

Oracle8i

パッケージ・プロシージャ リファレンス Vol. 1

リリース 8.1

部品番号 A62776-1

第 1 版 1999 年 5 月 (第 1 刷)

原本名: Oracle8i Supplied Packages Reference, Volume 1, Release 8.1.5

原本部品番号: A67840-01

原著者: Michele Cyran

原本協力者: 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. 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 © 1999, Oracle Corporation. All rights reserved.

Printed in Japan.

制限付権利の説明

プログラムの使用、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。

当ソフトウェア (プログラム) のリバース・エンジニアリング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 Legend が適用されます。

Restricted Rights Legend

Programs delivered subject to the DOD FAR Supplement are "commercial computer software" and use, duplication and disclosure of the Programs 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 52.227-14, Rights in Data -- General, including Alternate III (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
カレンダー	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	5-15
REVOKE_SYSTEM_PRIVILEGE プロシージャ	5-16
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	5-18
ALTER_SUBSCRIBER プロシージャ	5-19
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	5-22
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	5-25
DISABLE_PROPAGATION_SCHEDULE プロシージャ	5-26
MIGRATE_QUEUE_TABLE プロシージャ	5-27

6 DBMS_DDL

要件	6-2
サブプログラムの要約	6-2
ALTER_COMPILE プロシージャ	6-2
ANALYZE_OBJECT プロシージャ	6-3

7 DBMS_DEBUG

DBMS_DEBUG の使用方法	7-2
使用上の注意	7-5
型	7-6
定数	7-8
エラー・コード	7-10

例外	7-11
変数	7-11
サブプログラムの要約	7-12
共通セクション	7-13
PROBE_VERSION プロシージャ	7-13
SELF_CHECK プロシージャ	7-14
SET_TIMEOUT ファンクション	7-15
TARGET_SESSION セクション	7-15
INITIALIZE ファンクション	7-16
DEBUG_ON プロシージャ	7-16
DEBUG_OFF プロシージャ	7-17
デバッグ・セッション・セクション	7-17
ATTACH_SESSION プロシージャ	7-18
SYNCHRONIZE ファンクション	7-19
SHOW_SOURCE プロシージャ	7-19
PRINT_BACKTRACE プロシージャ	7-21
CONTINUE ファンクション	7-22
SET_BREAKPOINT ファンクション	7-23
DELETE_BREAKPOINT ファンクション	7-25
DISABLE_BREAKPOINT ファンクション	7-25
ENABLE_BREAKPOINT ファンクション	7-26
SHOW_BREAKPOINTS プロシージャ	7-27
GET_VALUE ファンクション	7-28
SET_VALUE ファンクション	7-30
DETACH_SESSION プロシージャ	7-32
GET_RUNTIME_INFO ファンクション	7-32
GET_INDEXES ファンクション	7-33
EXECUTE プロシージャ	7-34

8 DBMS_DEFER

サブプログラムの要約	8-2
CALL プロシージャ	8-2
COMMIT_WORK プロシージャ	8-4
datatype_ARG プロシージャ	8-4
TRANSACTION プロシージャ	8-5

9 DBMS_DEFER_QUERY

サブプログラムの要約.....	9-2
GET_ARG_FORM ファンクション	9-2
GET_ARG_TYPE ファンクション	9-3
GET_CALL_ARGS プロシージャ	9-5
GET_datatype_ARG ファンクション	9-6

10 DBMS_DEFER_SYS

サブプログラムの要約.....	10-2
ADD_DEFAULT_DEST プロシージャ	10-3
DELETE_DEFAULT_DEST プロシージャ	10-4
DELETE_DEF_DESTINATION プロシージャ	10-4
DELETE_ERROR プロシージャ	10-5
DELETE_TRAN プロシージャ	10-5
DISABLED ファンクション	10-6
EXCLUDE_PUSH ファンクション	10-7
EXECUTE_ERROR プロシージャ	10-8
EXECUTE_ERROR_AS_USER プロシージャ	10-9
PURGE ファンクション	10-9
PUSH ファンクション	10-11
REGISTER_PROPAGATOR プロシージャ	10-13
SCHEDULE_PURGE プロシージャ	10-14
SCHEDULE_PUSH プロシージャ	10-15
SET_DISABLED プロシージャ	10-17
UNREGISTER_PROPAGATOR プロシージャ	10-18
UNSCHEDULE_PURGE プロシージャ	10-19
UNSCHEDULE_PUSH プロシージャ	10-19

11 DBMS_DESCRIBE

セキュリティ	11-2
型	11-2
エラー	11-2
サブプログラムの要約.....	11-2
DESCRIBE_PROCEDURE プロシージャ	11-3

例.....	11-5
--------	------

12 DBMS_DISTRICTED_TRUST_ADMIN

要件.....	12-2
サブプログラムの要約	12-2
ALLOW_ALL プロシージャ.....	12-2
ALLOW_SERVER プロシージャ.....	12-3
DENY_ALL プロシージャ.....	12-4
DENY_SERVER プロシージャ.....	12-4
例.....	12-5

13 DBMS_HS

例外.....	13-2
サブプログラムの要約	13-6
ALTER_BASE_CAPS プロシージャ.....	13-8
ALTER_BASE_DD プロシージャ.....	13-9
ALTER_CLASS_CAPS プロシージャ.....	13-9
ALTER_CLASS_DD プロシージャ.....	13-9
ALTER_CLASS_INIT プロシージャ.....	13-9
ALTER_FDS_CLASS プロシージャ.....	13-10
ALTER_FDS_INST プロシージャ.....	13-10
ALTER_INST_CAPS プロシージャ.....	13-10
ALTER_INST_DD プロシージャ.....	13-11
ALTER_INST_INIT プロシージャ.....	13-11
COPY_CLASS プロシージャ.....	13-11
COPY_INST プロシージャ.....	13-12
CREATE_BASE_CAPS プロシージャ.....	13-12
CREATE_BASE_DD プロシージャ.....	13-12
CREATE_CLASS_CAPS プロシージャ.....	13-12
CREATE_CLASS_DD プロシージャ.....	13-13
CREATE_CLASS_INIT プロシージャ.....	13-13
CREATE_FDS_CLASS プロシージャ.....	13-13
CREATE_FDS_INST プロシージャ.....	13-14
CREATE_INST_CAPS プロシージャ.....	13-14

CREATE_INST_DD プロシージャ	13-14
CREATE_INST_INIT プロシージャ	13-15
DROP_BASE_CAPS プロシージャ	13-15
DROP_BASE_DD プロシージャ	13-15
DROP_CLASS_CAPS プロシージャ	13-15
DROP_CLASS_DD プロシージャ	13-16
DROP_CLASS_INIT プロシージャ	13-16
DROP_FDS_CLASS プロシージャ	13-16
DROP_FDS_INST プロシージャ	13-16
DROP_INST_CAPS プロシージャ	13-17
DROP_INST_DD プロシージャ	13-17
DROP_INST_INIT プロシージャ	13-17
REPLACE_BASE_CAPS プロシージャ	13-17
REPLACE_BASE_DD プロシージャ	13-18
REPLACE_CLASS_CAPS プロシージャ	13-18
REPLACE_CLASS_DD プロシージャ	13-18
REPLACE_CLASS_INIT プロシージャ	13-19
REPLACE_FDS_CLASS プロシージャ	13-19
REPLACE_FDS_INST プロシージャ	13-19
REPLACE_INST_CAPS プロシージャ	13-20
REPLACE_INST_DD プロシージャ	13-20
REPLACE_INST_INIT プロシージャ	13-21

14 DBMS_HS_PASSTHROUGH

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

FETCH_ROW ファンクション	14-13
GET_VALUE プロシージャ	14-14
GET_VALUE_RAW プロシージャ	14-15
OPEN_CURSOR ファンクション	14-16
PARSE プロシージャ	14-17

15 DBMS_IOT

サブプログラムの要約	15-2
BUILD_CHAIN_ROWS_TABLE プロシージャ	15-2
BUILD_EXCEPTIONS_TABLE プロシージャ	15-3

16 DBMS_JOB

要件	16-2
サブプログラムの要約	16-2
SUBMIT プロシージャ	16-3
REMOVE プロシージャ	16-4
CHANGE プロシージャ	16-5
WHAT プロシージャ	16-6
NEXT_DATE プロシージャ	16-6
INSTANCE プロシージャ	16-7
INTERVAL プロシージャ	16-7
BROKEN プロシージャ	16-8
RUN プロシージャ	16-9
USER_EXPORT プロシージャ	16-10
USER_EXPORT プロシージャ	16-10

17 DBMS_LOB

要件	17-2
LOB ロケータ	17-2
データ型	17-3
定数	17-3
例外	17-4
セキュリティ	17-4
規則および制限事項	17-5

BFILE 特有の規則および制限事項.....	17-6
テンポラリ LOB.....	17-9
テンポラリ LOB の使用上の注意.....	17-10
テンポラリ LOB の例外.....	17-12
サブプログラムの要約.....	17-12
APPEND プロシージャ.....	17-14
CLOSE プロシージャ.....	17-15
COMPARE ファンクション.....	17-16
COPY プロシージャ.....	17-19
CREATETEMPORARY プロシージャ.....	17-21
ERASE プロシージャ.....	17-22
FILECLOSE プロシージャ.....	17-24
FILECLOSEALL プロシージャ.....	17-25
FILEEXISTS ファンクション.....	17-26
FILEGETNAME プロシージャ.....	17-28
FILEISOPEN ファンクション.....	17-29
FILEOPEN プロシージャ.....	17-30
FREETEMPORARY プロシージャ.....	17-32
GETCHUNKSIZE ファンクション.....	17-33
GETLENGTH ファンクション.....	17-34
INSTR ファンクション.....	17-36
ISOPEN ファンクション.....	17-39
ISTEMPORARY ファンクション.....	17-40
LOADFROMFILE プロシージャ.....	17-40
OPEN プロシージャ.....	17-43
READ プロシージャ.....	17-44
SUBSTR ファンクション.....	17-47
TRIM プロシージャ.....	17-50
WRITE プロシージャ.....	17-52
WRITEAPPEND プロシージャ.....	17-54

18 DBMS_LOCK

要件.....	18-2
セキュリティ.....	18-2
ロックの表示と監視.....	18-2

定数.....	18-2
サブプログラムの要約	18-4
ALLOCATE_UNIQUE プロシージャ	18-4
REQUEST ファンクション	18-6
CONVERT ファンクション	18-7
RELEASE ファンクション	18-9
SLEEP プロシージャ	18-10
例.....	18-10

19 DBMS_LOGMNR

LogMiner の使用方法	19-2
定数.....	19-2
ブレース・ホルダ列の使用法.....	19-3
サブプログラムの要約	19-5
ADD_LOGFILE プロシージャ	19-5
START_LOGMNR プロシージャ.....	19-6
END_LOGMNR プロシージャ	19-8
例.....	19-8

20 DBMS_LOGMNR_D

ディクショナリ・ファイルの作成.....	20-2
サブプログラムの要約	20-2
BUILD プロシージャ	20-2
例.....	20-3

21 DBMS_OFFLINE_OG

サブプログラムの要約	21-2
BEGIN_INSTANTIATION プロシージャ	21-2
BEGIN_LOAD プロシージャ	21-3
END_INSTANTIATION プロシージャ	21-4
END_LOAD プロシージャ	21-5
RESUME_SUBSET_OF_MASTERS プロシージャ	21-6

