは、to_set の対応するバイト、対応するバイトが存在しない場合は pad バイトのいずれかに結果 RAW 内で置換されます。

r 内のバイトが from_set で未定義の場合は、結果にコピーされます。from_set にある最初（最左端）のバイトのみ使用されます。後続の複製部分はスキャンされずに無視されます。結果 RAW は、常に r と同じ長さになります。

to_set が from_set より短い場合、選択した from_set バイトに対応する to_set バイトがないと、pad バイトが結果 RAW に埋め込まれます（pad バイトを使用して、to_set が from_set と同じ長さまで拡張されたようになります）。

- 注意：** TRANSLATE とは、次の点で異なります。
- to_set で未定義の r バイトが埋め込まれます。
- 結果 RAW は、常に入力 RAW r と同じ長さになります。

構文

```
UTL_RAW.TRANSLITERATE (  
    r           IN RAW,  
    to_set      IN RAW DEFAULT NULL,  
    from_set    IN RAW DEFAULT NULL,  
    pad         IN RAW DEFAULT NULL)  
RETURN RAW;
```

プラグマ

```
pragma restrict_references(transliterate, WNDS, RNDS, WNPS, RNPS);
```

パラメータ

表 63-16 TRANSLITERATE ファンクションのパラメータ

パラメータ	説明
r	変換する RAW 入力バイト列
from_set	変換する RAW バイト・コード（r にある場合）（任意の長さ）
to_set	対応する from_set バイトが変換される RAW バイト・コード（任意の長さ）

表 63-16 TRANSLITERATE ファンクションのパラメータ

パラメータ	説明
pad	to-set が from_set より短い場合に使用する 1 バイト

デフォルトとオプション・パラメータ

表 63-17 TRANSLITERATE ファンクションのオプション・パラメータ

オプション・パラメータ	説明
from_set	x'00 ～ x'ff
to_set	NULL 文字列まで。実際は、必要に応じて pad を使用して、from_set の長さまで拡張されます。
pad	x'00'

戻り値

表 63-18 TRANSLITERATE ファンクションの戻り値

戻り値	説明
RAW	変換されたバイト列

エラー

表 63-19 TRANSLITERATE ファンクションのエラー

エラー	説明
VALUE_ERROR	R が NULL または長さ 0（あるいはその両方）です。

OVERLAY ファンクション

このファンクションは、ターゲット RAW の指定部分をオーバーレイ RAW でオーバーレイし、ターゲットのバイト位置 pos から始まる len バイト分を処理します。

overlay が len バイト未満の場合は、pad バイトを使用して len バイトまで拡張されます。overlay が len バイトを超える場合は、オーバーレイの余分なバイトは無視されます。ターゲットの位置 pos から始まる len バイトがターゲットの長さを超える場合、ターゲットは拡張されて overlay 全体の長さになります。

len が指定されている場合は、0 以上である必要があります。pos が指定されている場合は、1 以上である必要があります。pos がターゲットの長さを超えている場合、ターゲット

は pad バイトを使用して位置 pos まで埋め込まれ、さらにターゲットは overlay バイトを使用して拡張されます。

構文

```
UTL_RAW.OVERLAY (  
    overlay_str IN RAW,  
    target      IN RAW,  
    pos         IN BINARY_INTEGER DEFAULT 1,  
    len         IN BINARY_INTEGER DEFAULT NULL,  
    pad         IN RAW              DEFAULT NULL)  
RETURN RAW;
```

プラグマ

```
pragma restrict_references(overlay, WNDS, RNDS, WNPS, RNPS);
```

パラメータ

表 63-20 OVERLAY ファンクションのパラメータ

パラメータ	説明
overlay_str	ターゲットをオーバーレイするために使用するバイト列
target	オーバーレイするバイト列
pos	オーバーレイを開始する、ターゲット内での位置（1 から番号付けされている）
len	オーバーレイするターゲット・バイトの数
pad	オーバーレイ len がオーバーレイ長を超えた場合、または pos がターゲットの長さを超えた場合に使用するパッド・バイト

デフォルトとオプション・パラメータ

表 63-21 OVERLAY ファンクションのオプション・パラメータ

オプション・パラメータ	説明
pos	1
len	オーバーレイの長さまで
pad	x'00'

戻り値

表 63-22 OVERLAY ファンクションの戻り値

戻り値	説明
RAW	指定したとおりにオーバーレイされたターゲットの byte_string

エラー

表 63-23 OVERLAY ファンクションのエラー

エラー	説明
VALUE_ERROR	次のいずれかです。 <ul style="list-style-type: none">- オーバーレイが NULL または長さ 0（あるいはその両方）です。- ターゲットが不明か、未定義です。- ターゲットの長さが、RAW の最大長を超えました。- len < 0- pos < 1

COPIES ファンクション

このファンクションは、r を n 回コピーして連結したものを戻します。

構文

```
UTL_RAW.COPIES (  
  r IN RAW,  
  n IN NUMBER)  
RETURN RAW;
```

プラグマ

```
pragma restrict_references(copies, WNDS, RNDS, WNPS, RNPS);
```

パラメータ

表 63-24 COPIES ファンクションのパラメータ

パラメータ	説明
r	コピーする RAW
n	RAW をコピーする回数（必ず正の数で指定）

戻り値

このファンクションは、n 回コピーした RAW を戻します。

エラー

表 63-25 COPIES ファンクションのエラー

エラー	説明
VALUE_ERROR	次のいずれかです。 - r が不明、NULL または長さ 0（あるいはその両方）です。 - n < 1 - 結果の長さが、RAW の最大長を超えました。

XRANGE ファンクション

このファンクションは、連続した有効な 1 バイト・コードをすべて含む RAW を戻し、値 start_byte で始まり値 end_byte で終わります。start_byte が end_byte より大きい場合、結果バイトの連続は start_byte で始まり、'FF'x から '00'x に折り返して end_byte で終わります。start_byte と end_byte を指定する場合は、単一バイトの RAW である必要があります。

構文

```
UTL_RAW.XRANGE (  
  start_byte IN RAW DEFAULT NULL,  
  end_byte   IN RAW DEFAULT NULL)  
RETURN RAW;
```

プラグマ

```
pragma restrict_references(xrange, WNDS, RNDS, WNPS, RNPS);
```


パラメータ

表 63-26 XRANGE ファンクションのパラメータ

パラメータ	説明
start_byte	戻される連続値の最初のバイト・コード値
end_byte	戻される連続値の最後のバイト・コード値

デフォルトとオプション・パラメータ

start_byte - x'00'
end_byte - x'FF'

戻り値

表 63-27 XRANGE ファンクションの戻り値

戻り値	説明
RAW	連続した有効な 1 バイトの 16 進数コード

エラー

なし

REVERSE ファンクション

このファンクションは、RAW *r* のバイトの順序を、最後から最初に逆転させます。たとえば、x'0102F3' は x'F30201' に逆転し、'xyz' は 'zyx' に逆転します。結果の長さは、入力 RAW の長さと同一です。

構文

```
UTL_RAW.REVERSE (  
    r IN RAW)  
RETURN RAW;
```

プラグマ

```
pragma restrict_references(reverse, WNDS, RNDS, WNPS, RNPS);
```

パラメータ

表 63-28 REVERSE ファンクションのパラメータ

パラメータ	説明
r	逆転する RAW

戻り値

表 63-29 REVERSE ファンクションの戻り値

戻り値	説明
RAW	r の逆転値を含んでいます。

エラー

表 63-30 REVERSE ファンクションのエラー

エラー	説明
VALUE_ERROR	R が NULL または長さ 0（あるいはその両方）です。

COMPARE ファンクション

このファンクションは、RAW r1 と RAW r2 を比較します。r1 と r2 の長さが異なる場合、短い方の RAW は、必要に応じて pad バイトを使用して右側に拡張されます。

構文

```
UTL_RAW.COMPARE (  
    r1  IN RAW,  
    r2  IN RAW,  
    pad IN RAW DEFAULT NULL)  
RETURN NUMBER;
```

プラグマ

```
pragma restrict_references(compare, WNDS, RNDS, WNPS, RNPS);
```

パラメータ

表 63-31 COMPARE ファンクションのパラメータ

パラメータ	説明
r1	比較する 1 番目の RAW で、NULL または長さ 0（あるいはその両方）が可能
r2	比較する 2 番目の RAW で、NULL または長さ 0（あるいはその両方）が可能
pad	r1 または r2 の短い方を拡張するためのバイト

デフォルトとオプション・パラメータ

pad - x'00'

戻り値

表 63-32 COMPARE ファンクションの戻り値

戻り値	説明
NUMBER	RAW バイト列が両方とも NULL または等しい場合は、0。または、最初に不一致になったバイト位置（1 から番号付けされている）と等しい番号。

エラー

なし

CONVERT ファンクション

このファンクションは、RAW r をキャラクタ・セット from_charset からキャラクタ・セット to_charset に変換し、結果の RAW を戻します。

from_charset と to_charset は両方とも、Oracle Server に定義されているサポート・キャラクタのセットである必要があります。

構文

```
UTL_RAW.CONVERT (
  r                IN RAW,
  to_charset       IN VARCHAR2,
  from_charset     IN VARCHAR2)
RETURN RAW;
```

プラグマ

```
pragma restrict_references(convert, WNDS, RNDS, WNPS, RNPS);
```

パラメータ

表 63-33 CONVERT ファンクションのパラメータ

パラメータ	説明
r	変換する RAW バイト列
to_charset	r が変換される NLS キャラクタ・セットの名前
from_charset	r が提供される NLS キャラクタ・セットの名前

戻り値

表 63-34 CONVERT ファンクションの戻り値

戻り値	説明
RAW	指定したキャラクタ・セットに従って変換されたバイト列 r

エラー

表 63-35 CONVERT ファンクションのエラー

エラー	説明
VALUE_ERROR	次のいずれかです。 - r が不明、NULL または長さ 0（あるいはその両方）です。 - from_charset または to_charset が不明、NULL または長さ 0（あるいはその両方）です。 - from_charset または to_charset の名前が無効か、またはサポートされていません。

BIT_AND ファンクション

このファンクションは、RAW r1 と RAW r2 の値でビット単位の論理演算 "AND" を実行し、"AND" 演算後の結果 RAW を戻します。

r1 と r2 の長さが異なる場合、"AND" 演算は、短い方の RAW の最終バイト後に終了し、長い方の RAW の未処理部分は部分結果に追加されます。したがって、結果の長さは、2つの入力 RAW の長い方と同じ長さになります。

構文

```
UTL_RAW.BIT_AND (  
    r1 IN RAW,  
    r2 IN RAW)  
RETURN RAW;
```

プラグマ

```
pragma restrict_references(bit_and, WNDS, RNDS, WNPS, RNPS);
```

パラメータ

表 63-36 BIT_AND ファンクションのパラメータ

パラメータ	説明
r1	r2 と "AND" 演算をする RAW
r2	r1 と "AND" 演算をする RAW

戻り値

表 63-37 BIT_AND ファンクションの戻り値

戻り値	説明
RAW	r1 と r2 の "AND" 演算結果を含んでいます。
NULL	入力パラメータ r1 または r2 が NULL の場合。

エラー

なし

BIT_OR ファンクション

このファンクションは、RAW r1 と RAW r2 の値でビット単位の論理演算 "OR" を実行し、"OR" 演算後の結果 RAW を戻します。

r1 と r2 の長さが異なる場合、"OR" 演算は、短い方の RAW の最終バイト後に終了し、長い方の RAW の未処理部分は部分結果に追加されます。したがって、結果の長さは、2 つの入力 RAW の長い方と同じ長さになります。