22 DBMS_OFFLINE_SNAPSHOT

サブプログラムの要約.....	22-2
BEGIN_LOAD プロシージャ	22-2
END_LOAD プロシージャ	22-3

23 DBMS_OLAP

要件	23-2
エラー・メッセージ	23-2
サブプログラムの要約.....	23-3
ESTIMATE_SUMMARY_SIZE プロシージャ	23-3
EVALUATE_UTILIZATION プロシージャ	23-4
EVALUATE_UTILIZATION_W プロシージャ	23-4
RECOMMEND_MV プロシージャ	23-5
RECOMMEND_MV_W プロシージャ	23-7
VALIDATE_DIMENSION プロシージャ	23-8
DBMS_OLAP インタフェース表	23-9

24 DBMS_ORACLE_TRACE_AGENT

セキュリティ	24-2
サブプログラムの要約.....	24-2
SET_ORACLE_TRACE_IN_SESSION プロシージャ	24-2
例	24-3

25 DBMS_ORACLE_TRACE_USER

サブプログラムの要約.....	25-2
SET_ORACLE_TRACE プロシージャ	25-2
例	25-2

26 DBMS_OUTPUT

セキュリティ	26-2
エラー	26-2
型	26-2
DBMS_OUTPUT の使用方法	26-2

サブプログラムの要約	26-3
ENABLE プロシージャ.....	26-3
DISABLE プロシージャ.....	26-4
PUT および PUT_LINE プロシージャ.....	26-4
NEW_LINE プロシージャ.....	26-6
GET_LINE および GET_LINES プロシージャ.....	26-6
例.....	26-8

27 DBMS_PCLXUTIL

DBMS_PCLXUTIL の使用方法.....	27-2
制限事項.....	27-3
サブプログラムの要約	27-3
BUILD_PART_INDEX プロシージャ.....	27-3
例.....	27-4

28 DBMS_PIPE

パブリック・パイプ.....	28-2
プライベート・パイプ.....	28-2
パイプの使用法.....	28-3
セキュリティ.....	28-3
定数.....	28-4
エラー.....	28-4
サブプログラムの要約	28-4
CREATE_PIPE ファンクション.....	28-5
PACK_MESSAGE プロシージャ.....	28-7
SEND_MESSAGE ファンクション.....	28-8
RECEIVE_MESSAGE ファンクション.....	28-10
NEXT_ITEM_TYPE ファンクション.....	28-12
UNPACK_MESSAGE プロシージャ.....	28-13
REMOVE_PIPE ファンクション.....	28-14
PURGE プロシージャ.....	28-15
RESET_BUFFER プロシージャ.....	28-16
UNIQUE_SESSION_NAME ファンクション.....	28-16
例.....	28-17

29 DBMS_PROFILER

DBMS_PROFILER の使用方法	29-2
収集されたデータ	29-3
要件	29-3
エラー・コード	29-3
サブプログラムの要約	29-4
START_PROFILER ファンクション	29-4
STOP_PROFILER ファンクション	29-5
FLUSH_DATA ファンクション	29-5
GET_VERSION プロシージャ	29-6
INTERNAL_VERSION_CHECK ファンクション	29-6

30 DBMS_RANDOM

要件	30-2
サブプログラムの要約	30-2
INITIALIZE プロシージャ	30-2
SEED プロシージャ	30-3
RANDOM ファンクション	30-3
TERMINATE プロシージャ	30-3

31 DBMS_RECTIFIER_DIFF

サブプログラムの要約	31-2
DIFFERENCES プロシージャ	31-2
RECTIFY プロシージャ	31-5

32 DBMS_REFRESH

サブプログラムの要約	32-2
ADD プロシージャ	32-2
CHANGE プロシージャ	32-3
DESTROY プロシージャ	32-5
MAKE プロシージャ	32-6
REFRESH プロシージャ	32-8

SUBTRACT プロシージャ	32-9
-----------------------	------

索引

はじめに	xxxi
------------	------

33 DBMS_REPAIR

セキュリティ	33-2
列挙型	33-2
例外	33-2
サブプログラムの要約	33-4
ADMIN_TABLES プロシージャ	33-4
CHECK_OBJECT プロシージャ	33-5
DUMP_ORPHAN_KEYS プロシージャ	33-7
FIX_CORRUPT_BLOCKS プロシージャ	33-9
REBUILD_FREELISTS プロシージャ	33-10
SKIP_CORRUPT_BLOCKS プロシージャ	33-11

34 DBMS_REPCAT

サブプログラムの要約	34-2
ADD_GROUPED_COLUMN プロシージャ	34-5
ADD_MASTER_DATABASE プロシージャ	34-7
ADD_PRIORITY_datatype プロシージャ	34-8
ADD_SITE_PRIORITY_SITE プロシージャ	34-9
ADD_conflicttype_RESOLUTION プロシージャ	34-10
ALTER_MASTER_PROPAGATION プロシージャ	34-14
ALTER_MASTER_REPOBJECT プロシージャ	34-15
ALTER_PRIORITY プロシージャ	34-17
ALTER_PRIORITY_datatype プロシージャ	34-18
ALTER_SITE_PRIORITY プロシージャ	34-19
ALTER_SITE_PRIORITY_SITE プロシージャ	34-20
ALTER_SNAPSHOT_PROPAGATION プロシージャ	34-21
CANCEL_STATISTICS プロシージャ	34-22
COMMENT_ON_COLUMN_GROUP プロシージャ	34-23

COMMENT_ON_PRIORITY_GROUP/PRIORITY プロシージャ	34-24
COMMENT_ON_REPGROUP プロシージャ.....	34-25
COMMENT_ON_REPSITES プロシージャ.....	34-26
COMMENT_ON_REPOBJECT プロシージャ	34-27
COMMENT_ON_conflicttype_RESOLUTION プロシージャ.....	34-28
COMPARE_OLD_VALUES プロシージャ.....	34-29
CREATE_MASTER_REPGROUP プロシージャ	34-31
CREATE_MASTER_REPOBJECT プロシージャ	34-32
CREATE_SNAPSHOT_REPGROUP プロシージャ	34-35
CREATE_SNAPSHOT_REPOBJECT プロシージャ.....	34-36
DEFINE_COLUMN_GROUP プロシージャ.....	34-38
DEFINE_PRIORITY_GROUP プロシージャ.....	34-39
DEFINE_SITE_PRIORITY プロシージャ.....	34-41
DO_DEFERRED_REPCAT_ADMIN プロシージャ.....	34-41
DROP_COLUMN_GROUP プロシージャ	34-42
DROP_GROUPED_COLUMN プロシージャ.....	34-43
DROP_MASTER_REPGROUP プロシージャ.....	34-44
DROP_MASTER_REPOBJECT プロシージャ.....	34-46
DROP_PRIORITY プロシージャ.....	34-47
DROP_PRIORITY_GROUP プロシージャ	34-47
DROP_PRIORITY_datatype プロシージャ	34-48
DROP_SITE_PRIORITY プロシージャ	34-50
DROP_SITE_PRIORITY_SITE プロシージャ	34-50
DROP_SNAPSHOT_REPGROUP プロシージャ.....	34-51
DROP_SNAPSHOT_REPOBJECT プロシージャ	34-52
DROP_conflicttype_RESOLUTION プロシージャ	34-53
EXECUTE_DDL プロシージャ.....	34-55
GENERATE_REPLICATION_SUPPORT プロシージャ	34-56
GENERATE_SNAPSHOT_SUPPORT プロシージャ.....	34-57
MAKE_COLUMN_GROUP プロシージャ	34-59
PURGE_MASTER_LOG プロシージャ.....	34-60
PURGE_STATISTICS プロシージャ.....	34-61
REFRESH_SNAPSHOT_REPGROUP プロシージャ	34-62
REGISTER_SNAPSHOT_REPGROUP プロシージャ.....	34-63
REGISTER_STATISTICS プロシージャ	34-64

RELOCATE_MASTERDEF プロシージャ	34-65
REMOVE_MASTER_DATABASES プロシージャ	34-66
REPCAT_IMPORT_CHECK プロシージャ	34-67
RESUME_MASTER_ACTIVITY プロシージャ	34-68
SEND_OLD_VALUES プロシージャ	34-69
SET_COLUMNS プロシージャ	34-71
SUSPEND_MASTER_ACTIVITY プロシージャ	34-72
SWITCH_SNAPSHOT_MASTER プロシージャ	34-73
UNREGISTER_SNAPSHOT_REPGROUP プロシージャ	34-74
VALIDATE ファンクション	34-75
WAIT_MASTER_LOG プロシージャ	34-77

35 DBMS_REPCAT_ADMIN

サブプログラムの要約	35-2
GRANT_ADMIN_ANY_SCHEMA プロシージャ	35-2
GRANT_ADMIN_SCHEMA プロシージャ	35-3
REGISTER_USER_REPGROUP プロシージャ	35-3
REVOKE_ADMIN_ANY_SCHEMA プロシージャ	35-5
REVOKE_ADMIN_SCHEMA プロシージャ	35-6
UNREGISTER_USER_REPGROUP プロシージャ	35-6

36 DBMS_REPCAT_INSTANTIATE

サブプログラムの要約	36-2
DROP_SITE_INSTANTIATION プロシージャ	36-2
INSTANTIATE_OFFLINE ファンクション	36-3
INSTANTIATE_ONLINE ファンクション	36-5

37 DBMS_REPCAT_RGT

サブプログラムの要約	37-2
ALTER_REFRESH_TEMPLATE プロシージャ	37-4
ALTER_TEMPLATE_OBJECT プロシージャ	37-6
ALTER_TEMPLATE_PARM プロシージャ	37-9
ALTER_USER_AUTHORIZATION プロシージャ	37-10
ALTER_USER_PARM_VALUE プロシージャ	37-11

COMPARE_TEMPLATES ファンクション	37-14
COPY_TEMPLATE ファンクション	37-15
CREATE_OBJECT_FROM_EXISTING ファンクション	37-17
CREATE_REFRESH_TEMPLATE ファンクション	37-18
CREATE_TEMPLATE_OBJECT ファンクション	37-20
CREATE_TEMPLATE_PARM ファンクション	37-22
CREATE_USER_AUTHORIZATION ファンクション	37-24
CREATE_USER_PARM_VALUE ファンクション	37-26
DELETE_RUNTIME_PARMS プロシージャ	37-28
DROP_ALL_OBJECTS プロシージャ	37-28
DROP_ALL_TEMPLATE_PARMS プロシージャ	37-29
DROP_ALL_TEMPLATE_SITES プロシージャ	37-30
DROP_ALL_TEMPLATES プロシージャ	37-31
DROP_ALL_USER_AUTHORIZATIONS プロシージャ	37-31
DROP_ALL_USER_PARM_VALUES プロシージャ	37-32
DROP_REFRESH_TEMPLATE プロシージャ	37-33
DROP_SITE_INSTANTIATION プロシージャ	37-34
DROP_TEMPLATE_OBJECT プロシージャ	37-35
DROP_TEMPLATE_PARM プロシージャ	37-36
DROP_USER_AUTHORIZATION プロシージャ	37-36
DROP_USER_PARM_VALUE プロシージャ	37-37
GET_RUNTIME_PARM_ID ファンクション	37-38
INSERT_RUNTIME_PARMS プロシージャ	37-39
INstantiate_OFFLINE ファンクション	37-40
INstantiate_ONLINE ファンクション	37-42
LOCK_TEMPLATE_EXCLUSIVE プロシージャ	37-44
LOCK_TEMPLATE_SHARED プロシージャ	37-45

38 DBMS_REPUTIL

サブプログラムの要約	38-2
REPLICATION_OFF プロシージャ	38-2
REPLICATION_ON プロシージャ	38-2
REPLICATION_IS_ON ファンクション	38-3
FROM_REMOTE ファンクション	38-3
GLOBAL_NAME ファンクション	38-3

39 DBMS_RESOURCE_MANAGER

要件.....	39-2
サブプログラムの要約	39-2
CREATE_PLAN プロシージャ	39-3
UPDATE_PLAN プロシージャ	39-4
DELETE_PLAN プロシージャ	39-4
DELETE_PLAN_CASCADE プロシージャ	39-5
CREATE_CONSUMER_GROUP プロシージャ	39-6
UPDATE_CONSUMER_GROUP プロシージャ	39-6
DELETE_CONSUMER_GROUP プロシージャ	39-7
CREATE_PLAN_DIRECTIVE プロシージャ	39-8
UPDATE_PLAN_DIRECTIVE プロシージャ	39-9
DELETE_PLAN_DIRECTIVE プロシージャ	39-10
CREATE_PENDING_AREA プロシージャ	39-10
VALIDATE_PENDING_AREA プロシージャ	39-12
CLEAR_PENDING_AREA プロシージャ	39-12
SUBMIT_PENDING_AREA プロシージャ	39-12
SET_INITIAL_CONSUMER_GROUP プロシージャ	39-16
SWITCH_CONSUMER_GROUP_FOR_SESS プロシージャ	39-17
SWITCH_CONSUMER_GROUP_FOR_USER プロシージャ	39-17

40 DBMS_RESOURCE_MANAGER_PRIVS

サブプログラムの要約	40-2
GRANT_SYSTEM_PRIVILEGE プロシージャ	40-2
REVOKE_SYSTEM_PRIVILEGE プロシージャ	40-3
GRANT_SWITCH_CONSUMER_GROUP プロシージャ	40-3
REVOKE_SWITCH_CONSUMER_GROUP プロシージャ	40-5

41 DBMS_RLS

動的な述語.....	41-2
セキュリティ	41-3
使用上の注意.....	41-3
サブプログラムの要約	41-3
ADD_POLICY プロシージャ	41-3

DROP_POLICY プロシージャ	41-5
REFRESH_POLICY プロシージャ	41-6
ENABLE_POLICY プロシージャ	41-7
例	41-8

42 DBMS_ROWID

使用上の注意	42-2
要件	42-2
ROWID のタイプ	42-3
例外	42-4
サブプログラムの要約	42-4
ROWID_CREATE ファンクション	42-5
ROWID_INFO プロシージャ	42-6
ROWID_TYPE ファンクション	42-7
ROWID_OBJECT ファンクション	42-8
ROWID_RELATIVE_FNO ファンクション	42-9
ROWID_BLOCK_NUMBER ファンクション	42-10
ROWID_ROW_NUMBER ファンクション	42-10
ROWID_TO_ABSOLUTE_FNO ファンクション	42-11
ROWID_TO_EXTENDED ファンクション	42-12
ROWID_TO_RESTRICTED ファンクション	42-14
ROWID_VERIFY ファンクション	42-14

43 DBMS_SESSION

要件	43-2
サブプログラムの要約	43-2
SET_ROLE プロシージャ	43-3
SET_SQL_TRACE プロシージャ	43-3
SET_NLS プロシージャ	43-4
CLOSE_DATABASE_LINK プロシージャ	43-4
RESET_PACKAGE プロシージャ	43-5
UNIQUE_SESSION_ID ファンクション	43-6
IS_ROLE_ENABLED ファンクション	43-7
IS_SESSION_ALIVE ファンクション	43-7

SET_CLOSE_CACHED_OPEN_CURSORS プロシージャ	43-8
FREE_UNUSED_USER_MEMORY プロシージャ	43-8
SET_CONTEXT プロシージャ	43-11
LIST_CONTEXT プロシージャ	43-11
SWITCH_CURRENT_CONSUMER_GROUP プロシージャ	43-12
例.....	43-14

44 DBMS_SHARED_POOL

インストール時の注意.....	44-2
使用上の注意.....	44-2
サブプログラムの要約	44-2
SIZES プロシージャ	44-2
KEEP プロシージャ	44-3
UNKEEP プロシージャ	44-5
ABORTED_REQUEST_THRESHOLD プロシージャ	44-5

45 DBMS_SNAPSHOT

サブプログラムの要約	45-2
BEGIN_TABLE_REORGANIZATION プロシージャ	45-2
END_TABLE_REORGANIZATION プロシージャ	45-3
I_AM_A_REFRESH ファンクション.....	45-3
PURGE_DIRECT_LOAD_LOG プロシージャ	45-4
PURGE_LOG プロシージャ	45-4
PURGE_SNAPSHOT_FROM_LOG プロシージャ.....	45-5
REFRESH プロシージャ	45-7
REFRESH_ALL_MVIEWS プロシージャ	45-9
REFRESH_DEPENDENT プロシージャ	45-10
REGISTER_SNAPSHOT プロシージャ.....	45-12
UNREGISTER_SNAPSHOT プロシージャ	45-13

46 DBMS_SPACE

セキュリティ	46-2
要件	46-2
サブプログラムの要約	46-2

UNUSED_SPACE プロシージャ	46-2
FREE_BLOCKS プロシージャ	46-3
例.....	46-5

47 DBMS_SPACE_ADMIN

セキュリティ	47-2
定数.....	47-2
サブプログラムの要約	47-3
SEGMENT_VERIFY プロシージャ	47-3
SEGMENT_CORRUPT プロシージャ	47-4
SEGMENT_DROP_CORRUPT プロシージャ.....	47-5
SEGMENT_DUMP プロシージャ	47-6
TABLESPACE_VERIFY プロシージャ	47-7
TABLESPACE_FIX_BITMAPS プロシージャ	47-7
TABLESPACE_REBUILD_BITMAPS プロシージャ.....	47-8
TABLESPACE_REBUILD_QUOTAS プロシージャ	47-9
TABLESPACE_MIGRATE_FROM_LOCAL プロシージャ	47-10
例.....	47-10

48 DBMS_SQL

DBMS_SQL の使用方法.....	48-2
定数.....	48-3
型.....	48-3
例外	48-3
実行フロー	48-4
セキュリティ	48-7
サブプログラムの要約	48-8
OPEN_CURSOR ファンクション	48-9
PARSE プロシージャ	48-10
BIND_VARIABLE プロシージャ	48-12
BIND_ARRAY プロシージャ	48-12
問合せの処理.....	48-16
DEFINE_COLUMN プロシージャ	48-17
DEFINE_ARRAY プロシージャ.....	48-18

DEFINE_COLUMN_LONG プロシージャ	48-20
EXECUTE ファンクション	48-20
EXECUTE_AND_FETCH ファンクション	48-21
FETCH_ROWS ファンクション	48-21
COLUMN_VALUE プロシージャ	48-22
COLUMN_VALUE_LONG プロシージャ	48-24
VARIABLE_VALUE プロシージャ	48-25
更新、挿入、削除の処理	48-27
IS_OPEN ファンクション	48-27
DESCRIBE_COLUMNS プロシージャ	48-28
CLOSE_CURSOR プロシージャ	48-30
エラーの位置	48-31
LAST_ERROR_POSITION ファンクション	48-31
LAST_ROW_COUNT ファンクション	48-31
LAST_ROW_ID ファンクション	48-32
LAST_SQL_FUNCTION_CODE ファンクション	48-32
例	48-33

49 DBMS_STATS

DBMS_STATS の使用方法	49-2
型	49-2
サブプログラムの要約	49-3
統計情報の設定または取得	49-5
PREPARE_COLUMN_VALUES プロシージャ	49-5
SET_COLUMN_STATS プロシージャ	49-7
SET_INDEX_STATS プロシージャ	49-9
SET_TABLE_STATS プロシージャ	49-10
CONVERT_RAW_VALUE プロシージャ	49-11
GET_COLUMN_STATS プロシージャ	49-13
GET_INDEX_STATS プロシージャ	49-14
GET_TABLE_STATS プロシージャ	49-15
DELETE_COLUMN_STATS プロシージャ	49-16
DELETE_INDEX_STATS プロシージャ	49-17
DELETE_TABLE_STATS プロシージャ	49-18
DELETE_SCHEMA_STATS プロシージャ	49-20

DELETE_DATABASE_STATS プロシージャ	49-20
統計情報の転送.....	49-21
CREATE_STAT_TABLE プロシージャ	49-22
DROP_STAT_TABLE プロシージャ	49-22
EXPORT_COLUMN_STATS プロシージャ	49-23
EXPORT_INDEX_STATS プロシージャ	49-24
EXPORT_TABLE_STATS プロシージャ	49-25
EXPORT_SCHEMA_STATS プロシージャ	49-26
EXPORT_DATABASE_STATS プロシージャ	49-26
IMPORT_COLUMN_STATS プロシージャ	49-27
IMPORT_INDEX_STATS プロシージャ	49-28
IMPORT_TABLE_STATS プロシージャ	49-29
IMPORT_SCHEMA_STATS プロシージャ	49-30
IMPORT_DATABASE_STATS プロシージャ	49-30
オブティマイザ統計情報の収集.....	49-31
GATHER_INDEX_STATS プロシージャ.....	49-31
GATHER_TABLE_STATS プロシージャ.....	49-32
GATHER_SCHEMA_STATS プロシージャ.....	49-34
GATHER_DATABASE_STATS プロシージャ.....	49-37
GENERATE_STATS プロシージャ.....	49-39
例.....	49-40

50 DBMS_TRACE

要件.....	50-2
制限事項.....	50-2
定数.....	50-2
DBMS_TRACE の使用方法.....	50-2
サブプログラムの要約	50-4
SET_PLSQL_TRACE プロシージャ.....	50-4
CLEAR_PLSQL_TRACE プロシージャ	50-4
PLSQL_TRACE_VERSION プロシージャ	50-5

51 DBMS_TRANSACTION

要件.....	51-2
---------	------

サブプログラムの要約	51-2
READ_ONLY プロシージャ	51-3
READ_WRITE プロシージャ	51-3
ADVISE_ROLLBACK プロシージャ	51-3
ADVISE_NOTHING プロシージャ	51-4
ADVISE_COMMIT プロシージャ	51-4
USE_ROLLBACK_SEGMENT プロシージャ	51-4
COMMIT_COMMENT プロシージャ	51-5
COMMIT_FORCE プロシージャ	51-5
COMMIT プロシージャ	51-6
SAVEPOINT プロシージャ	51-6
ROLLBACK プロシージャ	51-7
ROLLBACK_SAVEPOINT プロシージャ	51-7
ROLLBACK_FORCE プロシージャ	51-8
BEGIN_DISCRETE_TRANSACTION プロシージャ	51-8
PURGE_MIXED プロシージャ	51-9
PURGE_LOST_DB_ENTRY プロシージャ	51-10
LOCAL_TRANSACTION_ID ファンクション	51-12
STEP_ID ファンクション	51-12