構文

```
UTL_RAW.BIT_OR (  
    r1 IN RAW,  
    r2 IN RAW)  
RETURN RAW;
```

プラグマ

```
pragma restrict_references(bit_or, WNDS, RNDS, WNPS, RNPS);
```

パラメータ

表 63-38 BIT_OR ファンクションのパラメータ

パラメータ	説明
r1	r2 と "OR" 演算をする RAW
r2	r1 と "OR" 演算をする RAW

戻り値

表 63-39 BIT_OR ファンクションの戻り値

戻り値	説明
RAW	r1 と r2 の "OR" 演算結果を含んでいます。
NULL	入力パラメータ r1 または r2 が NULL の場合。

エラー

なし

BIT_XOR ファンクション

このファンクションは、RAW r1 と RAW r2 の値でビット単位の論理演算 " 排他 OR" を実行し、" 排他 OR" 演算後の結果 RAW を戻します。

r1 と r2 の長さが異なる場合、" 排他 OR" 演算は、短い方の RAW の最終バイト後に終了し、長い方の RAW の未処理部分は部分結果に追加されます。したがって、結果の長さは、2 つの入力 RAW の長い方と同じ長さになります。

構文

```
UTL_RAW.BIT_XOR (  
    r1 IN RAW,  
    r2 IN RAW)  
RETURN RAW;
```

プラグマ

```
pragma restrict_references(bit_xor, WNDS, RNDS, WNPS, RNPS);
```

パラメータ

表 63-40 BIT_XOR ファンクションのパラメータ

パラメータ	説明
r1	r2 と " 排他 OR" 演算をする RAW
r2	r1 と " 排他 OR" 演算をする RAW

戻り値

表 63-41 BIT_XOR ファンクションの戻り値

戻り値	説明
RAW	r1 と r2 の " 排他 OR" 演算結果を含んでいます。
NULL	入力パラメータ r1 または r2 が NULL の場合。

エラー

なし

BIT_COMPLEMENT ファンクション

このファンクションは、RAW *r* の値でビット単位の論理演算 " 補数 " を実行し、" 補数 " 演算後の結果 RAW を戻します。結果の長さは、入力 RAW *r* の長さと等しくなります。

構文

```
UTL_RAW.BIT_COMPLEMENT (  
    r IN RAW)  
    RETURN RAW;
```

プラグマ

```
pragma restrict_references(bit_complement, WNDS, RNDS, WNPS, RNPS);
```

パラメータ

表 63-42 BIT_COMPLEMENT ファンクションのパラメータ

パラメータ	説明
<i>r</i>	" 補数 " 演算を実行する RAW

戻り値

表 63-43 BIT_COMPLEMENT ファンクションの戻り値

戻り値	説明
RAW	<i>r1</i> の " 補数 " 演算結果を含んでいます。
NULL	入力パラメータ <i>r</i> が NULL の場合。

エラー

なし

Oracle8i は、ユーザー定義のコンポジット型またはオブジェクト型をサポートします。オブジェクト型のインスタンスをオブジェクトと呼びます。オブジェクト型は、列の型としてまたは表の型として使用できます。

オブジェクト表では、表の各行にオブジェクトが格納されています。オブジェクト表にあるオブジェクトは、オブジェクト識別子で一意に識別できます。

参照はオブジェクトへの持続ポインタで、各参照にはオブジェクト識別子を含めることができます。参照は、オブジェクト型の属性にしたり、表の列に格納することができます。参照を指定して、オブジェクトを取り出すことができます。

UTL_REF パッケージは、参照ベースの操作をサポートするための PL/SQL プロシージャを提供します。SQL と異なり、UTL_REF プロシージャでは、オブジェクト表名が不明でも汎用型メソッドを書き込むことができます。

要件

このパッケージを使用するには、プロシージャ・オブションが必要です。このパッケージは、SYS（connect internal）によって作成される必要があります。このパッケージが提供する操作は、パッケージ所有者 SYS ではなく、現行のコール・ユーザーのもとで実行されます。

データ型

オブジェクト型は、ユーザーが定義するか、またはライブラリ型として提供されたコンポジット・データ型です。次の構文を使用して、オブジェクト型 `employee_type` を作成できます。

```
CREATE TYPE employee_type AS OBJECT (  
    name    VARCHAR2(20),  
    id      NUMBER,  
  
    member function GET_ID  
        (name VARCHAR2)  
        RETURN MEMBER);
```

オブジェクト型 `employee_type` はユーザー定義型で、`name` と `id` の 2 つの属性、およびメンバー・ファンクションの `GET_ID()` が含まれています。

次の SQL 構文を使用して、オブジェクト表を作成できます。

```
CREATE TABLE employee_table OF employee_type;
```

例外

UTL_REF ファンクションの実行中に、様々な理由で例外が戻る場合があります。たとえば、次の使用例では例外が発生します。

- 選択したオブジェクトが存在しません。これには、次のいずれかの理由が考えられます。
 1. オブジェクトが削除されたか、または指定した参照が無効です。
 2. オブジェクト表が削除されたか、または存在しません。
- シリアライズ可能トランザクションで、オブジェクトを変更またはロックできません。オブジェクトは、シリアライズ可能トランザクションの開始後に、別のトランザクションで変更されました。
- オブジェクトを選択または変更する権限がありません。UTL_REF サブプログラムのコール側は、選択または変更するオブジェクトに対する適切な権限が必要です。

表 64-1 UTL_REF の例外

例外	説明
errnum == 942	権限が不十分です。
errnum == 1031	権限が不十分です。
errnum == 8177	シリアライズ可能トランザクションの場合は、シリアライズ化できません。
errnum == 60	デッドロックが検出されました。
errnum == 1403	データが見つかりません (REF が NULL である場合など)。

UTL_REF パッケージは、名前の付いた例外を定義しません。特定の例外を捕捉して適切に処理するために、ブロックを処理する例外を定義できます。

セキュリティ

UTL_REF パッケージは、サーバー上のストアード PL/SQL プロシージャまたはパッケージとクライアント側の PL/SQL コードからも同様に使用できます。

サーバー上の PL/SQL プロシージャまたはパッケージから起動した場合、UTL_REF は、REF が指すオブジェクトに対する適切なアクセス権限が実行者にあることを検証します。

注意： これは、定義者の権限で操作する、サーバー上の PL/SQL パッケージまたはプロシージャとは対照的で、パッケージ所有者には、必要な操作を実行するための適切な権限が必要です。

したがって、UTL_REF がユーザーの SYS 権限で定義された場合、ユーザー A が参照からオブジェクトを選択するために UTL_REF.SELECT を起動すると、ユーザー A（実行者）は、権限をチェックする必要があります。

UTL_REF は、クライアント側 PL/SQL コードから起動すると、PL/SQL が実行されているクライアント・セッションの権限で操作が行われます。

サブプログラムの要約

表 64-2 UTL_REF サブプログラム

サブプログラム	説明
64-4 ページの SELECT_OBJECT プロシージャ	参照を指定してオブジェクトを選択します。
64-5 ページの LOCK_OBJECT プロシージャ	参照を指定してオブジェクトをロックします。
64-6 ページの UPDATE_OBJECT プロシージャ	参照を指定してオブジェクトを更新します。
64-7 ページの DELETE_OBJECT プロシージャ	参照を指定してオブジェクトを削除します。

SELECT_OBJECT プロシージャ

このプロシージャは、参照を指定してオブジェクトを選択します。選択されたオブジェクトはデータベースから取り出され、その値は PL/SQL の変数 'object' に入れます。このサブプログラムの意味は、次の SQL 文に似ています。

```
SELECT VALUE(t)
INTO object
FROM object_table t
WHERE REF(t) = reference;
```

前述の SQL 文と異なり、このサブプログラムでは、オブジェクトが常駐しているオブジェクト表名を指定する必要はありません。

構文

```
UTL_REF.SELECT_OBJECT (
    reference IN REF "<typename>",
    object    IN OUT "<typename>");
```

パラメータ

表 64-3 SELECT_OBJECT プロシージャのパラメータ

パラメータ	説明
reference	選択または取り出すオブジェクトへの参照。
object	選択したオブジェクトを格納する PL/SQL 変数。この変数は、参照されたオブジェクトと同じオブジェクト型である必要があります。

戻り値

なし

プラグマ

なし

例外

発生する可能性があります。

LOCK_OBJECT プロシージャ

このプロシージャは、参照を指定してオブジェクトをロックします。さらに、このプロシージャでは、プログラムによって、ロックされたオブジェクトを選択できます。このサブプログラムの意味は、次の SQL 文に似ています。

```
SELECT VALUE (t)
  INTO object
  FROM object_table t
 WHERE REF(t) = reference
  FOR UPDATE;
```

前述の SQL 文と異なり、このサブプログラムでは、オブジェクトが常駐しているオブジェクト表名を指定する必要はありません。オブジェクトの更新または削除前に、オブジェクトをロックする必要はありません。

構文

```
UTL_REF.LOCK_OBJECT (
  reference IN REF "<typename>");

UTL_REF.LOCK_OBJECT (
  reference IN REF "<typename>",
  object    IN OUT "<typename>");
```

パラメータ

表 64-4 LOCK_OBJECT プロシージャのパラメータ

パラメータ	説明
reference	ロックするオブジェクトの参照。
object	ロックされたオブジェクトを格納する PL/SQL 変数。この変数は、ロックされたオブジェクトと同じオブジェクト型である必要があります。

戻り値

なし

プラグマ

なし

例外

発生する可能性があります。

UPDATE_OBJECT プロシージャ

このプロシージャは、参照を指定してオブジェクトを更新します。参照されたオブジェクトは、PL/SQL 変数 'object' に含まれている値で更新されます。このサブプログラムの意味は、次の SQL 文に似ています。

```
UPDATE object_table t
SET VALUE(t) = object
WHERE REF(t) = reference;
```

前述の SQL 文と異なり、このサブプログラムでは、オブジェクトが常駐しているオブジェクト表名を指定する必要はありません。

構文

```
UTL_REF.UPDATE_OBJECT (
    reference IN REF "<typename>",
    object    IN    "<typename>");
```

パラメータ

表 64-5 UPDATE_OBJECT プロシージャのパラメータ

パラメータ	説明
reference	更新するオブジェクトの参照。
object	オブジェクトの新規の値を含める PL/SQL 変数。この変数は、更新されたオブジェクトと同じオブジェクト型である必要があります。

戻り値

なし

プラグマ

なし

例外

発生する可能性があります。

DELETE_OBJECT プロシージャ

このプロシージャは、参照を指定してオブジェクトを削除します。このサブプログラムの意味は、次の SQL 文に似ています。

```
DELETE FROM object_table
WHERE REF(t) = reference;
```

前述の SQL 文と異なり、このサブプログラムでは、オブジェクトが常駐しているオブジェクト表名を指定する必要はありません。

構文

```
UTL_REF.DELETE_OBJECT (
    reference IN REF "<typename>");
```

パラメータ

表 64-6 DELETE_OBJECT プロシージャのパラメータ

パラメータ	説明
reference	削除するオブジェクトの参照

戻り値

なし

プラグマ

なし

例外

発生する可能性があります。

例

この例は、次のシナリオを実行するための UTL_REF パッケージの使用法を示しています。会社の従業員が自宅住所の変更を上司に連絡するとします。

... Address_t の宣言など ...

```
CREATE OR REPLACE TYPE Person_t (  
    name    VARCHAR2(64),  
    gender  CHAR(1),  
    address Address_t,  
    MEMBER PROCEDURE setAddress(addr IN Address_t)  
);  
  
CREATE OR REPLACE TYPE BODY Person_t (  
    MEMBER PROCEDURE setAddress(addr IN Address_t) IS  
    BEGIN  
        address := addr;  
    END;  
);
```

```
CREATE OR REPLACE TYPE Employee_t (  

```

Person_t で、REF を使用した Person_t への継承の実装と setAddress の委任をシミュレーションします。

```
    thePerson REF Person_t,  
    empno     NUMBER(5),  
    deptREF   Department_t,  
    mgrREF    Employee_t,  
    reminders StringArray_t,  
    MEMBER PROCEDURE setAddress(addr IN Address_t),  
    MEMBER procedure addReminder(reminder VARCHAR2);  
);
```

```
CREATE TYPE BODY Employee_t (  
    MEMBER PROCEDURE setAddress(addr IN Address_t) IS  
        myMgr Employee_t;  
        meAsPerson Person_t;  
    BEGIN
```

責任を thePerson に委任して、アドレスを更新します。個人オブジェクトを参照からロックして、それをまた選択します。

```
        UTL_REF.LOCK_OBJECT(thePerson, meAsPerson);  
        meAsPerson.setAddress(addr);
```

thePerson に委任します。


```
UTL_REF.UPDATE_OBJECT(thePerson, meAsPerson);  
if mgr is NOT NULL THEN
```

マネージャに覚書きを渡します。

```
    UTL_REF.LOCK_OBJECT(mgr);  
    UTL_REF.SELECT_OBJECT(mgr, myMgr);  
    myMgr.addReminder  
    ('Update address in the employee directory for' ||  
     thePerson.name || ', new address: ' || addr.asString);  
    UTL_REF.UPDATE_OBJECT(mgr, myMgr);  
END IF;  
EXCEPTION  
    WHEN OTHERS THEN  
        errnum := SQLCODE;  
        errmsg := SUBSTR(SQLERRM, 1, 200);
```


UTL_SMTP パッケージは、E メールを送信する目的で設計されています。メール・クライアントが SMTP 方式で E メールを送信するための SMTP サーバーをインプリメントする機能はありません。

SMTP パッケージへのインタフェースの多くが、ファンクションおよびプロシージャとして記述されています。ファンクション形式の場合、クライアントで処理するための応答はサーバーから戻されます。プロシージャ形式の場合、応答は廃棄されますが、一時的なエラー（400 番台の応答コード）または永続的なエラー（500 番台の応答コード）を示す応答のときには例外が発生します。

オリジナルの SMTP プロトコルによる通信には、7 ビット ASCII が使用されることに注意してください。UTL_SMTP を使用すると、すべてのテキスト・データ（つまり、VARCHAR2 型データ）は、サーバーに送信される前に US7ASCII に変換されます。SMTP 拡張 8BITMIME [RFC1652] をサポートしている SMTP サーバーの実装の一部は、クライアントとサーバーの間の完全 8 ビット通信をサポートしています。

注意： RFC ドキュメントとは Request for Comments ドキュメントのこと
で、インターネット上の公開閲覧に関する規格を提案する文書です。実際の RFC ドキュメントは、次の URL を参照してください。

<http://www.ietf.org/rfc/>

DATA コマンドの本体は完全 8 ビットで転送できますが、それ以外の SMTP のコマンドと応答は完全 8 ビットでは転送できません。ターゲットの SMTP サーバーが 8BITMIME 拡張要素をサポートしている場合、マルチバイト・データベースのユーザーは、非 US7ASCII のマルチバイト VARCHAR2 データを RAW に変換し、write_raw_data() API を使用して 8 ビット MIME のコード化を使用したマルチバイト・データを送信できます。

UTL_SMTP では、RFC821 に指定されている SMTP 通信の API を提供していることにも注意してください。メッセージの内容を RFC822 に従ってフォーマットするための API（E メールの件名の設定など）は提供していません。適切なメッセージのフォーマットは、ユーザー各自で行ってください。

注意： このパッケージには、JServer オプションのインストールが必要です。

connection

用途

SMTP 接続を示す PL/SQL レコード型です。

構文

```
TYPE connection IS RECORD (  
    host          VARCHAR2(255),      -- remote host name  
    port          PLS_INTEGER,        -- remote port number  
    private_tcp_con utl_tcp.connection, -- private, for implementation use  
    private_state  PLS_INTEGER        -- private, for implementation use  
);
```

パラメータ

host 接続が確立したときのリモート・ホストの名前。接続が確立されない場合は NULL です。

port 接続したリモート SMTP サーバーのポート番号。接続が確立されない場合は NULL です。

private_tcp_con 実装目的でのみ使用されるプライベートのパラメータ。このフィールドは変更しないでください。

コメント

フィールド `private_tcp_con` は、実装目的にのみ使用されます。

reply、replies

用途

SMTP 応答行を示す PL/SQL レコード型です。各 SMTP 応答行は、応答コードとそれに続くテキスト・メッセージで構成されています。大半の SMTP コマンドに対しては単一の応答行ですが、一部の SMTP コマンドに対しては複数の応答行となります。このような場合は、複数の応答行を示すために応答レコードの PL/SQL 表が使用されます。

構文

```
TYPE reply IS RECORD (  
    code    PLS_INTEGER,      -- 3-digit reply code  
    text    VARCHAR2(508)    -- text message  
);  
TYPE replies IS TABLE OF reply INDEX BY BINARY_INTEGER;  -- multiple reply lines
```

パラメータ

code 3桁の応答コード

text 応答のテキスト・メッセージ

open_connection()

用途

SMTP サーバーへの接続をオープンします。

構文

```
FUNCTION open_connection (host IN  VARCHAR2,  
                          port IN  PLS_INTEGER DEFAULT 25,  
                          c      OUT NOCOPY connection  
)  
RETURN reply;  
FUNCTION open_connection (host IN VARCHAR2,  
                          port IN PLS_INTEGER DEFAULT 25  
)  
RETURN connection;
```

パラメータ

host (IN)

SMTP サーバー・ホストの名前

port (IN)

SMTP サーバーがリスニングするポート番号（通常は 25）

コメント

サーバーからの応答は、ステータス・コード 220 で開始するメッセージになります。

utl_smtp.connection レコードを戻す open_connection() API のバージョンは、実際には、接続が初めて確立された時点で SMTP サーバーが戻す応答コードをチェックする open_connection プロシージャ・バージョンです。

command()、command_replies()

用途

一般的な SMTP コマンドを実行します。

構文

```
FUNCTION command(c      IN connection,
                  cmd    IN VARCHAR2,
                  arg     IN VARCHAR2 DEFAULT NULL) RETURN reply;
PROCEDURE command(c      IN connection,
                  cmd     IN VARCHAR2,
                  arg     IN VARCHAR2 DEFAULT NULL);
FUNCTION command_replies(c      IN connection,
                        cmd     IN VARCHAR2,
                        arg     IN VARCHAR2 DEFAULT NULL) RETURN replies;
```

パラメータ

c (IN) SMTP 接続。

cmd (IN) サーバーに送信する SMTP コマンド。

arg (IN) SMTP 引数へのオプション引数。p_command と p_argument の間には空白が挿入されます。

コメント

これらの API は、一般的な SMTP コマンドを起動します。command() は、単一の応答行が予想される場合にのみ使用します。command_replies() は、応答行が複数になる（つまり、EXPN や HELP）場合に使用します。

command() は、SMTP サーバーからの応答が複数行ある場合、最終応答行のみ戻します。

helo()

用途

接続後に SMTP サーバーとの初期ハンドシェイクを実行します。

構文

```
FUNCTION helo (c IN OUT NOCOPY connection, domain IN VARCHAR2 DEFAULT NULL) RETURN  
reply;  
PROCEDURE helo (c IN OUT NOCOPY connection, domain IN VARCHAR2 DEFAULT NULL);
```

パラメータ

c (IN OUT NOCOPY) SMTP 接続。

domain (IN OUT NOCOPY) ローカル（送信側）ホストのドメイン名。識別の目的で使用されます。

コメント

RFC821 では、接続した後でクライアントはサーバーに対してクライアント自体を識別する必要があると指定されています。このルーチンは、その識別を実行します。このルーチンをコールする前に、open_connection() をコールして接続をオープンしておく必要があります。

サーバーからの応答は、ステータス・コード 250 で開始するメッセージになります。

関連ファンクション

ehlo()

ehlo()

用途

接続後、戻された詳細な情報を使用して SMTP サーバーとの初期ハンドシェイクを実行します。

構文

```
FUNCTION ehlo (c IN OUT NOCOPY connection, domain IN VARCHAR2 DEFAULT NULL)
RETURN replies;
PROCEDURE ehlo (c IN OUT NOCOPY connection, domain IN VARCHAR2 DEFAULT NULL);
```

パラメータ

c (IN OUT NOCOPY) SMTP 接続。

domain (IN OUT NOCOPY) ローカル（送信側）ホストのドメイン名。識別の目的で使われます。

コメント

ehlo() のインタフェースは、サーバーがその構成に関する詳細な情報を戻せることを除くと helo() と同じです。[RFC1869] には、戻される情報のフォーマットが指定されています。フォーマットは、PL/SQL アプリケーションでこのコールのファンクション形式を使用して取り出すことができます。helo() との互換性のために、サーバーが戻すテキストの各行はステータス・コード 250 で開始されます。

関連ファンクション

helo()

mail()

用途

サーバーとのメール・トランザクションを開始します。宛先はメールボックスです。

構文

```
FUNCTION mail (c IN OUT NOCOPY connection, sender IN VARCHAR2, parameters IN
VARCHAR2 DEFAULT NULL) RETURN reply;
PROCEDURE mail (c IN OUT NOCOPY connection, sender IN VARCHAR2, parameters IN
VARCHAR2 DEFAULT NULL);
```


パラメータ

c (IN OUT NOCOPY) SMTP 接続。

sender (IN OUT NOCOPY) メッセージを送信するユーザーの E メール・アドレス。

parameters (IN OUT NOCOPY) [RFC1869] のセクション 6 に定義されている MAIL コマンドへの追加パラメータ。XXX=XXX (XXX=XXX) の形式に従ってください。

コメント

このコマンドはメッセージを送信しません。準備を開始するのみです。トランザクションを完了するには、このコマンドの後に `rcpt()` と `data()` へのコールが必要です。SMTP サーバーへの接続はオープン状態で、`helo()` または `ehlo()` コマンドはすでに送信されている必要があります。

サーバーからの応答は、ステータス・コード 250 で開始するメッセージになります。

rcpt()

用途

E メール・メッセージのレシピエントを指定します。

構文

```
FUNCTION rcpt (c IN OUT NOCOPY connection, recipient IN VARCHAR2,
               parameters IN OUT NOCOPY VARCHAR2 DEFAULT NULL) RETURN reply;
PROCEDURE rcpt (c IN OUT NOCOPY connection, recipient IN VARCHAR2,
                parameters IN VARCHAR2 DEFAULT NULL);
```

c (IN OUT NOCOPY) SMTP 接続。

recipient (IN OUT NOCOPY) メッセージの送信先ユーザーの E メール・アドレス。

parameters (IN OUT NOCOPY) [RFC1869] のセクション 6 に定義されている RCPT コマンドへの追加パラメータ。XXX=XXX (XXX=XXX) の形式に従ってください。

コメント

複数のレシピエントにメッセージを送信するには、このルーチンを複数回コールします。各起動で単一の E メール・アドレスへの配信がスケジュールされます。メッセージ・トランザクションは、先行する `mail()` へのコールによって開始されており、メール・サーバーへの

接続はすでにオープン状態にあり、先行する `open_connection()` および `helo()` または `ehlo()` へのコールによってそれぞれ初期化されている必要があります。

サーバーからの応答は、ステータス・コード 250 または 251 で開始するメッセージになります。

data()

用途

E メール・メッセージの本文を指定します。

構文

```
FUNCTION data (c IN OUT NOCOPY connection, body IN VARCHAR2) RETURN reply;  
PROCEDURE data (c IN OUT NOCOPY connection, body IN VARCHAR2);
```

パラメータ

c (IN OUT NOCOPY) SMTP 接続

body (IN OUT NOCOPY) 送信するメッセージのヘッダーを含めたテキスト ([RFC822] フォーマットで)

コメント

message パラメータの内容が [RFC822] 仕様に準拠していることを、アプリケーションで確認する必要があります。data() ルーチンは、[RFC821] の要件に従って、"<CR><LF>.<CR><LF>" の順序（行の開始はシングル・ピリオド）でメッセージを終了します。また、message 中にある "<CR><LF>.<CR><LF>" の順序（シングル・ピリオド）を "<CR><LF>..

data() は、必ず `open_connection()`、`helo()` または `ehlo()`、`mail()` および `rcpt()` がコールされた後にコールしてください。このルーチンがコールされるときは、SMTP サーバーへの接続がオープン状態で、メール・トランザクションがアクティブ状態である必要があります。

サーバーからの応答は、ステータス・コード 250 で開始するメッセージになります。初期 DATA コマンドから受信した 354 という応答は、コール側に戻されません。

open_data(), write_data(), write_raw_data(), close_data()