52 DBMS_TTS

例外	52-2
サブプログラムの要約	52-2
TRANSPORT_SET_CHECK プロシージャ	52-2
DOWNGRADE プロシージャ	52-3

53 DBMS_UTILITY

要件	53-2
型	53-2
サブプログラムの要約	53-2
COMPILE_SCHEMA プロシージャ	53-4
ANALYZE_SCHEMA プロシージャ	53-5
ANALYZE_DATABASE プロシージャ	53-6
FORMAT_ERROR_STACK ファンクション	53-7

FORMAT_CALL_STACK ファンクション	53-7
IS_PARALLEL_SERVER ファンクション	53-8
GET_TIME ファンクション	53-8
GET_PARAMETER_VALUE ファンクション	53-9
NAME_RESOLVE プロシージャ	53-10
NAME_TOKENIZE プロシージャ	53-12
COMMA_TO_TABLE プロシージャ	53-12
TABLE_TO_COMMA プロシージャ	53-13
PORT_STRING ファンクション	53-14
DB_VERSION プロシージャ	53-14
MAKE_DATA_BLOCK_ADDRESS ファンクション	53-15
DATA_BLOCK_ADDRESS_FILE ファンクション	53-15
DATA_BLOCK_ADDRESS_BLOCK ファンクション	53-16
GET_HASH_VALUE ファンクション	53-17
ANALYZE_PART_OBJECT プロシージャ	53-18
EXEC_DDL_STATEMENT プロシージャ	53-19
CURRENT_INSTANCE ファンクション	53-19
ACTIVE_INSTANCES プロシージャ	53-19

54 DEBUG_EXTPROC

要件	54-2
インストレーション時の注意	54-2
DEBUG_EXTPROC の使用方法	54-2
サブプログラムの要約	54-3
STARTUP_EXTPROC_AGENT プロシージャ	54-3

55 OUTLN_PKG

要件	55-2
セキュリティ	55-2
サブプログラムの要約	55-2
DROP_UNUSED プロシージャ	55-2
DROP_BY_CAT プロシージャ	55-3
UPDATE_BY_CAT プロシージャ	55-3

56 UTL_COLL

サブプログラムの要約.....	56-2
IS_LOCATOR ファンクション	56-2
例.....	56-3

57 UTL_FILE

セキュリティ	57-2
ファイルの所有権と保護.....	57-3
例 (UNIX 固有).....	57-3
型.....	57-4
例外	57-4
サブプログラムの要約.....	57-5
FOPEN ファンクション	57-6
IS_OPEN ファンクション	57-7
FCLOSE プロシージャ.....	57-8
FCLOSE_ALL プロシージャ.....	57-8
GET_LINE プロシージャ	57-9
PUT プロシージャ.....	57-10
NEW_LINE プロシージャ	57-11
PUT_LINE プロシージャ	57-12
PUTF プロシージャ.....	57-12
FFLUSH プロシージャ	57-14
FOPEN ファンクション	57-14

58 UTL_HTTP

例外	58-2
使用上の注意.....	58-3
サブプログラムの要約.....	58-3
REQUEST ファンクション	58-3
REQUEST_PIECES ファンクション.....	58-5
例.....	58-6

59 UTL_RAW

使用上の注意.....	59-2
-------------	------

サブプログラムの要約	59-2
CONCAT ファンクション	59-3
CAST_TO_RAW ファンクション	59-4
CAST_TO_VARCHAR2 ファンクション	59-5
LENGTH ファンクション	59-6
SUBSTR ファンクション	59-7
TRANSLATE ファンクション	59-8
TRANSLITERATE ファンクション	59-10
OVERLAY ファンクション	59-11
COPIES ファンクション	59-13
XRANGE ファンクション	59-14
REVERSE ファンクション	59-15
COMPARE ファンクション	59-16
CONVERT ファンクション	59-17
BIT_AND ファンクション	59-19
BIT_OR ファンクション	59-20
BIT_XOR ファンクション	59-21
BIT_COMPLEMENT ファンクション	59-22

60 UTL_REF

要件	60-2
データ型	60-2
例外	60-2
セキュリティ	60-3
サブプログラムの要約	60-4
SELECT_OBJECT プロシージャ	60-4
LOCK_OBJECT プロシージャ	60-5
UPDATE_OBJECT プロシージャ	60-6
DELETE_OBJECT プロシージャ	60-7
例	60-8

索引

はじめに

このマニュアルでは、Oracle8i、リリース 8.1.5 プログラムに付属の Oracle 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、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、システム固有の情報は含まれていません。

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](#)
- [対象読者](#)
- [関連マニュアル](#)
- [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（本体は必要ない場合もあります）。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];
```

仕様部は、アプリケーションで参照できるパブリック宣言を含んでいます。本体は、インプリメンテーションの詳細とプライベート宣言を含んでいます。これらは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、容易なアプリケーション設計、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。

対象読者

このマニュアルは、Oracle パッケージを使用する、または Oracle パッケージを使用する予定があるすべてのユーザー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を行う予定があるユーザーにも有効です。

このマニュアルでは、読者がアプリケーション・プログラミングに関する作業知識があり、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を理解していることも前提としています。

関連マニュアル

詳細は、Oracle8i マニュアル・セットにある次のマニュアル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』
- 『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、Oracle Enterprise Manager 行モード (Server Manager) および SQL*Plus のコマンドまたは文は、クーリエ・フォントで表示されます。 例： <pre>INSERT INTO emp (empno, ename) VALUES (1000, 'SMITH'); ALTER TABLESPACE users ADD DATAFILE 'users2.ora' SIZE 50K;</pre>
< >	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 などのさまざま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から選択するには、そのビューの SELECT 権限が必要です。ビューからの選択には個々の 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、『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受け入れます。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、例外が発生します。

```
    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、最初に順序番号 emp_sequence を作成してください。次の SQL*Plus 文で作成できます。

```
SQL> EXECUTE 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、今後のリリース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、データベースの内部エラーと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・イベントの非同期式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_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 6 章
DBMS_DEBUG	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。	第 7 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 8 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 9 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 10 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 11 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 12 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 13 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 14 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 15 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 16 章
DBMS_LOB	Oracle ラージ・オブジェクト (Oracle Large Object: LOB) データ型 - BLOB、CLOB (読取り / 書込み) および BFILE (読取り専用) における操作に対する汎用ルーチンを提供します。	第 17 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 18 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 19 章
DBMS_LOGMNR_D	現行データベースのディクショナリ表を問い合わせ、その内容を格納しているテキスト・ベースのファイルを作成します。	第 20 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_OFFLINE_OG	マスター・グループのオフライン・インスタンスーション用のパブリック API を提供します。	第 21 章
DBMS_OFFLINE_SNAPSHOT	スナップショットのオフライン・インスタンスーション用のパブリック API を提供します。	第 22 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 23 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 24 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 25 章
DBMS_OUTPUT	後で取り出せるようにして情報をバッファに蓄積します。	第 26 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 27 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 28 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 29 章
DBMS_RANDOM	組込み式の乱数ジェネレータを提供します。	第 30 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 31 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 32 章
DBMS_REPAIR	データの破損修復プロシージャを提供します。	第 33 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 34 章
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 35 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_INSTANTIATE	配置テンプレートをインスタス化します。レプリケーション・オプションが必要です。	第 36 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 37 章
DBMS_REPUTIL	表複製用のシャドー表、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 38 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 39 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 40 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 41 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 42 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 43 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 44 章
DBMS_SNAPSHOT (シノニム DBMS_MVIEW)	同じリフレッシュ・グループやバージ・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 45 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 46 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 47 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 48 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)
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 49 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_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 50 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 51 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 52 章
DBMS_UTILITY	さまざまなユーティリティ・ルーチンを提供します。	第 53 章
DEBUG_EXTPROC	実行プロセスに連結できるデバッグを使用して、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 54 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 55 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 56 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・ファイルの読取りと書込みを可能にし、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 57 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 58 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 59 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 60 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。

カレンダー

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。
NumInvalidTimeStamps Between	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 の数を戻します。
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戻しま 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戻しま 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 の数を戻します。
OffsetDate	オフセット入力に対応するタイムスタンプを戻します。
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 スタンプを戻します。
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 (ORDTDateTab) を戻します。
UnionCals	2 つの入力カレンダーを結合したカレンダーを戻します。
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決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・グループの合計値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、さまざま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 ORDTVarchar2Tab) 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 ORDTVarchar2Series) 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）の非同期式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。このため、特定のアラートに対して、同時実行の送信と待機がいくつあってもかまいません。

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ストア・プロシージャをコールする場合の制限については、Oracle Forms のマニュアル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 プロシーダを使用し、すべてのイベントの発生を待機できます。

WAITONE または WAITANY プロシーダに対して、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 28 章の「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、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、その先行したアラートに関するデータが読み込まれていないためです。

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 プロシーダで設定できます。
- WAITANY プロシーダ。WAITANY プロシーダを使用していて、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
REGISTER プロシージャ (2-4 ページ)	アラートからメッセージを受け取ります。
REMOVE プロシージャ (2-5 ページ)	アラートからの通知を無効にします。
REMOVEALL プロシージャ (2-5 ページ)	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。
SET_DEFAULTS プロシージャ (2-6 ページ)	ポーリング間隔を設定します。
SIGNAL プロシージャ (2-6 ページ)	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)。
WAITANY プロシージャ (2-7 ページ)	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。
WAITONE プロシージャ (2-8 ページ)	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、アラートを必要な数だけ登録できます。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： 'ORAS' 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。

このアラートと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直列化されます。

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

現在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 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 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、後で行うさまざまなモジュールのパフォーマンスを追跡する時に使用されます。

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形式名または 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
SET_MODULE プロシージャ (3-2 ページ)	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。
SET_ACTION プロシージャ (3-3 ページ)	現行モジュール内の現行アクション名を設定します。
READ_MODULE プロシージャ (3-4 ページ)	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。
SET_CLIENT_INFO プロシージャ (3-5 ページ)	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。
READ_CLIENT_INFO プロシージャ (3-6 ページ)	現行セッションの client_info フィールドの値を読み込みます。
SET_SESSION_LONGOPS プロシージャ (3-6 ページ)	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 PROCEDURE add_employee(  
    name VARCHAR2(20),  
    salary NUMBER(7,2),  
    manager NUMBER,  
    title VARCHAR2(9),  
    commission NUMBER(7,2),  
    department NUMBER(2)) 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, next.emp_seq, 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 列の使用方法の例です。

```
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。

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

このプロシージャでは、V\$SESSION_LONGOP 表に行を設定します。この表は通常、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。Parallel Query や Server Managed Recovery などの一部の Oracle 機能では、この表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に関する情報を通知するために、アプリケーションがこの機能を使用することもできます。

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。新規行を開始するには、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。

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、それを定義す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ーフェースは、RAW パ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 記憶域文字列を構成することによって、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、『Oracle8iSQL 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ず。</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>number: 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 仕様は上書きされます。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。 number: 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>number: 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、「エージェント」(4-4 ページ) 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。このパラメータはエン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。 デフォルト : 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。 デフォルト : 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
ENQUEUE プロシージャ (4-11 ページ)	指定したキューにメッセージを追加します。
DEQUEUE プロシージャ (4-13 ページ)	指定したキューからメッセージをデキューします。
LISTEN プロシージャ (4-15 ページ)	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。