用途

これら 3 つの API は、data() API、つまり、SMTP DATA 操作により強力なファイングレイイン・コントロールを提供しています。open_data() は、DATA コマンドを送信します。その後、write_data() で E メール・メッセージの一部を書き込みます。write_data() へのコールを繰り返して、E メール・メッセージにデータを追加します。close_data() は、"<CR><LF>.<CR><LF>" の順序（行の開始はシングル・ピリオド）を送信して、E メール・メッセージを終了します。

構文

```
FUNCTION open_data (c IN OUT NOCOPY connection) RETURN reply;
PROCEDURE open_data (c IN OUT NOCOPY connection);
PROCEDURE write_data (c IN OUT NOCOPY connection, data IN VARCHAR2);
PROCEDURE write_raw_data (c IN OUT NOCOPY connection, data IN RAW);
FUNCTION close_data (c IN OUT NOCOPY connection) RETURN reply;
PROCEDURE close_data (c IN OUT NOCOPY connection);
```

パラメータ

c (IN OUT NOCOPY) SMTP 接続

data (IN OUT NOCOPY) 送信するメッセージのヘッダーを含めたテキストの一部
([RFC822] フォーマットで)

コメント

open_data()、write_data()、write_raw_data() および close_data() へのコールは、正しい順序で実行する必要があります。最初に、SMTP サーバーに DATA コマンドを送信するために、open_data() をコールすると、次に、実際のデータを送信するために、write_data() または write_raw_data() を繰り返しコールできます。データは、close_data() をコールして終了します。open_data() がコールされた後にコールできる API は、write_data()、write_raw_data() または close_data() のみです。他の API へのコールは、INVALID_OPERATION 例外の発生原因となります。

message パラメータの内容が [RFC822] 仕様に準拠していることを、アプリケーションで確認する必要があります。close_data() ルーチンは、[RFC821] の要件に従って、"<CR><LF>.<CR><LF>" 順序で（行の開始はシングル・ピリオド）メッセージを終了します。また、write_data() は、message 中にある "<CR><LF>.<CR><LF>" の順序（シングル・ピリオド）を "<CR><LF>..<CR><LF>" の順序（ダブル・ピリオド）に変換します。この変換によって、[RFC821] のセクション 4.5.2 に記述されている透過性が提供されます。この変換は完全でないことに注意してください。このコードの断片部分を次に示します。

```
utl_smtp.write_data('some message.' || chr(13) || chr(10));  
utl_smtp.write_data('.') || chr(13) || chr(10));
```

順序 "<CR><LF>.<CR><LF>" は、write_data() への 2 回のコールに分割されるため、write_data() の実装によって、データの終了順序は検出されません。したがって、変換も実行されません。このような状態は、ユーザーの責任で処理する必要があります。処理が行われない場合、メッセージ・データの終了が不完全となる可能性があります。

XXX_data() は、必ず open_connection()、helo() または ehlo()、mail() および rcpt() がコールされた後にコールしてください。このルーチンがコールされるときは、SMTP サーバーへの接続がオープン状態で、メール・トランザクションがアクティブ状態である必要があります。

SMTP サーバーからの応答は、close_data() へのコール中にデータの終了が送信されるまで行われないため、write_data() のファンクション形式は存在しないことに注意してください。

write_data() API を使用して送信するテキスト (VARCHAR2) データは、送信前に US7ASCII に変換されます。テキストにマルチバイト・キャラクタが含まれている場合は、テキスト内の US7ASCII に変換できない各マルチバイト・キャラクタが '?' 文字に置き換えられます。EHLO() API を使用して SMTP サーバーとの間で 8BITMIME 拡張要素を処理する場合は、最初に UTL_RAW パッケージでテキストを RAW に変換してから、write_raw_data() を使用して RAW データを送信することによって、マルチバイトの VARCHAR2 データを送信できます。

reset()

用途

現行のメール・トランザクションを異常終了します。

構文

```
FUNCTION reset (c IN OUT NOCOPY connection) RETURN reply;  
PROCEDURE reset (c IN OUT NOCOPY connection);
```

パラメータ

c (IN OUT NOCOPY) SMTP 接続

コメント

このコマンドによって、クライアントは構成処理中のメール・メッセージを放棄できます。メールは送信されません。クライアントは、SMTP サーバーへの接続が `open_connection()` でオープンされた後は、いつでも `rset()` をコールできます。サーバーは、RSET に対してステータス・コード 250 で開始するメッセージで常に応答します。

関連ファンクション

`quit()`

vrfy()

用途

宛先の E メール・アドレスの妥当性を検証します。

構文

```
FUNCTION vrfy (c IN OUT NOCOPY connection, recipient IN VARCHAR2) RETURN reply;
```

パラメータ

c (IN OUT NOCOPY) SMTP 接続

recipient (IN OUT NOCOPY) 検証される E メール・アドレス

コメント

サーバーは、宛先アドレス `recipient` の解決を試みます。解決した場合、サーバーはレシピエントのフルネームと完全修飾メールボックス・パスを返します。この要求の実行前には、`open_connection()` および `helo()` または `ehlo()` を介して、サーバーへの接続が確立されている必要があります。

検証が正常に終了した場合は、ステータス・コード 250 または 251 で開始する 1 行以上の行が戻されます。

関連ファンクション

`expn()`

noop()

用途

NULL のコマンドです。

構文

```
FUNCTION noop (c IN OUT NOCOPY connection) RETURN VARCHAR2;  
PROCEDURE noop (c IN OUT NOCOPY connection);
```

パラメータ

c (IN OUT NOCOPY) SMTP 接続

コメント

このコマンドは、サーバーから正常な応答を引き出すことにのみ有効です。open_connection() でサーバーへの接続を確立した後は、いつでもこのコマンドを発行できます。noop() コマンドは、サーバーが接続され、正しくリスニングが行われていることを検証するために使用します。

このコマンドの応答は、常にステータス・コード 250 で開始する単一の行になります。

quit()

用途

SMTP セッションを終了し、サーバーとの接続を切断します。

構文

```
FUNCTION quit (c IN OUT NOCOPY connection) RETURN VARCHAR2;  
PROCEDURE quit (c IN OUT NOCOPY connection);
```

パラメータ

c (IN OUT NOCOPY) SMTP 接続

コメント

quit() コマンドは、セッションを終了するというクライアントの意図を SMTP サーバーに伝えます。伝えた後に、open_connection() で確立した接続をクローズします。この open_connection() は、このコマンドの実行前にコールされている必要があります。quit() の発行時に、メール・トランザクションが進行中の場合は、reset() と同様に放棄されます。

このコマンドのファンクション形式は、正常終了時にステータス・コード 221 で開始する単一の行を戻します。SMTP サーバーへの接続は、どのような場合にもクローズされます。p_connection のフィールド remote_host と remote_port はリセットされます。

関連ファンクション

reset()

例外

表 65-1 に、UTL_SMTP パッケージの API が呼び出す可能性のある例外を示します。ネットワーク・エラーは、応答コード 421（サービスは使用不可）になります。

表 65-1 UTL_SMTP パッケージの例外

例外	説明
INVALID_OPERATION	無効な操作が行われたときに呼び出されます。つまり、open_data() の後に write_data()、write_raw_data() または close_data() 以外の API をコールした場合、または open_data() をコールしないで、write_data()、write_raw_data() または close_data() をコールした場合です。
TRANSIENT_ERROR	400 番台の応答コードを受け取った場合に呼び出されます。
PERMANENT_ERROR	500 番台の応答コードを受け取った場合に呼び出されます。

制限事項

API による制限または範囲チェックはありません。ただし、SMTP プロトコルの各種要素については、次のサイズ制限に注意する必要があります。制限を超えるデータが送信されると、サーバーからエラーが戻ります。

表 65-2 SMTP のサイズ制限

要素	サイズ制限
ユーザー	ユーザー名の最大合計長は、64 文字です。
ドメイン	ドメイン名またはドメイン番号の最大合計長は、64 文字です。
パス	reverse-path または forward-path の最大合計長は 256 文字です（句読点と要素のセパレータも含む）。
コマンドライン	コマンド自体と <CRLF> も含めたコマンドラインの最大合計長は、512 文字です。
応答行	応答コードと <CRLF> も含めた応答行の最大合計長は、512 文字です。
テキスト行	<CRLF> を含めたテキスト行の最大合計長は、1000 文字です（ただし、透過性に必要な先行ドットはカウントされません）。
レシピエント・バッファ	バッファリングを要するレシピエントの最大合計数は、100 レシピエントです。

応答コード

次のリストに SMTP 応答コードを示します。

表 65-3 SMTP 応答コード

応答コード	意味
211	システム・ステータスまたはシステム・ヘルプの応答です。
214	ヘルプ・メッセージ（使用方法や特定の非標準コマンドの意味に関する情報で、この応答は人間であるユーザーにのみ有用です）。
220	<domain> サービスは準備完了です。
221	<domain> サービスは送信チャネルをクローズしています。
250	要求したメール・アクションは正常で、完了しました。
251	ユーザーがローカルではありません。<forward-path> にフォワードします。
345	メール入力を開始し、<CRLF>.<CRLF> で終了します。

表 65-3 SMTP 応答コード

応答コード	意味
421	<domain> サービスが利用不可のため、送信チャネルをクローズしています（サービスのシャットダウンが必要な場合には、すべてのコマンドに対してこの応答が使用される可能性があります）。
450	要求したメール・アクションが実行されていません。メールボックスは利用不可です（メールボックス・ビジーなど）。
451	要求したアクションが異常終了しました。処理中にローカル・エラーが発生しました。
452	要求したアクションが実行されていません。システム記憶域不足です。
500	構文エラーのため、コマンドを認識できません（コマンドラインが長すぎるなどのエラーも含まれます）。
501	パラメータまたは引数に構文エラーがあります。
502	コマンドがインプリメントされていません。
503	コマンドの順序が不正です。
504	コマンド・パラメータがインプリメントされていません。
550	要求したアクションが実行されていません。メールボックスは利用不可です（メールボックスが見つからない、アクセスできないなど）。
551	ユーザーがローカルではありません。<forward-path> を試行してください。
552	要求したメール・アクションが異常終了しました。記憶域の割当て超過です。
553	要求したアクションが実行されていません。メールボックス名が使用不可です（メールボックスの構文が正しくないなど）。
554	トランザクションに失敗しました。

例

次のコードは、アプリケーションが SMTP パッケージを使用して、E メールを送信する方法を示した例です。ポート 25 で SMTP サーバーに接続し、簡単なテキスト・メッセージを送信します。

```
PROCEDURE send_mail (sender    IN VARCHAR2,
                    recipient IN VARCHAR2,
                    message   IN VARCHAR2)
IS
    mailhost    VARCHAR2(30) := 'mailhost.mydomain.com';
    mail_conn   utl_smtp.connection;

BEGIN
    mail_conn := utl_smtp.open_connection(mailhost, 25);
    utl_smtp.helo(mail_conn, mailhost);
    utl_smtp.mail(mail_conn, sender);
    utl_smtp.rcpt(mail_conn, recipient);
    utl_smtp.data(mail_conn, message);
    utl_smtp.quit(mail_conn);
EXCEPTION
    WHEN OTHERS THEN
        -- Handle the error
END;
```

UTL_TCP は TCP/IP パッケージで、サーバーと外部との TCP/IP ベースの通信をサポートするための PL/SQL プロシージャを提供します。UTL_TCP は、[第 65 章「UTL_SMTP」](#) で説明した SMTP パッケージで使用され、インターネット E メール・プロトコルに関する Oracle Server ベースのクライアントをインプリメントします。

注意： このパッケージには、JServer オプションのインストールが必要です。

TCP/IP パッケージ (UTL_TCP)

connection

用途

TCP/IP 接続を示す PL/SQL レコード型です。

構文

```
TYPE connection IS RECORD (  
    remote_host    VARCHAR2(255), -- remote host name  
    remote_port    PLS_INTEGER,   -- remote port number  
    local_host     VARCHAR2(255), -- local host name  
    local_port     PLS_INTEGER,   -- local port number  
    charset        VARCHAR2(30),  -- character set for on-the-wire communication  
    newline        VARCHAR2(2),   -- newline character sequence  
    private_sd      PLS_INTEGER,   -- private, for implementation use  
    private_bf      RAW(32767),    -- private, for implementation use  
    private_bfsz    PLS_INTEGER,   -- private, for implementation use  
    private_pos     PLS_INTEGER,   -- private, for implementation use  
    private_end     PLS_INTEGER,   -- private, for implementation use  
    private_mkpos   PLS_INTEGER    -- private, for implementation use  
);
```

パラメータ

remote_host 接続が確立したときのリモート・ホストの名前。接続が確立されない場合は NULL です。

remote_port 接続したリモート・ホストのポート番号。接続が確立されない場合は NULL です。

local_host 接続の確立に使用されるローカル・ホスト名。接続が確立されない場合は NULL です。

local_port 接続の確立に使用されるローカル・ホストのポート番号。接続が確立されない場合は NULL です。

charset 回線用キャラクタ・セット。データベース内のテキスト・メッセージは、回線上とは異なるキャラクタ・セット（通信プロトコルで指定されているキャラクタ・セットや他の通信上の終端で規定されているキャラクタ・セットなど）にコード化されている可能性がある

ります。したがって、ネットワーク上での送受信時には、データベース内のテキスト・メッセージと回線用キャラクタ・セットの間の変換が行われます。

newline 改行文字列の順序。この改行文字列の順序は、`write_line()` API で送信されるテキスト行に追加されます。

private_sd、private_bf、private_bfsz、private_pos、private_end、private_mkpos 実装目的で使用するプライベートのパラメータ。これらのフィールドは変更しないでください。

コメント

フィールド `private_XXXX` は、実装目的にのみ使用されます。

CRLF

用途

CR-LF 改行文字の順序。多くの通信規格で一般的に使用されている改行の順序です。

構文

```
CRLF constant varchar2(10) := chr(13) || chr(10);
```

コメント

このパッケージ定数によって、多くのインターネット・プロトコルで一般的に使用されている改行文字列の順序を定義します。これは `write_line()` に対する改行文字列順序のデフォルト値で、接続のオープン時に指定されます。このようなプロトコルでは <CR><LF> を使用して新しい行を示しますが、一部の実装では <LF> のみの使用を選択できます。この場合、ユーザーは、接続のオープン時に異なる改行文字列順序を指定できます。

open_connection()

用途

指定したサービスへの TCP/IP 接続をオープンします。

構文

```
FUNCTION open_connection (remote_host      IN VARCHAR2,  
                          remote_port     IN PLS_INTEGER,  
                          local_host       IN VARCHAR2 DEFAULT NULL,  
                          local_port      IN PLS_INTEGER DEFAULT NULL,  
                          in_buffer_size  IN PLS_INTEGER DEFAULT NULL,  
                          out_buffer_size IN PLS_INTEGER DEFAULT NULL,  
                          charset         IN VARCHAR2 DEFAULT NULL,  
                          newline         IN VARCHAR2 DEFAULT CRLF) RETURN  
connection;
```

パラメータ

remote_host (IN) サービスを提供するホストの名前。remote_host が NULL の場合は、ローカル・ホストに接続します。

remote_port (IN) サービスが接続をリスニングするポート番号。

local_host (IN) サービスを提供するホストの名前。NULL の場合は無視されます。

local_port (IN) サービスが接続をリスニングするポート番号。NULL の場合は無視されません。

in_buffer_size (IN) 入力バッファのサイズ。入力バッファを使用すると、サーバーからデータを受信する際の実行パフォーマンスを高速化できます。バッファの適正サイズは、クライアントとサーバーの間のデータの流れやネットワーク条件によって異なります。NULL または 0 の値は、バッファを使用しないことを意味します。入力バッファの最大サイズは、32767 バイトです。

out_buffer_size (IN) 出力バッファのサイズ。出力バッファを使用すると、サーバーにデータを送信する際の実行パフォーマンスを高速化できます。バッファの適正サイズは、クライアントとサーバーの間のデータの流れやネットワーク条件によって異なります。NULL または 0 の値は、バッファを使用しないことを意味します。出力バッファの最大サイズは、32767 バイトです。

charset (IN) 回線用キャラクタ・セット。データベース内のテキスト・メッセージは、回線上とは異なるキャラクタ・セット（通信プロトコルで指定されているキャラクタ・セットや他の通信上の終端で規定されているキャラクタ・セットなど）にコード化されている可能性があります。したがって、ネットワーク上での送受信時には、read_text()、read_line()、write_text() および write_line() を使用して、データベース内のテキスト・メッセージと回線用キャラクタ・セットの間の変換が行われます。変換が不要な場合は、このパラメータを NULL に設定してください。

newline (IN) 改行文字列の順序。この改行文字列の順序は、`write_line()` API で送信されるテキスト行に追加されます。

コメント

この UTL_TCP パッケージでオープンした接続は、オープン状態のまま、あるデータベース・コールから別のデータベース・コールに MTS 構成内で渡すことができます。ただし、接続は明示的にクローズする必要があります。接続を格納している PL/SQL レコード変数が PL/SQL プログラムで無効になると、接続はオープン状態のままになります。不要な接続を適切にクローズしないと、ローカルおよびリモートのシステム・リソースが無駄になります。

関連ファンクション

`close_connection()`、`close_all_connections()`

available()

用途

TCP/IP 接続からの読み込みに使用可能なバイト数を決定します。この数は、ブロッキングなしで即時に読み込むことができるバイト数です。接続からデータを読み込めるかどうかを判断します。

構文

```
FUNCTION available (c IN OUT NOCOPY connection) RETURN PLS_INTEGER;
```

パラメータ

c (IN OUT NOCOPY) 読み込み可能なデータ量を判断する TCP 接続

コメント

`open_connection()` へのコールを介して、接続がすでにオープン状態になっている必要があります。ユーザーはこの API を使用して、読み込み API をコールする前に、データが読み込み可能かどうかを確認できるため、入力からデータを読み込む準備が整っていないことでプログラムがブロックされることはありません。

関連ファンクション

`read_raw()`、`read_text()`、`read_line()`

read_raw()

用途

オープン接続中のサービスからバイナリ・データを受信します。

構文

```
FUNCTION read_raw(c      IN OUT NOCOPY connection,
                  data IN OUT NOCOPY RAW,
                  len  IN              PLS_INTEGER DEFAULT 1,
                  peek IN              BOOLEAN      DEFAULT FALSE)
RETURN PLS_INTEGER;
```

パラメータ

c (IN OUT NOCOPY) データの受信元 TCP 接続。

data (IN OUT COPY) 受信データ。

len (IN) 受信するデータのバイト数。

peek (IN) 通常、ユーザーはデータを読み込むとそのデータを入力キューから削除、つまり消費しようとしめます。しかし、入力キューからデータを削除しないで、単にデータを前もって確認、つまり覗くだけにし、次回コール時にも依然としてそのデータが読み込み可能な（または覗ける）状態にしておくことが望まれる場合もあります。入力キューにデータを保持するには、このフラグを **TRUE** に設定し、接続のオープン時に入力バッファをセットアップする必要があります。覗く（つまり、読み込んでも入力キューに保持）ことができるデータ量は、入力バッファ・サイズより小さくしてください。

戻り値 実際に受信したデータのバイト数。

コメント

`open_connection()` へのコールを介して、接続がすでにオープン状態になっている必要があります。このファンクションは、指定した文字数が読み込まれるまで、または入力の最後に達するまで戻りません。テキスト・メッセージは、コール側に戻される前に、接続のオープン時に指定した回線用キャラクタ・セットからデータベース・キャラクタ・セットに変換されます。

関連ファンクション

`read_text()`、`read_line()`、`available()`

write_raw()

用途

オープン接続中のサービスにバイナリ・メッセージを送信します。

構文

```
FUNCTION write_raw(c      IN OUT NOCOPY connection,
                  data IN      RAW,
                  len  IN      PLS_INTEGER DEFAULT NULL)
RETURN PLS_INTEGER;
```

c (IN OUT NOCOPY) データの送信先 TCP 接続。

data (IN) 送信するデータを含んだバッファ。

len (IN) 送信するデータのバイト数。len が NULL の場合は、データの長さ全体が書き込まれます。実際に書き込まれるデータ量は、ネットワーク条件のために削減される可能性があります。

戻り値 実際に送信したデータのバイト数。

コメント

open_connection() へのコールを介して、接続がすでにオープン状態になっている必要があります。

関連ファンクション

write_text()、write_line()、flush()

read_text()

用途

オープン接続中のサービスからテキスト・データを受け取ります。

構文

```
FUNCTION read_text(c      IN OUT NOCOPY connection,
                  data IN OUT NOCOPY VARCHAR2,
                  len  IN      PLS_INTEGER DEFAULT 1,
                  peek  IN      BOOLEAN      DEFAULT FALSE) RETURN PLS_INTEGER;
```

c (IN OUT NOCOPY) データの受信元 TCP 接続。

data (IN OUT NOCOPY) 受信データ。

len (IN) 受信するデータのバイト数。

peek (IN) 通常、ユーザーはデータを読み込むとそのデータを入力キューから削除、つまり消費しようとしています。しかし、入力キューからデータを削除しないで、単にデータを前もって確認、つまり覗くだけにし、次回コール時にも依然としてそのデータが読み込み可能な（または覗ける）状態にしておくことが望まれる場合もあります。入力キューにデータを保持するには、このフラグを **TRUE** に設定し、接続のオープン時に入力バッファをセットアップする必要があります。覗く（つまり、読み込んでも入力キューに保持）ことができるデータ量は、入力バッファ・サイズより小さくしてください。

戻り値 実際に受信したデータの文字数。

コメント

`open_connection()` へのコールを介して、接続がすでにオープン状態になっている必要があります。このファンクションは、指定したバイト数が読み込まれるまで、または入力 of 最後に達するまで戻りません。テキスト・メッセージは、コール側に戻される前に、接続のオープン時に指定した回線用キャラクタ・セットからデータベース・キャラクタ・セットに変換されます。

関連ファンクション

`read_raw()`、`read_line()`、`available()`

write_text()

用途

オープン接続中のサービスにテキスト・メッセージを送信します。

構文

```
FUNCTION write_text(c      IN OUT NOCOPY connection,
                    data IN          VARCHAR2,
                    len  IN          PLS_INTEGER DEFAULT NULL)
RETURN PLS_INTEGER;
```

c (IN OUT NOCOPY) データの送信先 TCP 接続。

data (IN) 送信するデータを含んだバッファ。

len (IN) 送信するデータの文字数。len が NULL の場合は、データの長さ全体が書き込まれます。実際に書き込まれるデータ量は、ネットワーク条件のために削減される可能性があります。

戻り値 実際に送信したデータの文字数。

コメント

open_connection() へのコールを介して、接続がすでにオープン状態になっている必要があります。テキスト・メッセージは、回線に送信される前に、接続のオープン時に指定した回線用キャラクタ・セットに変換されます。

関連ファンクション

write_raw()、write_line()、flush()

read_line()

用途

オープン接続中のサービスからテキスト行を受け取ります。行は、LF 改行、CR 改行または CR-LF 改行（CR の後に LF が続く）で終了します。

構文

```
FUNCTION read_line(c          IN OUT NOCOPY connection,
                   data       IN OUT NOCOPY VARCHAR2,
                   remove_crlf IN          BOOLEAN DEFAULT FALSE,
                   peek        IN          BOOLEAN DEFAULT FALSE)
RETURN PLS_INTEGER;
```

c (IN OUT NOCOPY) データの受信元 TCP 接続。

data (IN OUT NOCOPY) 受信データ。

remove_crlf (IN) TRUE の場合は、後に続く CR または LF 文字が受信メッセージから削除されます。

peek (IN) 通常、ユーザーはデータを読み込むとそのデータを入力キューから削除、つまり消費しようとしています。しかし、入力キューからデータを削除しないで、単にデータを前もって確認、つまり覗くだけにし、次回コール時にも依然としてそのデータが読み可能な（または覗ける）状態にしておくことが望まれる場合もあります。入力キューにデータを保持するには、このフラグを TRUE に設定し、接続のオープン時に入力バッファをセットアップする必要があります。覗く（つまり、読み込んでも入力キューに保持）ことができるデータ量は、入力バッファ・サイズより小さくしてください。