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	「エンキュー・オプション・タイプ」 (4-5 ページ) を参照してください。
message_properties	「メッセージ・プロパティ・タイプ」 (4-8 ページ) 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。受け入れ可能なパラメータは NULL です。 <type_name> の定義の詳細は、 「タイプ名」 (4-3 ページ) を参照してください。

表 4-8 ENQUEUE プロシージャのパラメータ

パラメータ	説明
msgid	システムによって生成されるメッセージ 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。その他のメッセージの処理順序は、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	「 デキュー・オプション・タイプ 」(4-6 ページ)を参照してください。
message_properties	「 メッセージ・プロパティ・タイプ 」(4-8 ページ)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、「 タイプ名 」(4-3 ページ)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： 現在エンキューされている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のメッセージから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み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・キューのみが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 章の「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
CREATE_QUEUE_TABLE プロシージャ (5-4 ページ)	事前定義型のメッセージのキュー表を作成します。
ALTER_QUEUE_TABLE プロシージャ (5-7 ページ)	既存のキュー表を変更します。
DROP_QUEUE_TABLE プロシージャ (5-8 ページ)	既存のキュー表を削除します。
CREATE_QUEUE プロシージャ (5-9 ページ)	指定したキュー表にキューを作成します。
CREATE_NP_QUEUE プロシージャ (5-11 ページ)	非永続の RAW キューを作成します。
ALTER_QUEUE プロシージャ (5-12 ページ)	キューの既存のプロパティを変更します。
DROP_QUEUE プロシージャ (5-13 ページ)	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
START_QUEUE プロシージャ (5-14 ページ)	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。
STOP_QUEUE プロシージャ (5-15 ページ)	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。
GRANT_SYSTEM_PRIVILEGE プロシージャ (5-15 ページ)	ユーザーとロールに AQ システム権限を付与します。
REVOKE_SYSTEM_PRIVILEGE プロシージャ (5-16 ページ)	ユーザーとロールから AQ システム権限を取り消します。
GRANT_QUEUE_PRIVILEGE プロシージャ (5-17 ページ)	ユーザーとロールにキューの権限を付与します。
REVOKE_QUEUE_PRIVILEGE プロシージャ (5-18 ページ)	ユーザーとロールからキューの権限を取り消します。
ADD_SUBSCRIBER プロシージャ (5-18 ページ)	キューにデフォルトのサブスクライバを追加します。
ALTER_SUBSCRIBER プロシージャ (5-19 ページ)	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。
REMOVE_SUBSCRIBER プロシージャ (5-20 ページ)	キューからデフォルトのサブスクライバを削除します。
SCHEDULE_PROPAGATION プロシージャ (5-21 ページ)	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。
UNSCHEDULE_PROPAGATION プロシージャ (5-22 ページ)	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。
VERIFY_QUEUE_TYPES プロシージャ (5-23 ページ)	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。
ALTER_PROPAGATION_SCHEDULE プロシージャ (5-24 ページ)	伝播スケジュールのパラメータを変更します。
ENABLE_PROPAGATION_SCHEDULE プロシージャ (5-25 ページ)	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。
DISABLE_PROPAGATION_SCHEDULE プロシージャ (5-26 ページ)	伝播スケジュールを使用禁止にします。

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・データの問合せに使用する読み専用ビュー。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、サブスク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、サブスク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、「 タイプ名 」(4-3 ページ)を参照してください。
storage_clause	記憶領域パラメータ。 記憶領域パラメータは、キュー表の作成時に、CREATE TABLE 文に組み込まれます。記憶領域パラメータは、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。 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。 これらのパラメータの使用方法については、『Oracle8iSQL リファレンス』を参照してください。
sort_list	昇順ソート・キーに使用する列。 Sort_list の書式は次のとおりです。 '<sort_column_1>,<sort_column_2>' 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。 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。 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。 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。
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。

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 8.1.5 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

```
DBMS_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てはならず、予約語に関しては、『Oracle8iSQL 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>NUMBER: 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>将来使用される予定で、現在は使用されていません。</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する必要があります。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てはならず、予約語に関しては、『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する必要があります。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	<p>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。</p> <p>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。</p> <p>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。</p>
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）。デフォルトは NULL で、値が変更されないことを意味します。</p> <p>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。</p>
retention_time	<p>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。</p>

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	キューの名前。
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	キューの名前。
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	キューの名前。
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	<p>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。</p> <p>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（オプション・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。</p> <p>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。</p>

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	キューの名前。
subscriber	サブスクリプションを変更するエージェント。「 エージェント 」(4-4 ページ) 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。「 エージェント 」(4-4 ページ) 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。
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。

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

パラメータ	説明
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ファンクション。 この値が NULL の場合、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伝播されるメッセージをキューが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のスキーマ名に設定されます。

表 5-20 UNSCHEDULE_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。

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。

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
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	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。
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。

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
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ファンクション。 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)。 デフォルト値は 60 です。 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。 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伝播されるメッセージをキューが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。

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。

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
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。

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 6-1 DBMS_DDL パッケージのサブプログラム

サブプログラム	説明
ALTER_COMPILE プロシージャ (6-2 ページ)	PL/SQL オブジェクトをコンパイルします。
ANALYZE_OBJECT プロシージャ (6-3 ページ)	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 文と同等です。

```
ALTER PROCEDURE|FUNCTION|PACKAGE [<schema>.] <name> COMPILE [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 6-2 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 6-3 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 文と同等です。

```
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 6-4 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 6-4 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	推定する行の割合。 estimate_rows を指定すると、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 6-5 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ポストされます。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 を使用して初期化される必要があります。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・セッションからポスト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 7-1](#) と [図 7-2](#) 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または変更を実行できません。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 7-1 ターゲット・セッション

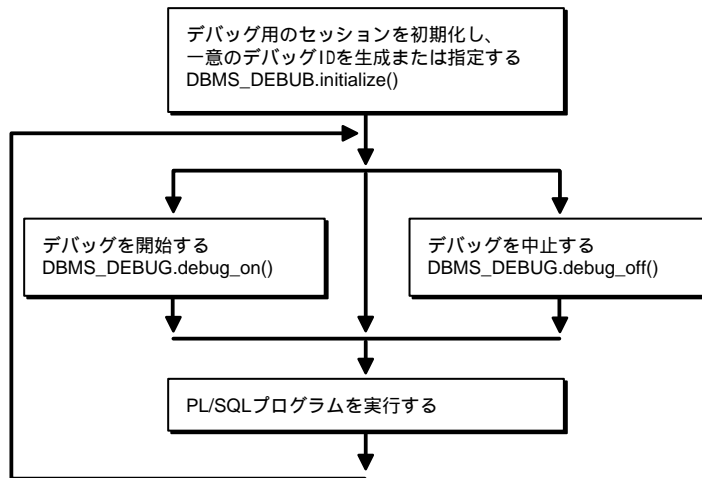


図 7-2 デバッグ・セッション

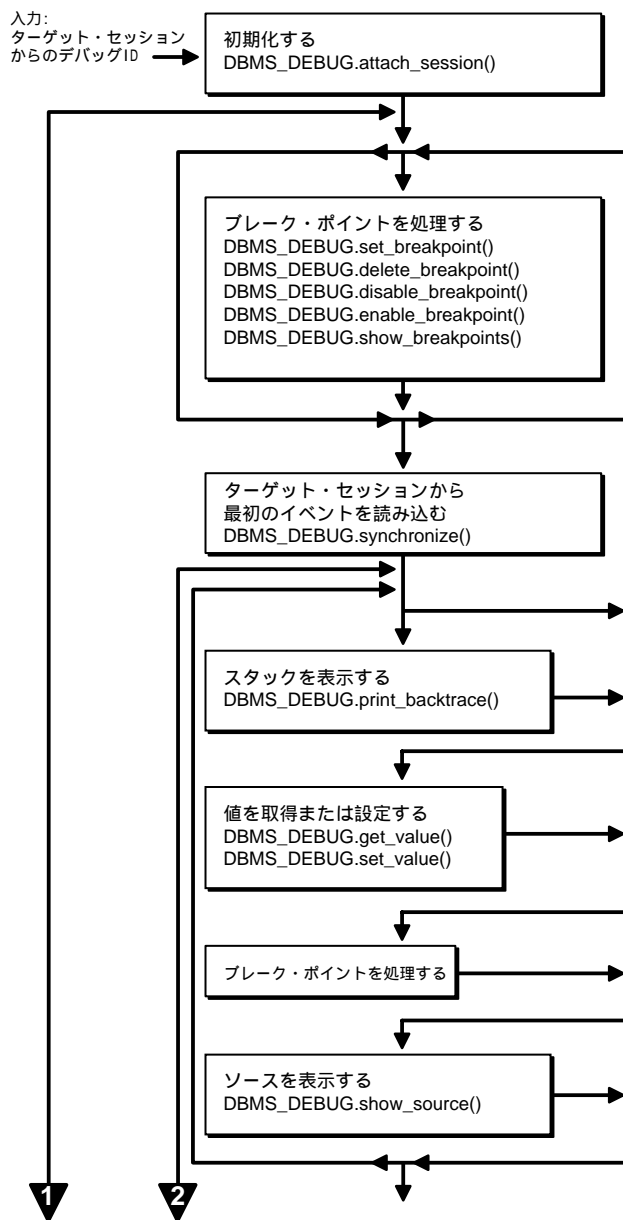
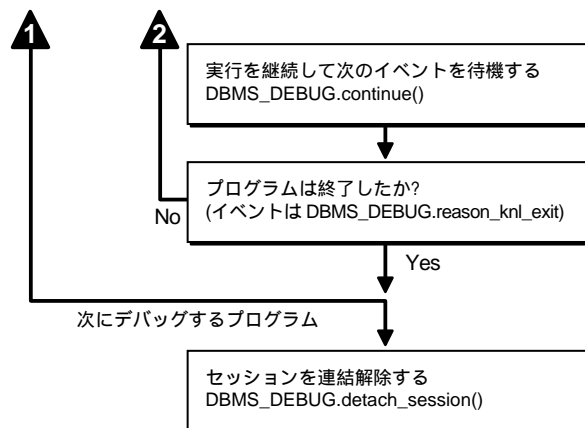


図 7-2 デバッグ・セッション（続き）



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。読み込み専用フィールドは、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、「[Libunit 型](#)」(7-8 ページ) 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）

NAMESPACES 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する必要があります。

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ブ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。

<code>break_next_line</code>	次のソース行でブレイクします（コールをスキップ）。
<code>break_any_call</code>	次のソース行でブレイクします（コールを開始）。
<code>break_any_return</code>	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。
<code>break_return</code>	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 <code>Proc2</code> をコールする <code>Proc1</code> を実行している場合、 <code>break_return</code> は <code>Proc2</code> の終了時に停止します。）
<code>break_exception</code>	例外が発生したときにブレイクします。
<code>break_handler</code>	例外ハンドラが実行されたときにブレイクします。
<code>abort_execution</code>	実行を停止し、 <code>DBMS_DEBUG.CONTINUE</code> がコールされるとすぐに、 <code>'exit'</code> 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>	ブレイク・ポイント番号を取得します。
<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イクします。

<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>	ブレイク・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) でコールされるさまざま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。

<code>success</code>	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。

<code>error_bogus_frame</code>	該当するエントリポイントがスタックにありません。
<code>error_no_debug_info</code>	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。
<code>error_no_such_object</code>	該当する変数またはパラメータがありません。
<code>error_unknown_type</code>	デバッグ情報を読み取ることができません。
<code>error_indexed_table</code>	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。
<code>error_illegal_index</code>	該当する要素がコレクション内に存在しません。
<code>error_nullcollection</code>	表がアトミック NULL です。
<code>error_nullvalue</code>	値が NULL です。

SET_VALUE が戻すステータスは次のとおりです。

<code>error_illegal_value</code>	制約違反。
<code>error_illegal_null</code>	制約違反。
<code>error_value_malformed</code>	指定された値を解読できません。
<code>error_other</code>	その他のエラー。
<code>error_name_incomplete</code>	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。

<code>error_no_such_breakpt</code>	該当するブレーク・ポイントがありません。
<code>error_idle_breakpt</code>	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。
<code>error_bad_handle</code>	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 7-1 DBMS_DEBUG パッケージのサブプログラム

サブプログラム	説明
PROBE_VERSION プロシージャ (7-13 ページ)	サーバー上の DBMS_DEBUG のバージョン番号を戻します。
SELF_CHECK プロシージャ (7-14 ページ)	内部一貫性チェックを実行します。
SET_TIMEOUT ファンクション (7-15 ページ)	タイムアウト値を設定します。
INITIALIZE ファンクション (7-16 ページ)	ターゲット・セッションのデバッグ ID を設定します。
DEBUG_ON プロシージャ (7-16 ページ)	デバッグ・モードをオンにします。
DEBUG_OFF プロシージャ (7-17 ページ)	デバッグ・モードをオフにします。
ATTACH_SESSION プロシージャ (7-18 ページ)	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。
SYNCHRONIZE ファンクション (7-19 ページ)	プログラムの実行開始を待機します。
SHOW_SOURCE プロシージャ (7-19 ページ)	プログラム・ソースを取り出します。
PRINT_BACKTRACE プロシージャ (7-21 ページ)	スタックのバックトレースを印刷します。
CONTINUE ファンクション (7-22 ページ)	ターゲット・プログラムの実行を継続します。
SET_BREAKPOINT ファンクション (7-23 ページ)	プログラム・ユニットにブレーク・ポイントを設定します。
DELETE_BREAKPOINT ファンクション (7-25 ページ)	ブレーク・ポイントを削除します。
DISABLE_BREAKPOINT ファンクション (7-25 ページ)	ブレーク・ポイントを使用禁止にします。
ENABLE_BREAKPOINT ファンクション (7-26 ページ)	既存のブレーク・ポイントをアクティブ化します。
SHOW_BREAKPOINTS プロシージャ (7-27 ページ)	現行ブレーク・ポイントのリストを戻します。

表 7-1 DBMS_DEBUG パッケージのサブプログラム

サブプログラム	説明
GET_VALUE ファンクション (7-28 ページ)	現在実行中のプログラムから値を取得します。
SET_VALUE ファンクション (7-30 ページ)	現在実行中のプログラムに値を設定します。
DETACH_SESSION プロシージャ (7-32 ページ)	ターゲット・プログラムのデバッグを停止します。
GET_RUNTIME_INFO ファンクション (7-32 ページ)	現行プログラムに関する情報を戻します。
GET_INDEXES ファンクション (7-33 ページ)	索引表に対する一連の索引を戻します。
EXECUTE プロシージャ (7-34 ページ)	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 7-2 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。

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 7-3 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 7-4 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 7-4 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 7-5 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 7-6 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 7-7 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 ではない場合、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 7-8 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 7-9 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。 「 情報フラグ 」(7-9 ページ) を参照してください。

戻り値

表 7-10 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 7-11 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た分の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、フォーマット済みバッファに行番号の付いたソースを戻します。索引表を使用する方が処理は高速ですが、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