戻り値 実際に受信したデータの文字数。

コメント

`open_connection()` へのコールを介して、接続がすでにオープン状態になっている必要があります。このファンクションは、行の終わりまたは入力最後に達するまで戻りません。テキスト・メッセージは、コール側に戻される前に、接続のオープン時に指定した回線用キャラクタ・セットからデータベース・キャラクタ・セットに変換されます。

関連ファンクション

`read_raw()`、`read_text()`、`available()`

write_line()

用途

オープン接続中のサービスにテキスト行を送信します。送信前に、改行文字の順序がメッセージに追加されます。

構文

```
FUNCTION write_line(c      IN OUT NOCOPY connection,
                   data IN          VARCHAR2 DEFAULT NULL)
RETURN PLS_INTEGER;
```

c (IN OUT NOCOPY) データの送信先 TCP 接続

data (IN) 送信するデータを含んだバッファ

戻り値 実際に送信したデータの文字数

コメント

`open_connection()` へのコールを介して、接続がすでにオープン状態になっている必要があります。テキスト・メッセージは、回線に送信される前に、接続のオープン時に指定した回線用キャラクタ・セットに変換されます。

関連ファンクション

`write_raw()`、`write_text()`、`flush()`

`get_raw()`、`get_text()`、`get_line()`

用途

読み込みファンクションの簡便な形式で、読み込みデータ量ではなく読み込んだデータを戻します。

構文

```
FUNCTION get_raw(c      IN OUT NOCOPY connection,
                 len    IN              PLS_INTEGER DEFAULT 1,
                 peek   IN              BOOLEAN      DEFAULT FALSE) RETURN RAW;

FUNCTION get_text(c     IN OUT NOCOPY connection,
                  len    IN              PLS_INTEGER DEFAULT 1,
                  peek   IN              BOOLEAN      DEFAULT FALSE) RETURN VARCHAR2;

FUNCTION get_line(c     IN OUT NOCOPY connection,
                  remove_crlf IN         BOOLEAN DEFAULT false,
                  peek    IN              BOOLEAN DEFAULT FALSE) RETURN VARCHAR2;
```

c (IN OUT NOCOPY) データの受信元 TCP 接続。

len (IN) 受信するデータのバイト数（または VARCHAR2 の文字数）。デフォルトは 1 です。

peek (IN) 通常、ユーザーはデータを読み込むとそのデータを入力キューから削除、つまり消費しようとしています。しかし、入力キューからデータを削除しないで、単にデータを前もって確認、つまり覗くだけにし、次回コール時にも依然としてそのデータが読み込み可能な（または覗ける）状態にしておくことが望まれる場合もあります。入力キューにデータを保持するには、このフラグを TRUE に設定し、接続のオープン時に入力バッファをセットアップする必要があります。覗く（つまり、読み込んだでも入力キューに保持）ことができるデータ量は、入力バッファ・サイズより小さくしてください。

remove_crlf (IN) TRUE の場合は、後に続く CR または LF 文字が受信メッセージから削除されます。

コメント

`open_connection()` へのコールを介して、接続がすでにオープン状態になっている必要があります。

関連ファンクション

`read_raw()`、`read_text()`、`read_line()`

flush()

用途

バッファが使用されている場合は、出力バッファの全データをサーバーに即時送信します。

構文

```
PROCEDURE flush (c IN OUT NOCOPY connection);
```

パラメータ

c (IN OUT NOCOPY) データの送信先 TCP 接続

コメント

`open_connection()` へのコールを介して、接続がすでにオープン状態になっている必要があります。

関連ファンクション

`write_raw()`、`write_text()`、`write_line()`

close_connection()

用途

オープン状態の TCP/IP 接続をクローズします。

構文

```
PROCEDURE close_connection (c IN OUT NOCOPY connection);
```

パラメータ

c (IN OUT NOCOPY) クローズする TCP 接続

コメント

接続は、`open_connection()` へのコールによって事前にオープンされている必要があります。`c` のフィールド `remote_host`、`remote_port`、`local_host`、`local_port` および `charset` は、接続がクローズされた後にリセットされます。

オープン状態の接続は、明示的にクローズする必要があります。オープン状態の接続は、接続を格納している PL/SQL レコード変数が PL/SQL プログラムで無効になると、そのままオープン状態となります。不要な接続を適切にクローズしないと、ローカルおよびリモートのシステム・リソースが無駄になります。

`close_all_connections()`

用途

オープン状態の TCP/IP 接続をすべてクローズします。

構文

```
PROCEDURE close_all_connections;
```

パラメータ

なし

コメント

このコールは、PL/SQL プログラムが参照先のない接続を回避する前に、すべての接続をクローズするために用意されています。

関連ファンクション

`open_connection()`、`close_connection()`

例外

表 66-1 に、TCP/IP パッケージで発生する例外を示します。

表 66-1 TCP/IP の例外

例外	説明
BUFFER_TOO_SMALL	事前確認に必要な入力バッファが小さすぎます。
END_OF_INPUT	接続先から読み込み可能なデータがこれ以上ない場合に発生します。
NETWORK_ERROR	一般的なネットワーク・エラーです。
BAD_ARGUMENT	不正な引数が API コールに渡されました（つまり、負のバッファ・サイズ）。

例

次の例は、HTTP を介して Web ページを取り出すときの TCP/IP パッケージの使用方法を示したコードです。ポート 80（HTTP の標準ポート）でリスニングする Web サーバーに接続し、ルート・ドキュメントを要求します。

```
DECLARE
  c utl_tcp.connection; -- TCP/IP connection to the Web server
BEGIN
  c := utl_tcp.open_connection('www.acme.com', 80); -- open connection

  utl_tcp.write_line(c, 'GET / HTTP/1.0');          -- send HTTP request
  utl_tcp.write_line(c);

  BEGIN
    LOOP
      dbms_output.put_line(utl_tcp.get_line(c, TRUE)); -- read result
    END LOOP;
  EXCEPTION
    WHEN utl_tcp.end_of_input THEN
      NULL; -- end of input
  END;
  utl_tcp.close_connection(c);
END;
```

次の例は、アプリケーションが TCP/IP パッケージを使用して、E メールを送信する方法を示したコードです。ポート 25 で SMTP サーバーに接続し、簡単なテキスト・メッセージを送信します。

```
PROCEDURE send_mail (sender    IN VARCHAR2,
                    recipient IN VARCHAR2,
                    message   IN VARCHAR2)
```



```

IS
    mailhost    VARCHAR2(30) := 'mailhost.mydomain.com';
    smtp_error  EXCEPTION;
    mail_conn   utl_tcp.connection;
    PROCEDURE smtp_command(command IN VARCHAR2,
                           ok      IN VARCHAR2 DEFAULT '250')
    IS
        response varchar2(3);
    BEGIN
        utl_tcp.write_line(mail_conn, command);
        response := substr(utl_tcp.get_line(mail_conn), 1, 3);
        IF (response <> ok) THEN
            RAISE smtp_error;
        END IF;
    END;

BEGIN
    mail_conn := utl_tcp.open_connection(mailhost, 25);
    smtp_command('HELO ' || mailhost);
    smtp_command('MAIL FROM: ' || sender);
    smtp_command('RCPT TO: ' || recipient);
    smtp_command('DATA', '354');
    smtp_command(message);
    smtp_command('QUIT', '221');
    utl_tcp.close_connection(mail_conn);
EXCEPTION
    WHEN OTHERS THEN
        -- Handle the error
END;

```


A

available()
 UTL_TCP のファンクション, 66-5

C

Calendar パッケージ, 1-13
catproc.sql スクリプト, 1-2
CLOB データ型
 NCLOB, 18-3
close_all_connections()
 UTL_TCP のファンクション, 66-13
close_connection()
 UTL_TCP のファンクション, 66-12
command(), command_replies()
 UTL_SMTP のファンクション, 65-4
connection
 UTL_SMTP のファンクション, 65-2
 UTL_TCP のファンクション, 66-2
CREATE PACKAGE BODY コマンド, 1-3
CREATE PACKAGE コマンド, 1-3
CRLF
 UTL_TCP のファンクション, 66-3

D

data()
 UTL_SMTP のファンクション, 65-8
DBA_REPCATALOG ビュー
 削除, 37-69
DBA_REPCOLUMN_GROUP ビュー
 更新, 37-26
DBA_REPGROUP ビュー
 更新, 37-28

DBA_REPOBJECT ビュー
 更新, 37-29
DBA_REPPRIORITY_GROUP ビュー
 更新, 37-27
DBA_REPRESOLUTION_STATISTICS ビュー
 削除, 37-70
DBA_REPRESOLUTION ビュー
 更新, 37-33
DBA_REPSITES ビュー
 更新, 37-31
DBMS_ALERT パッケージ, 2-1
DBMS_APPLICATION_INFO パッケージ, 3-2
DBMS_AQADM パッケージ, 5-1
DBMS_AQ パッケージ, 4-1
DBMS_BACKUP_RESTORE パッケージ, 6-1
DBMS_DDL パッケージ, 7-1
DBMS_DEBUG パッケージ, 8-1
DBMS_DEFER_QUERY パッケージ, 10-1, 10-2
 GET_ARG_FORM ファンクション, 10-3
 GET_ARG_TYPE ファンクション, 10-4
 GET_CALL_ARGS プロシージャ, 10-6
 GET_CHAR_ARG プロシージャ, 10-7
 GET_datatype_ARG ファンクション, 10-7
 GET_NUMBER_ARG プロシージャ, 10-7
 GET_RAW_ARG プロシージャ, 10-7
 GET_ROWID_ARG プロシージャ, 10-7
 GET_VARCHAR2_ARG プロシージャ, 10-7
DBMS_DEFER_SYS パッケージ, 11-1, 11-2
 ADD_DEFAULT_DEST プロシージャ, 11-4
 DELETE_DEF_DESTINATION プロシージャ, 11-5
 DELETE_DEFAULT_DEST プロシージャ, 11-4
 DELETE_ERROR プロシージャ, 11-5
 DELETE_TRAN プロシージャ, 11-6, 11-7, 11-9
DISABLED ファンクション, 11-7
EXCLUDE_PUSH ファンクション, 11-8