```
DBMS_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 7-12 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 7-13 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み文字バッファ。

構文

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

パラメータ

表 7-14 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레이크・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 7-15 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。「ブレーク・フラグ」(7-9 ページ) 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 に戻す必要のある情報。「情報フラグ」(7-9 ページ) を参照してください。

戻り値

表 7-16 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ーク・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・プログラムがブレーク・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 7-17 SET_BREAKPOINT ファンクションのパラメータ

パラメータ	説明
program	ブレーク・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、この場合のブレーク・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#	ブレーク・ポイントが設定される行。
breakpoint#	正常に完了すると、ブレーク・ポイントを参照するための一意のブレーク・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、ブレーク・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、ブレーク・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	このブレーク・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 7-18 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イク・ポイントを削除します。

構文

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

パラメータ

表 7-19 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イク・ポイント番号。

戻り値

表 7-20 DELETE_BREAKPOINT ファンクションの戻り値

戻り値	説明
success	
error_no_such_breakpt	該当するブレイク・ポイントが存在しません。
error_idle_breakpt	未使用のブレイク・ポイントは削除できません。
error_stale_breakpt	ブレイク・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、既存のブレイク・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 7-21 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 7-22 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 7-23 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 7-24 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 7-25 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みバッファ（改行を含む）。

構文

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

パラメータ

表 7-26 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 7-27 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 7-28 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 7-29 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 7-30 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 7-31 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 7-32 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。

表 7-32 SET_VALUE ファンクションの戻り値

戻り値	説明
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 7-33 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 7-34 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。 <ul style="list-style-type: none"> - 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 7-35 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。「 情報フラグ 」(7-9 ページ) 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 7-36 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 7-37 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、ブレーク・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 7-38 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 7-38 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;
```

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、行トリガーの後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。

サブプログラムの要約

表 8-1 DBMS_DEFER パッケージのサブプログラム

サブプログラム	説明
CALL プロシージャ (8-2 ページ)	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。
COMMIT_WORK プロシージャ (8-4 ページ)	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。
datatype_ARG プロシージャ (8-4 ページ)	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。
TRANSACTION プロシージャ (8-5 ページ)	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 := '' } );
```


パラメータ

表 8-2 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、DBMS_DEFER.datatype_ARG のコールが 1 つずつ必要です。
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。

注意： CALL 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。

例外

表 8-3 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 8-4 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 8-5 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。

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 8-6 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 8-7 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);
```

パラメータ

表 8-8 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 8-9 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。

使用上の注意

TRANSACTION 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

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。

サブプログラムの要約

表 9-1 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
GET_ARG_FORM ファンクショ ン (9-2 ページ)	遅延コールの引数の形式を判別します。
GET_ARG_TYPE ファンクショ ン (9-3 ページ)	遅延コールの引数の型を判別します。
GET_CALL_ARGS プロシージャ (9-5 ページ)	指定されたコールに対するさまざまな引数をテキストで戻します。
GET_datatype_ARG ファンク ション (9-6 ページ)	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： 遅延トランザクションの表示方法の詳細は、『Oracle8i レプリケーション・ガイド』の「遅延トランザクションの表示」を参照してください。エラー・トランザクションの表示方法の詳細は、『Oracle8i レプリケーション・ガイド』の「エラー・トランザクションの表示」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 9-2 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 9-3 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 9-4 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 RPC のパラメータの型が戻されます。

関連項目： 遅延トランザクションの表示方法の詳細は、『Oracle8i レプリケーション・ガイド』の「遅延トランザクションの表示」を参照してください。エラー・トランザクションの表示方法の詳細は、『Oracle8i レプリケーション・ガイド』の「エラー・トランザクションの表示」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 9-5 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 9-6 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 9-7 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さまざま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_ARY,  
    forms       OUT TYPE_ARY,  
    vals        OUT VAL_ARY);
```

パラメータ

表 9-8 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 9-9 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： 遅延トランザクションの表示方法の詳細は、『Oracle8i レプリケーション・ガイド』の「遅延トランザクションの表示」を参照してください。エラー・トランザクションの表示方法の詳細は、『Oracle8i レプリケーション・ガイド』の「エラー・トランザクションの表示」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 9-10 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。プロシージャの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 9-11 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。

サブプログラムの要約

表 10-1 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
ADD_DEFAULT_DEST プロシージャ (10-3 ページ)	DEFDEFAULTDEST ビューに宛先データベースを追加します。
DELETE_DEFAULT_DEST プロシージャ (10-4 ページ)	DEFDEFAULTDEST ビューから宛先データベースを削除します。
DELETE_DEF_DESTINATION プロシージャ (10-4 ページ)	DEFSCHEDULE ビューから宛先データベースを削除します。
DELETE_ERROR プロシージャ (10-5 ページ)	DEFERROR ビューからトランザクションを削除します。
DELETE_TRAN プロシージャ (10-5 ページ)	DEFTRANDEST ビューからトランザクションを削除します。
DISABLED ファンクション (10-6 ページ)	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。
EXCLUDE_PUSH ファンクション (10-7 ページ)	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。
EXECUTE_ERROR プロシージャ (10-8 ページ)	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。
EXECUTE_ERROR_AS_USER プロシージャ (10-9 ページ)	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。
PURGE ファンクション (10-9 ページ)	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、送信済みトランザクションをパージします。
PUSH ファンクション (10-11 ページ)	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。
REGISTER_PROPAGATOR プロシージャ (10-13 ページ)	指定したユーザーをローカル・データベースの伝播担当者として登録します。
SCHEDULE_PURGE プロシージャ (10-14 ページ)	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、送信済み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。
SCHEDULE_PUSH プロシージャ (10-15 ページ)	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。
SET_DISABLED プロシージャ (10-17 ページ)	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。

表 10-1 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
UNREGISTER_PROPAGATOR プロシージャ (10-18 ページ)	ローカル・データベースから伝播担当者としてのユーザーの登録を解除します。
UNSCHEDULE_PURGE プロシージャ (10-19 ページ)	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みトランザクションの自動パージを停止します。
UNSCHEDULE_PUSH プロシージャ (10-19 ページ)	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 10-2 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 10-3 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 10-4 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 10-5 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 10-6 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

このトランザクションは、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 10-7 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

このプロシージャは、現行サイトから指定した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は、TRUE が戻されます。

構文

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

パラメータ

表 10-8 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 10-9 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 10-10 DISABLED ファンクションの例外

例外	説明
NO_DATA_FOUND	DESTINATION が DEFSCHEDULE ビューにありません。

EXCLUDE_PUSH ファンクション

このプロシージャは、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行われます。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 10-11 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 10-12 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、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。NULL と NULL 以外のパラメータを不正に組み合わせて使用すると、ORA-24275 エラーが発生します。

構文

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

パラメータ

表 10-13 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。

例外

表 10-14 EXECUTE_ERROR プロシージャの例外

例外	説明
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。NULL と NULL 以外のパラメータを不正に組み合わせて使用すると、ORA-24275 エラーが発生します。

構文

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

パラメータ

表 10-15 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 10-16 EXECUTE_ERROR_AS_USER プロシージャの例外

例外	説明
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、送信済み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 10-17 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が削減でき、purge_method_precise を指定すると精度が高くなります。
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 ファンクションで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 10-18 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 10-19 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。

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（実行、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 10-20 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 = シリアル伝播、 $n > 0 = n$ 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、1 = 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。
heap_size	パラレル伝播スケジューリングのために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に、デフォルト設定が自動的に計算されます。オラクル社カスタマ・サポート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。

表 10-20 PUSH ファンクションのパラメータ

パラメータ	説明
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（可能な場合はクリーンに）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。
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た場合は、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

表 10-21 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 10-22 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。
missingpropagator	伝播担当者が存在しません。

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・データベースの伝播担当者として登録します。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 10-23 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 10-24 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伝播担当者です。

表 10-24 REGISTER_PROPAGATOR プロシージャの例外

例外	説明
duplicatepropagator	別の伝播担当者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、送信済み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 10-25 SCHEDULE_PURGE プロシージャのパラメータ

パラメータ	説明
interval	<p>パ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。</p> <p>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。</p>
next_date	<p>サイトのキューから送信済みトランザクションをパ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。</p> <p>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。</p>

表 10-25 SCHEDULE_PURGE プロシージャのパラメータ

パラメータ	説明
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・キューのパージ方法を制御します。purge_method_quick を指定するとコストが削減でき、purge_method_precise を指定すると精度が高くなります。
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。

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 10-26 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	<p>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 DEFSCCHEDULE 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。</p> <p>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。</p>
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 DEFSCCHEDULE 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。
parallelism	0 = シリアル伝播、 $n > 0 = n$ 個の平行・サーバー・プロセスを使用する平行伝播、1 = 1 つの平行・サーバー・プロセスのみ使用する平行伝播。
heap_size	平行伝播スケジューリングのために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に、デフォルト設定が自動的に計算されます。オラクル社カスタム・サポート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。

表 10-26 SCHEDULE_PUSH プロシージャのパラメータ

パラメータ	説明
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使用します。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、所定の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です。この処理でキューは送信されませんが、その後で PUSH を起動すると所定の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 10-27 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 10-28 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・データベースから伝播担当者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。

- 指定した伝播担当者を 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を含みます)。
- 指定した伝播担当者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 10-29 UNREGISTER_PROPAGATOR プロシージャのパラメータ

パラメータ	説明
username	伝播担当者のユーザー名。
timeout	タイムアウト (秒数)。伝播担当者が使用中の場合は、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 10-30 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伝播担当者ではありません。
propagator_inuse	伝播担当者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みトランザクションの自動バージを停止します。

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 10-31 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 10-32 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 11-1 DBMS_DESCRIBE のエラー

エラー	エラー・メッセージ
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: procedure 'X' within package 'Y' does not exist.
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 のプロシージャは DESCRIBE_PROCEDURE 1 つのみです。

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 11-2 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>

表 11-2 DBMS_DESCRIBE.DESCRIBE_PROCEDURE のパラメータ

パラメータ	説明
reserved1 reserved2	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。
overload	プロシージャの署名に割り当てられた一意の番号。 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。
position	パラメータ・リスト内の引数の位置。 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。
level	引数がレコードなどの複合型の場合は、そのデータ型のレベルが戻されません。 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。
argument_name	記述するプロシージャに関連付けられた引数の名前。
datatype	記述する引数の Oracle データ型。 データ型とその数値型のコードは、次のとおりです。 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	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。
in_out	パラメータのモードを記述します。値は次のとおりです。 0 IN 1 OUT 2 IN OUT
length	記述する引数のデータ長 (バイト)。

表 11-2 DBMS_DESCRIBE.DESCRIBE_PROCEDURE のパラメータ

パラメータ	説明
precision	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。
scale	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。
radix	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。
spare	将来使用するために予約されています。

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,  
    ppl sint IN    PLS_INTEGER,  
    pbool   IN    BOOLEAN,  
    pnat    IN    NATURAL,  
    ppos    IN    POSITIVE,  
    pposn   IN    POSITTIVEN,  
    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するとします。

```

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;
        dty            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,
        dty,
        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_DISTRICTED_TRUST_ADMIN

DBMS_DISTRICTED_TRUST_ADMIN プロシーダは、Trusted Database リストをメンテナンスします。

注意： このリストは、中央認可レベル (Central Authority: CA) のリストとともに使用されます。これによって、特定のサーバからの特権データベース・リンクが受入れ可能かどうかを判別します。CA でリストされているかどうかに関係なく、特定のサーバを Trusted Database リストにローカルでリストできます。

要件

DBMS_DISTRIBUTED_TRUST_ADMIN を実行するためには、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 からの選択には、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。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には、ALLOW_SERVER プロシージャで特定のサーバーを信頼しても何も影響はありません。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には、DENY_SERVER プロシージャで特定のサーバーを拒否しても影響はありません。

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 12-1 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
ALLOW_ALL プロシージャ (12-2 ページ)	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。
ALLOW_SERVER プロシージャ (12-3 ページ)	特定のサーバーへのアクセスを可能にします。リストにすべて deny が指定されている場合でも有効です。
DENY_ALL プロシージャ (12-4 ページ)	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。
DENY_SERVER プロシージャ (12-4 ページ)	特定のサーバーへのアクセスを拒否します。リストに allow 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 Database リストを空にし、Oracle Security Server などの中央認可レベルで信頼されているすべてのサーバーへのアクセス許可を指定します。

ビュー TRUSTED_SERVERS には、"TRUSTED ALL" が表示されます。これは、Oracle Security Server などの中央認可レベルで現在すべてのサーバーが信頼されていることを示します。

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、中央認可レベルで信頼されているとしてリストされている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 12-2 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、特定のサーバーへのアクセスを可能にします。リストにすべて拒否が指定されている場合でも有効です。

ビュー TRUSTED_SERVERS には、"UNTRUSTED ALL" が表示されます。これは、現在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 12-3 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 を使用して信頼を変更したことが一度もない場合、デフォルトでは、Oracle Security Server で定義したすべてのサーバーが信頼されているとみなされます。

```
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_GRANTED';

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_GRANTED';

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_GRANTED';

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 13-1 DBMS_HS パッケージのサブプログラム

サブプログラム	説明
ALTER_BASE_CAPS プロシージャ (13-8 ページ)	HS\$_BASE_CAPS 表の行を変更します。
ALTER_BASE_DD プロシージャ (13-9 ページ)	HS\$_BASE_DD 表の行を変更します。
ALTER_CLASS_CAPS プロシージャ (13-9 ページ)	HS\$_CLASS_CAPS 表の内容を変更します。
ALTER_CLASS_DD プロシージャ (13-9 ページ)	HS\$_CLASS_DD 表の内容を変更します。
ALTER_CLASS_INIT プロシージャ (13-9 ページ)	HS\$_CLASS_INIT 表の内容を変更します。
ALTER_FDS_CLASS プロシージャ (13-10 ページ)	HS\$_FDS_CLASS 表の内容を変更します。
ALTER_FDS_INST プロシージャ (13-10 ページ)	HS\$_FDS_INST 表の内容を変更します。
ALTER_INST_CAPS プロシージャ (13-10 ページ)	\$HS_INST_CAPS 表の内容を変更します。
ALTER_INST_DD プロシージャ (13-11 ページ)	HS\$_INST_DD 表の内容を変更します。
ALTER_INST_INIT プロシージャ (13-11 ページ)	HS\$_INST_INIT 表の内容を変更します。
COPY_CLASS プロシージャ (13-11 ページ)	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。
COPY_INST プロシージャ (13-12 ページ)	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。
CREATE_BASE_CAPS プロシージャ (13-12 ページ)	HS\$_BASE_CAPS 表に行を作成します。
CREATE_BASE_DD プロシージャ (13-12 ページ)	HS\$_BASE_DD 表に行を作成します。
CREATE_CLASS_CAPS プロシージャ (13-12 ページ)	HS\$_CLASS_CAPS 表に行を作成します。
CREATE_CLASS_DD プロシージャ (13-13 ページ)	HS\$_CLASS_DD 表に行を作成します。

表 13-1 DBMS_HS パッケージのサブプログラム

サブプログラム	説明
CREATE_CLASS_INIT プロシージャ (13-13 ページ)	HS\$_CLASS_INIT 表に行を作成します。
CREATE_FDS_CLASS プロシージャ (13-13 ページ)	HS\$_FDS_CLASS 表に行を作成します。
CREATE_FDS_INST プロシージャ (13-14 ページ)	HS\$_FDS_INST 表に行を作成します。
CREATE_INST_CAPS プロシージャ (13-14 ページ)	HS\$_INST_CAPS 表に行を作成します。
CREATE_INST_DD プロシージャ (13-14 ページ)	HS\$_INST_DD 表に行を作成します。
CREATE_INST_INIT プロシージャ (13-15 ページ)	HS\$_INST_INIT 表に行を作成します。
DROP_BASE_CAPS プロシージャ (13-15 ページ)	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。
DROP_BASE_DD プロシージャ (13-15 ページ)	table_name の指定どおりに、HS\$_BASE_DD 表から行を削除します。
DROP_CLASS_CAPS プロシージャ (13-15 ページ)	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。
DROP_CLASS_DD プロシージャ (13-16 ページ)	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。
DROP_CLASS_INIT プロシージャ (13-16 ページ)	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。
DROP_FDS_CLASS プロシージャ (13-16 ページ)	FDS_CLASS_NAME の指定どおりに、HS\$_FDS_CLASS の行を削除します。
DROP_FDS_INST プロシージャ (13-16 ページ)	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。
DROP_INST_CAPS プロシージャ (13-17 ページ)	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。
DROP_INST_DD プロシージャ (13-17 ページ)	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。
DROP_INST_INIT プロシージャ (13-17 ページ)	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。

表 13-1 DBMS_HS パッケージのサブプログラム

サブプログラム	説明
REPLACE_BASE_CAPS プロシージャ (13-17 ページ)	HS\$_BASE_CAPS 表の行を作成または置換します。
REPLACE_BASE_DD プロシージャ (13-18 ページ)	HS\$_BASE_DD 表の行を作成または置換します。
REPLACE_CLASS_CAPS プロシージャ (13-18 ページ)	HS\$_CLASS_CAPS 表上で作成または置換を実行します。
REPLACE_CLASS_DD プロシージャ (13-18 ページ)	HS\$_CLASS_DD 表上で作成または置換を実行します。
REPLACE_CLASS_INIT プロシージャ (13-19 ページ)	HS\$_CLASS_INIT 表の行を作成または更新します。
REPLACE_FDS_CLASS プロシージャ (13-19 ページ)	HS\$_FDS_CLASS 表上で作成または置換操作を実行します。
REPLACE_FDS_INST プロシージャ (13-19 ページ)	HS\$_FDS_INST 表の行を作成または置換します。
REPLACE_INST_CAPS プロシージャ (13-20 ページ)	HS\$_INST_CAPS 表上で作成または置換を実行します。
REPLACE_INST_DD プロシージャ (13-20 ページ)	HS\$_INST_DD 表上で作成または置換操作を実行します。
REPLACE_INST_INIT プロシージャ (13-21 ページ)	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

```
DBMS_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

```
DBMS_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

```
DBMS_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' (変換済み) または 'M' (代替済み) 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' (変換済み) または 'M' (代替済み) 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

```
DBMS_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

```
DBMS_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

```
DBMS_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

```
DBMS_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。

```
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。

```
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

```
DBMS_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

```
DBMS_HS.REPLACE_BASE_CAPS (  
    FDS_INST_NAME  IN VARCHAR2,  
    FDS_CLASS_NAME IN VARCHAR2,  
    CAP_NUMBER     IN NUMBER,  
    INIT_VALUE_NAME IN VARCHAR2,  
    new_fds_class_name VARCHAR2,  
    new_fds_class_comments VARCHAR2 default '-');
```

```
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