EXECUTE_ERROR_AS_USER プロシージャ, 11-10
EXECUTE_ERROR プロシージャ, 11-9
PURGE ファンクション, 11-11
PUSH ファンクション, 11-13
REGISTER_PROPAGATOR プロシージャ, 11-16
SCHEDULE_EXECUTION プロシージャ, 11-19
SCHEDULE_PURGE プロシージャ, 11-17
SCHEDULE_PUSH プロシージャ, 11-19
SET_DISABLED プロシージャ, 11-21
UNREGISTER_PROPAGATOR プロシージャ,
11-22
UNSCHEDULE_PURGE プロシージャ, 11-23
UNSCHEDULE_PUSH プロシージャ, 11-23
DBMS_DEFER パッケージ, 9-1, 9-2
CALL プロシージャ, 9-3
CHAR_ARG プロシージャ, 9-5
COMMIT_WORK プロシージャ, 9-4
datatype_ARG プロシージャ, 9-5
DATE_ARG プロシージャ, 9-5
NUMBER_ARG プロシージャ, 9-5
RAW_ARG プロシージャ, 9-5
ROWID_ARG プロシージャ, 9-5
TRANSACTION プロシージャ, 9-7
VARCHAR2_ARG プロシージャ, 9-5
DBMS_DESCRIBE パッケージ, 12-1
DBMS_DISTRIBUTED_TRUST_ADMIN パッケージ,
13-1
DBMS_HS_PASSTHROUGH パッケージ, 15-1
DBMS_HS パッケージ, 14-1
DBMS_IOT パッケージ, 16-1
DBMS_JOB パッケージ, 17-1
インスタンス親和性, 17-12
DBMS_LOB パッケージ, 18-1
DBMS_LOCK パッケージ, 19-1
DBMS_LOGMNR_D パッケージ, 21-1
DBMS_LOGMNR パッケージ, 20-1
DBMS_MVIEW パッケージ, 22-1
DBMS_SNAPSHOT パッケージ, 22-1, 48-1
DBMS_OBFUSCATION_TOOLKIT パッケージ, 23-1
DBMS_OFFLINE_OG パッケージ, 24-1, 24-2
BEGIN_INSTANTIATION プロシージャ, 24-3
BEGIN_LOAD プロシージャ, 24-4
END_INSTANTIATION プロシージャ, 24-5
END_LOAD プロシージャ, 24-7
RESUME_SUBSET_OF_MASTERS プロシージャ,
24-8
DBMS_OFFLINE_SNAPSHOT パッケージ, 25-1, 25-2
BEGIN_LOAD プロシージャ, 25-3
END_LOAD プロシージャ, 25-5
DBMS_OLAP パッケージ, 26-1
DBMS_ORACLE_TRACE_AGENT パッケージ, 27-1
DBMS_ORACLE_TRACE_USER パッケージ, 28-1
DBMS_OUTPUT パッケージ, 29-1
DBMS_PCLXUTIL パッケージ, 30-1
DBMS_PIPE パッケージ, 31-1
DBMS_PROFILER パッケージ, 32-1
DBMS_RANDOM パッケージ, 33-1
DBMS_RECTIFIER_DIFF パッケージ, 34-1, 34-2
DIFFERENCES プロシージャ, 34-3
RECTIFY プロシージャ, 34-6
DBMS_REFRESH パッケージ, 35-1, 35-2
ADD プロシージャ, 35-3
CHANGE プロシージャ, 35-4
DESTROY プロシージャ, 35-6
MAKE プロシージャ, 35-7
REFRESH プロシージャ, 35-9
SUBTRACT プロシージャ, 35-10
DBMS_REPAIR パッケージ, 36-1
DBMS_REPCAT_ADMIN パッケージ, 38-1, 38-2
GRANT_ADMIN_ANY_SCHEMA プロシージャ,
38-3
GRANT_ADMIN_SCHEMA プロシージャ, 38-3
REGISTER_USER_REPGROUP プロシージャ, 38-4
REVOKE_ADMIN_ANY_SCHEMA プロシージャ,
38-6
REVOKE_ADMIN_SCHEMA プロシージャ, 38-7
UNREGISTER_USER_REPGROUP プロシージャ,
38-8
DBMS_REPCAT_INSTANTIATE パッケージ, 39-1,
39-2
DROP_SITE_INSTANTIATION プロシージャ, 39-3
INSTANTIATE_OFFLINE_REPAPI ファンクシ
ョン, 39-6
INSTANTIATE_OFFLINE ファンクション, 39-3
INSTANTIATE_ONLINE ファンクション, 39-9
DBMS_REPCAT_RGT パッケージ, 40-1, 40-2
ALTER_REFRESH_TEMPLATE プロシージャ, 40-5
ALTER_TEMPLATE_OBJECT プロシージャ, 40-6
ALTER_TEMPLATE_PARM プロシージャ, 40-9
ALTER_USER_AUTHORIZATION プロシージャ,
40-11
ALTER_USER_PARM_VALUE プロシージャ,
40-12
COMPARE_TEMPLATES ファンクション, 40-15

COPY_TEMPLATE ファンクション, 40-16
 CREATE_OBJECT_FROM_EXISTING ファンクション, 40-18
 CREATE_REFRESH_TEMPLATE ファンクション, 40-20
 CREATE_TEMPLATE_OBJECT ファンクション, 40-22
 CREATE_TEMPLATE_PARM ファンクション, 40-26
 CREATE_USER_AUTHORIZATION ファンクション, 40-28
 CREATE_USER_PARM_VALUE ファンクション, 40-29
 DELETE_RUNTIME_PARMS プロシージャ, 40-32
 DROP_ALL_OBJECTS プロシージャ, 40-33
 DROP_ALL_TEMPLATE_PARMS プロシージャ, 40-34
 DROP_ALL_TEMPLATE_SITES プロシージャ, 40-35
 DROP_ALL_TEMPLATES プロシージャ, 40-36
 DROP_ALL_USER_AUTHORIZATIONS プロシージャ, 40-36
 DROP_ALL_USER_PARM_VALUES プロシージャ, 40-37
 DROP_REFRESH_TEMPLATE プロシージャ, 40-39
 DROP_SITE_INSTANTIATION プロシージャ, 40-40
 DROP_TEMPLATE_OBJECT プロシージャ, 40-41
 DROP_TEMPLATE_PARM プロシージャ, 40-43
 DROP_USER_AUTHORIZATION プロシージャ, 40-44
 DROP_USER_PARM_VALUE プロシージャ, 40-45
 GET_RUNTIME_PARM_ID ファンクション, 40-46
 INSERT_RUNTIME_PARMS プロシージャ, 40-47
 INSTITUTE_OFFLINE_REPAPI ファンクション, 40-51
 INSTITUTE_OFFLINE ファンクション, 40-49
 INSTITUTE_ONLINE ファンクション, 40-54
 LOCK_TEMPLATE_EXCLUSIVE プロシージャ, 40-57
 LOCK_TEMPLATE_SHARED プロシージャ, 40-57
 DBMS_REPCAT パッケージ, 37-1, 37-2
 ADD_DELETE_RESOLUTION プロシージャ, 37-12
 ADD_GROUPED_COLUMN プロシージャ, 37-6
 ADD_MASTER_DATABASE プロシージャ, 37-7
 ADD_PRIORITY_CHAR プロシージャ, 37-9
 ADD_PRIORITY_datatype プロシージャ, 37-9
 ADD_PRIORITY_DATE プロシージャ, 37-9
 ADD_PRIORITY_NUMBER プロシージャ, 37-9
 ADD_PRIORITY_VARCHAR2 プロシージャ, 37-9
 ADD_SITE_PRIORITY_SITE プロシージャ, 37-10
 ADD_UNIQUENESS_RESOLUTION プロシージャ, 37-12
 ADD_UPDATE_RESOLUTION プロシージャ, 37-12
 ALTER_MASTER_PROPAGATION プロシージャ, 37-16
 ALTER_MASTER_REPOBJECT プロシージャ, 37-17
 ALTER_PRIORITY_CHAR プロシージャ, 37-20
 ALTER_PRIORITY_datatype プロシージャ, 37-20
 ALTER_PRIORITY_DATE プロシージャ, 37-20
 ALTER_PRIORITY_NUMBER プロシージャ, 37-20
 ALTER_PRIORITY_RAW プロシージャ, 37-20
 ALTER_PRIORITY プロシージャ, 37-19
 ALTER_SITE_PRIORITY_SITE プロシージャ, 37-23
 ALTER_SITE_PRIORITY プロシージャ, 37-22
 ALTER_SNAPSHOT_PROPAGATION プロシージャ, 37-24
 CANCEL_STATISTICS プロシージャ, 37-25
 COMMENT_ON_COLUMN_GROUP プロシージャ, 37-26
 COMMENT_ON_DELETE_RESOLUTION プロシージャ, 37-33
 COMMENT_ON_PRIORITY_GROUP プロシージャ, 37-27
 COMMENT_ON_REPGROUP プロシージャ, 37-28
 COMMENT_ON_REPOBJECT プロシージャ, 37-29
 COMMENT_ON_REPSITES プロシージャ, 37-31
 COMMENT_ON_SITE_PRIORITY プロシージャ, 37-27
 COMMENT_ON_SNAPSHOT_REPSITES プロシージャ, 37-32
 COMMENT_ON_UNIQUE_RESOLUTION プロシージャ, 37-33
 COMMENT_ON_UPDATE_RESOLUTION プロシージャ, 37-33
 COMPARE_OLD_VALUES プロシージャ, 37-35
 CREATE_MASTER_REPGROUP プロシージャ, 37-37
 CREATE_MASTER_REPOBJECT プロシージャ, 37-38
 CREATE_SNAPSHOT_REPGROUP プロシージャ, 37-41

CREATE_SNAPSHOT_REPOBJECT プロシージャ, 37-42
DEFINE_COLUMN_GROUP プロシージャ, 37-44
DEFINE_PRIORITY_GROUP プロシージャ, 37-45
DEFINE_SITE_PRIORITY プロシージャ, 37-47
DO_DEFERRED_REPCAT_ADMIN プロシージャ, 37-48
DROP_COLUMN_GROUP プロシージャ, 37-49
DROP_DELETE_RESOLUTION プロシージャ, 37-61
DROP_GROUPED_COLUMN プロシージャ, 37-50
DROP_MASTER_REPGROUP プロシージャ, 37-51
DROP_MASTER_REPOBJECT プロシージャ, 37-52
DROP_PRIORITY_CHAR プロシージャ, 37-55
DROP_PRIORITY_datatype プロシージャ, 37-55
DROP_PRIORITY_DATE プロシージャ, 37-55
DROP_PRIORITY_GROUP プロシージャ, 37-54
DROP_PRIORITY_NUMBER プロシージャ, 37-55
DROP_PRIORITY_VARCHAR2 プロシージャ, 37-55
DROP_PRIORITY プロシージャ, 37-53
DROP_SITE_PRIORITY_SITE プロシージャ, 37-58
DROP_SITE_PRIORITY プロシージャ, 37-57
DROP_SNAPSHOT_REPGROUP プロシージャ, 37-59
DROP_SNAPSHOT_REPOBJECT プロシージャ, 37-60
DROP_UNIQUE_RESOLUTION プロシージャ, 37-61
DROP_UPDATE_RESOLUTION プロシージャ, 37-61
EXECUTE_DDL プロシージャ, 37-63
GENERATE_REPLICATION_SUPPORT プロシージャ, 37-64
GENERATE_SNAPSHOT_SUPPORT プロシージャ, 37-66
MAKE_COLUMN_GROUP プロシージャ, 37-68
PURGE_MASTER_LOG プロシージャ, 37-69
PURGE_STATISTICS プロシージャ, 37-70
REFRESH_SNAPSHOT_REPGROUP プロシージャ, 37-71
REGISTER_SNAPSHOT_REPGROUP プロシージャ, 37-72
REGISTER_STATISTICS プロシージャ, 37-74
RELOCATE_MASTERDEF プロシージャ, 37-75
REMOVE_MASTER_DATABASES プロシージャ, 37-76

REPCAT_IMPORT_CHECK プロシージャ, 37-77
RESUME_MASTER_ACTIVITY プロシージャ, 37-78
SEND_OLD_VALUES プロシージャ, 37-79
SET_COLUMNS プロシージャ, 37-36, 37-81
SUSPEND_MASTER_ACTIVITY プロシージャ, 37-83
SWITCH_SNAPSHOT_MASTER プロシージャ, 37-83
UNREGISTER_SNAPSHOT_REPGROUP プロシージャ, 37-84
VALIDATE プロシージャ, 37-85
DBMS_REPUTIL パッケージ, 41-1, 41-2
FROM_REMOTE ファンクション, 41-4
GLOBAL_NAME ファンクション, 41-5
MAKE_INTERNAL_PKG プロシージャ, 41-5
REPLICATION_IS_ON ファンクション, 41-4
REPLICATION_OFF プロシージャ, 41-3
REPLICATION_ON プロシージャ, 41-3
SYNC_UP_REP プロシージャ, 41-6
DBMS_RESOURCE_MANAGER_PRIVS パッケージ, 43-1
DBMS_RESOURCE_MANAGER パッケージ, 42-1
DBMS_RLS パッケージ, 44-1
DBMS_ROWID パッケージ, 45-1
DBMS_SESSION パッケージ, 46-1
DBMS_SHARED_POOL パッケージ, 47-1
DBMS_SNAPSHOT パッケージ, 48-2
BEGIN_TABLE_REORGANIZATION プロシージャ, 48-3
DBMS_MVIEW パッケージ, 48-1
END_TABLE_REORGANIZATION プロシージャ, 48-3
I_AM_A_REFRESH ファンクション, 48-4
PURGE_DIRECT_LOAD_LOG プロシージャ, 48-4
PURGE_LOG プロシージャ, 48-5
PURGE_SNAPSHOT_FROM_LOG プロシージャ, 48-6
REFRESH_ALL_MVIEWS プロシージャ, 48-11
REFRESH_DEPENDENT プロシージャ, 48-12
REFRESH プロシージャ, 48-8
REGISTER_SNAPSHOT プロシージャ, 48-14
UNREGISTER_SNAPSHOT プロシージャ, 48-16
シノニムとしての DBMS_MVIEW, 22-1
DBMS_SPACE_ADMIN パッケージ, 50-1
DBMS_SPACE パッケージ, 49-1
DBMS_SQL パッケージ

エラーの位置, 51-32
DBMS_STATS パッケージ, 52-1
DBMS_TRACE パッケージ, 53-1
DBMS_TRANSACTION パッケージ, 54-1
DBMS_TTS パッケージ, 55-1
DBMS_UTILITY パッケージ, 56-1
DDL, 「データ定義言語」を参照
DEBUG_EXPTOC パッケージ, 57-1
DEFDEFAULTDEST ビュー
宛先の削除, 11-4, 11-5
宛先の追加, 11-4
DEFERROR ビュー
トランザクションの削除, 11-5
DES (Data Encryption Standard), 23-1
DESC_TAB データ型, 51-30
DESDecrypt プロシージャ, 23-3
DESEncrypt プロシージャ, 23-2

E

ehlo()
UTL_SMTP のファンクション, 65-6
E メール
UTL_SMTP で送信, 65-1

F

flush()
UTL_TCP のファンクション, 66-12
FORCE パラメータ
ジョブとインスタンスの親和性, 17-12

G

get_host_address()
UTL_INADDR のファンクション, 62-2
get_raw(), get_text(), get_line()
UTL_TCP のファンクション, 66-11

H

helo()
UTL_SMTP のファンクション, 65-5

L

LOB

DBMS_LOB パッケージ, 18-1

M

mail()
UTL_SMTP のファンクション, 65-6

N

noop()
UTL_SMTP のファンクション, 65-12

O

open_connection()
UTL_SMTP のファンクション, 65-3
UTL_TCP のファンクション, 66-3
open_data(), write_data(), write_raw_data(), close_data()
UTL_SMTP のファンクション, 65-9
OR REPLACE 句
パッケージの作成, 1-3
Oracle アドバンスド・キューイング (Oracle AQ)
DBMS_AQADM パッケージ, 5-1
OUTLN_PKG パッケージ, 58-1

P

PL/SQL
データ型, 12-6
数値コード, 12-9

Q

quit()
UTL_SMTP のファンクション, 65-12

R

rcpt()
UTL_SMTP のファンクション, 65-7
read_line()
UTL_TCP のファンクション, 66-9
read_raw()
UTL_TCP のファンクション, 66-6
read_text()
UTL_TCP のファンクション, 66-7

reply、replies

 UTL_SMTP のファンクション、65-3

ROWID のデータ型

 DBMS_ROWID パッケージ、45-1

 拡張形式、45-12

rset()

 UTL_SMTP のファンクション、65-10

S

SDO_ADMIN パッケージ、1-15

SDO_GEOM パッケージ、1-15

SDO_MIGRATE パッケージ、1-15

SDO_TUNE パッケージ、1-15

set_disabled、11-21

SQL*Plus

 順序の作成、1-5

SQL 文

 32KB を超える、51-12

T

TimeScale パッケージ、1-16

TimeSeries パッケージ、1-17

TPC/IP

 UTL_TCP でサポート、66-1

TRACETAB.SQL、53-3

TSTools パッケージ、1-20

U

UTL_COLL パッケージ、59-1

UTL_FILE パッケージ、60-1

UTL_HTTP パッケージ、61-1

UTL_INADDR パッケージ、62-1

UTL_PG パッケージ、1-21

UTL_RAW パッケージ、63-1

UTL_REF パッケージ、64-1

UTL_SMTP パッケージ、65-1

UTL_TCP パッケージ、66-1

V

Vir_Pkg パッケージ、1-22

vrify()

 UTL_SMTP のファンクション、65-11

W

write_line()

 UTL_TCP のファンクション、66-10

write_raw()

 UTL_TCP のファンクション、66-7

write_text()

 UTL_TCP のファンクション、66-8

あ

アップグレード

 アップグレード後のアクション、23-1、6-1

アドバンスト・キューイング

 DBMS_AQADM パッケージ、5-1

 管理インタフェース、4-11

い

移行

 移行後のアクション、6-1、23-1

インターネット・アドレッシング

 UTL_INADDR の使用、62-1

インポート

 オブジェクト・グループ

 オフライン・インスタンスセッション、24-4、
 24-7

 状態チェック、37-77

 スナップショット

 オフライン・インスタンスセッション、25-3、
 25-5

え

エラー

 DBMS_ALERT パッケージが戻すエラー、16-3

 動的 SQL 内での位置、51-32

お

オブジェクト

 削除

 スナップショット・サイト、37-60

 作成、37-38

 スナップショット・サイト、37-42

 マスター・グループ、37-37

 マスター・サイト、37-38

- スナップショット・サイトに追加, 37-42
- 変更, 37-17
- レプリケーション・サポートの生成, 37-64
- オフライン・インスタンス化セッション
 - スナップショット, 25-3, 25-5
 - レプリケート・オブジェクト・グループ, 24-3, 24-4, 24-5, 24-7, 24-8

か

- カーソル
 - DBMS_SQL パッケージ, 51-4
- 各国語サポート
 - NCLOB, 18-3

き

- 機能
 - 新規, xxxv
- キャラクタ・セット
 - ANY_CS, 18-3
- キューイング
 - DBMS_AQADM パッケージ, 5-1
- 休止中
 - マスター・グループ, 37-83
- 競合の解消
 - 追加方法, 37-12
 - 統計, 37-25, 37-74

こ

- コレクション
 - 表項目, 51-15

さ

- サイト優先グループ
 - 削除, 37-57
 - 作成
 - 構文, 37-47
 - メンバーを削除, 37-58
 - メンバーを追加, 37-10
- サイト優先順位
 - 変更, 37-22
- 削除
 - DBA_REPCATLOG 表, 37-69
- 作成

- パッケージ, 1-3

し

- 実行フロー
 - 動的 SQL, 51-4
- 使用禁止
 - 伝播, 11-21
- 状態
 - 伝播, 11-7
- ジョブ
 - キュー
 - ジョブの削除, 11-23
- 新機能, xxxv

す

- ストアド・アウトライン
 - OUTLN_PKG パッケージ, 58-1
- スナップショット
 - オフライン・インスタンス化セッション, 25-3, 25-5
 - サポートの生成, 37-66
 - リフレッシュ, 48-8, 48-11, 48-12
- スナップショット・グループ
 - 作成, 37-41
- スナップショット・サイト
 - 削除, 37-59
 - マスターの変更, 37-83
 - マスターへの変更内容の伝播, 11-19
 - リフレッシュ, 37-71
- スナップショット・ログ
 - マスター表
 - 削除, 48-4, 48-5, 48-6

ち

- 遅延トランザクション
 - DEFDEFAULTDEST ビュー
 - 宛先の削除, 11-4
 - 宛先の追加, 11-4
 - DefDefaultDest 表
 - 宛先の削除, 11-5
 - 開始, 9-7
 - キューからの削除, 11-6
 - 再実行, 11-9
 - スケジュールの実行, 11-19
 - 遅延リモート・プロシージャ・コール (RPC)

- 作成, 9-3
- 即時実行, 11-13
- 引数, 9-5
- 引数型, 10-4
- 引数値, 10-7

違い

- 複数の表, 34-3
- 調整, 34-6

調整

- 表, 34-6

て

データ暗号化規格, 23-1

データ型

- DBMS_DESCRIBE, 12-5
- DESC_TAB, 51-30
- NCLOB, 18-3
- PL/SQL
 - 数値コード, 12-9
- ROWID, 45-1

データ定義言語

- 非同期, 37-63
- 非同期の提供, 37-63

データベース表

- DBMS_TRACE の作成, 53-3

伝播

- 使用禁止, 11-21
- 状態, 11-7
- 変更内容, 37-16
- 変更方法, 37-24

と

統計

- 削除, 37-70
- 収集, 37-74

動的 SQL

- DBMS_SQL ファンクション、使用方法, 51-2
- エラー、位置, 51-32
- 実行フロー, 51-4
- 無名ブロック, 51-2

登録

- ローカル・データベースのプロパゲータ, 11-16

は

配置テンプレート

オブジェクト

- 削除, 40-41
- 作成, 40-22
- すべてを削除, 40-33

オブジェクトの変更, 40-6

オフライン・インスタンスエーション, 39-3, 39-6, 40-49, 40-51

オンライン・インスタンスエーション, 39-9, 40-54

既存のオブジェクトから作成, 40-18

サイト

- 削除, 40-40
- すべてを削除, 40-35

サイト・インスタンスエーションの削除, 39-3

削除, 40-39

- すべてを削除, 40-36

テンプレートのコピー, 40-16

テンプレートの作成, 40-20

テンプレートの比較, 40-15

テンプレートの変更, 40-5

テンプレートのロック, 40-57

パラメータ

- 削除, 40-43
- 作成, 40-26
- すべてを削除, 40-34

パラメータの変更, 40-9

ユーザー認証

- 削除, 40-44
- 作成, 40-28
- すべてを削除, 40-36

ユーザー認証の変更, 40-11

ユーザー・パラメータ値

- 削除, 40-45
- 作成, 40-29
- すべてを削除, 40-37

ユーザー・パラメータ値の変更, 40-12

ランタイム・パラメータ

- 削除, 40-32
- 作成, 40-47
- 取得 ID, 40-46
- 挿入, 40-47

配列

- BIND_ARRAY プロシージャ, 51-6
- DBMS_SQL を使用したバルク DML, 51-15

はじめに

- 表記規則の例の一覧, xxxvii
- パッケージ
 - 作成, 1-3
 - 参照, 1-5
 - 参照先, 1-6
- パッケージの概要, 1-2
- パッケージ変数
 - i_am_a_refresh, 48-4

ひ

- 比較
 - 表, 34-3
- 表
 - 調整, 34-6
 - 配列としての表項目, 51-15
 - 比較, 34-3

ふ

- ファイル名
 - DBMS_BACKUP_RESTORE を使用した正規化, 6-2
- ファイングレイン・アクセス・コントロール
 - DBMS_RLS パッケージ, 44-1
- プランの安定性, 58-1
- プロパゲータ
 - 登録, 11-16

へ

- 変更
 - 伝播方法, 37-16, 37-24

ま

- マスター・グループ
 - 休止中, 37-83
 - 削除, 37-51
 - 作成, 37-37
 - レプリケーション・アクティビティの再開, 37-78
- マスター・サイト
 - 削除, 37-76
 - 作成, 37-7
 - 変更内容の伝播, 11-19
- マスター定義サイト
 - 再配置, 37-75
- マテリアライズド・ビュー

- DBMS_MVIEW を使用したリフレッシュ, 22-1

む

- 無名 PL/SQL ブロック
 - 動的 SQL, 51-2

ゆ

- 優先グループ
 - サイト優先グループ
 - メンバーを追加, 37-10
 - 削除, 37-54
 - 作成, 37-45
 - メンバーの変更
 - 値, 37-20
 - 優先順位, 37-19
 - メンバーを削除, 37-53, 37-55
 - メンバーを追加, 37-9

り

- リフレッシュ
 - スナップショット, 48-8, 48-11, 48-12
 - スナップショット・サイト, 37-71
- リフレッシュ・グループ
 - 削除, 35-6
 - 作成, 35-7
 - メンバーを削除, 35-10
 - メンバーを追加, 35-3
 - リフレッシュ
 - 手動, 35-9
 - リフレッシュ間隔
 - 変更, 35-4

れ

- 列
 - 列グループ
 - 削除, 37-49
 - 作成, 37-44, 37-68
 - メンバーを削除, 37-50
 - メンバーを追加, 37-6
- レプリケーション
 - 使用可能, 41-3
 - 使用禁止, 41-3
- レプリケーション・アクティビティの再開, 37-78

レプリケート・オブジェクト

 マスター・サイトから削除, 37-52

レプリケート・オブジェクト・グループ

 オフライン・インスタンスエーション, 24-3, 24-4,
 24-5, 24-7, 24-8