```
DBMS_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の詳細は、『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 14-1 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
BIND_VARIABLE プロシージャ (14-3 ページ)	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。
BIND_VARIABLE_RAW プロシージャ (14-4 ページ)	RAW 型の IN 変数をバインドします。
BIND_OUT_VARIABLE プロシージャ (14-5 ページ)	OUT 変数を PL/SQL プログラム変数にバインドします。
BIND_OUT_VARIABLE_RAW プロシージャ (14-7 ページ)	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。
BIND_INOUT_VARIABLE プロシージャ (14-8 ページ)	IN OUT バインド変数をバインドします。
BIND_INOUT_VARIABLE_RAW プロシージャ (14-9 ページ)	データ型 RAW の IN OUT バインド変数をバインドします。
CLOSE_CURSOR プロシージャ (14-10 ページ)	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。
EXECUTE_IMMEDIATE プロシージャ (14-11 ページ)	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。
EXECUTE_NON_QUERY ファンクション (14-12 ページ)	(SELECT 以外の) SQL 文を実行します。
FETCH_ROW ファンクション (14-13 ページ)	問合せから行をフェッチします。
GET_VALUE プロシージャ (14-14 ページ)	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。
GET_VALUE_RAW プロシージャ (14-15 ページ)	データ型 RAW が対象である以外は GET_VALUE と同じです。
OPEN_CURSOR ファンクション (14-16 ページ)	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 14-1 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
PARSE プロシージャ (14-17 ページ)	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、[「BIND_VARIABLE_RAW プロシージャ」](#)(14-4 ページ)を使用します。

パラメータ

表 14-2 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 14-3 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 型の 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 14-4 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 14-4 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 14-5 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 : WINDS, 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、「[BIND_OUT_VARIABLE_RAW プロシージャ](#)」(14-7 ページ) を参照してください。

パラメータ

表 14-6 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 14-7 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 14-8 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 バインド変数がある値を格納する変数。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 14-9 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。

表 14-9 BIND_OUT_VARIABLE_RAW プロシージャの例外

例外	説明
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

```
DEMS_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、「[BIND_INOUT_VARIABLE_RAW プロシージャ](#)」(14-9 ページ) を参照してください。

パラメータ

表 14-10 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 14-10 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 14-11 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。

プラグマ

Purity level defined : WINDS, 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 14-12 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 14-13 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 14-14 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 14-15 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 PASSTHROUGH 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 14-16 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 14-17 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 14-18 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 14-19 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 14-20 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 14-21 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 : WNDS

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、「[GET_VALUE_RAW プロシージャ](#)」(14-15 ページ) を参照してください。

パラメータ

表 14-22 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 14-23 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 : WNDS

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 14-24 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 14-25 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 14-26 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 14-27 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 14-28 PARSE 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

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 15-1 DBMS_IOT パッケージのサブプログラム

サブプログラム	説明
BUILD_CHAIN_ROWS_TABLE プロシージャ (15-2 ページ)	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。
BUILD_EXCEPTIONS_TABLE プロシージャ (15-3 ページ)	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 15-2 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 互換で稼働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 15-3 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)

16

DBMS_JOB

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 を実行、または DBMS_JOB を代行する権限です。DBMS_JOB では、ユーザー自身が所有しているジョブ以外のジョブに触れることはできません。

サブプログラムの要約

表 16-1 DBMS_JOB パッケージのサブプログラム

サブプログラム	説明
SUBMIT プロシージャ (16-3 ページ)	新規ジョブをジョブ・キューに送信します。
REMOVE プロシージャ (16-4 ページ)	指定したジョブをジョブ・キューから削除します。
CHANGE プロシージャ (16-5 ページ)	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。
WHAT プロシージャ (16-6 ページ)	指定したジョブに関するジョブの記述を変更します。
NEXT_DATE プロシージャ (16-6 ページ)	指定したジョブの次の実行時間を変更します。
INSTANCE プロシージャ (16-7 ページ)	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。
INTERVAL プロシージャ (16-7 ページ)	指定したジョブの実行間隔を変更します。
BROKEN プロシージャ (16-8 ページ)	ジョブの実行を禁止します。
RUN プロシージャ (16-9 ページ)	指定したジョブを強制的に実行します。
USER_EXPORT プロシージャ (16-10 ページ)	指定したジョブをエクスポート用に再作成します。
USER_EXPORT プロシージャ (16-10 ページ)	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。ジョブ番号 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 16-2 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 に設定すると、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 16-3 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 16-4 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。
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 16-5 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 16-6 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 16-7 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り、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 16-8 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

```
DEMS_JOB.BROKEN (  
  job      IN  BINARY_INTEGER,  
  broken   IN  BOOLEAN,  
  next_date IN  DATE DEFAULT SYSDATE);
```

パラメータ

表 16-9 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。

表 16-9 Broken プロシージャのパラメータ

パラメータ	説明
next_data	次のリフレッシュ日付。

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

```
DBMS_JOB.RUN (  
    job      IN  BINARY_INTEGER,  
    force    IN  BOOLEAN DEFAULT FALSE);
```

パラメータ

表 16-10 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 DBMS_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

```
DBMS_JOB.USER_EXPORT (  
    job      IN      BINARY_INTEGER,  
    mycall   IN OUT  VARCHAR2);
```

パラメータ

表 16-11 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 16-12 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。

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、現在のコール・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 ロケータを基にして動作します。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。

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みであることを示す 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大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 17-1 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 パッファ (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パッファ (CLOB および NCLOB とともに使用されます)。
INTEGER	パッファまたは 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 17-2 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、現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 procl authid definer ...
```

または

```
CREATE PROCEDURE procl authid current_user ....
```

関連項目： AUTHID と権限の詳細は、『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。この機能については、『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 機能」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。

```
JAI6SJISFIXED
```

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すべき事項です。

- %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 パラメータの型が CLOB の場合、バッファには CHAR データが含まれている必要があります。

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 であることが必要です。

- 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、オープン済みの 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。この状況を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 パッケージ、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 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 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 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 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 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 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 LOB にはアクセスできません。テンポラリ 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うべきではありませんが、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 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 LOB 数に関する情報が含まれています。`v$` ビューは、DBA が使用するためのものです。セッションから、Oracle はどのユーザーがそのテンポラリ LOB を所有しているかを判断できます。`V$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 LOB 用に使用さ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 LOB が使用しているテンポラリ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, dbms_lob.session);
  -- 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, dbms_lob.session);
  -- 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、『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 17-3 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 17-4 DBMS_LOB のサブプログラム (1 / 2 ページ)

サブプログラム	説明
APPEND プロシージャ (17-14 ページ)	ソース LOB の内容を宛先 LOB に追加します。
CLOSE プロシージャ (17-15 ページ)	オープンしている内部または外部 LOB をクローズします。
COMPARE ファンクション (17-16 ページ)	2 つの LOB 全体、または 2 つの LOB の一部を比較します。
COPY プロシージャ (17-19 ページ)	ソース LOB 全体または一部を宛先 LOB にコピーします。
CREATETEMPORARY プロシージャ (17-21 ページ)	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。
ERASE プロシージャ (17-22 ページ)	LOB 全体または一部を消去します。
FILECLOSE プロシージャ (17-24 ページ)	ファイルをクローズします。
FILECLOSEALL プロシージャ (17-25 ページ)	オープンしているファイルをすべてクローズします。

表 17-4 DBMS_LOB のサブプログラム (2 / 2 ページ)

サブプログラム	説明
FILEEXISTS ファンクション (17-26 ページ)	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。
FILEGETNAME プロシージャ (17-28 ページ)	ディレクトリ別名とファイル名を取得します。
FILEISOPEN ファンクション (17-29 ページ)	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。
FILEOPEN プロシージャ (17-30 ページ)	ファイルをオープンします。
FREETEMPORARY プロシージャ (17-32 ページ)	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。
GETCHUNKSIZE ファンクション (17-33 ページ)	LOB 値を格納する LOB チャンクの使用領域容量を戻します。
GETLENGTH ファンクション (17-34 ページ)	LOB 値の長さを取得します。
INSTR ファンクション (17-36 ページ)	LOB にあるパターンの <i>n</i> 番目のオカレンスのマッチング位置を戻します。
ISOPEN ファンクション (17-39 ページ)	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。
ISTEMPORARY ファンクション (17-40 ページ)	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。
LOADFROMFILE プロシージャ (17-40 ページ)	BFILE データを内部 LOB にロードします。
OPEN プロシージャ (17-43 ページ)	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。
READ プロシージャ (17-44 ページ)	指定されたオフセット以降の LOB データを読み込みます。
SUBSTR ファンクション (17-47 ページ)	指定されたオフセット以降の LOB 値の一部を戻します。
TRIM プロシージャ (17-50 ページ)	LOB 値を指定された長さに切り捨てます。
WRITE プロシージャ (17-52 ページ)	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。
WRITEAPPEND プロシージャ (17-54 ページ)	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 17-5 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 17-6 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 $\text{Max}(\text{length}(\text{clob1}), \text{length}(\text{clob2}))$ 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 17-7 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 17-8 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。

表 17-8 BFILE 操作に関する COMPARE ファンクションの例外

例外	説明
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 17-9 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。

表 17-9 COPY プロシージャのパラメータ

パラメータ	説明
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。
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 17-10 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 17-11 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, DBMS_LOB.SESSION)
```

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、「[TRIM プロシージャ](#)」(17-50 ページ) 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 17-12 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 17-13 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：「[TRIM プロシージャ](#)」(17-50 ページ)

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 17-14 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 17-15 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：「[FILEOPEN プロシージャ](#)」(17-30 ページ)
「[FILECLOSEALL プロシージャ](#)」(17-25 ページ)

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 17-16 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：「[FILEOPEN プロシージャ](#)」(17-30 ページ)、「[FILECLOSE プロシージャ](#)」(17-24 ページ)

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 17-17 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 17-18 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 17-19 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だけで、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 17-20 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 17-21 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はあります。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 17-22 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 17-23 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：「[FILEEXISTS ファンクション](#)」(17-26 ページ)

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 17-24 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・アクセスは読み取り専用です。

戻り値

なし。

例外

表 17-25 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：「[FILECLOSE プロシージャ](#)」(17-24 ページ)
「[FILECLOSEALL プロシージャ](#)」(17-25 ページ)

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 17-26 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.session);
  dbms_lob.createtemporary(b, TRUE, dbms_lob.session);
  ...
  -- 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 17-27 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取り / 書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、書込みの場合にはさらに利点があります。すべての書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 つのチャンクが十分となるまで 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 17-28 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 17-29 INSTR ファン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 17-30 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 17-31 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。

表 17-31 BFILES に関する INSTR ファンクションの例外

例外	説明
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がありません。

例

```
CREATE OR REPLACE PROCEDURE Example_12a IS
    lobd          CLOB;
    pattern        VARCHAR2 := 'abode';
    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：「SUBSTR ファンクション」(17-47 ページ)

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 17-32 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

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);
```

パラメータ

表 17-33 ISTEPMORARY 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	ブール値。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 17-34 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。

表 17-34 LOADFROMFILE プロシージャのパラメータ

パラメータ	説明
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 17-35 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_12f 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（内部または外部）で、LOB をオープンします。有効なモードは読取り専用と読取り書込みです。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 17-36 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 17-37 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。

表 17-37 READ プロシージャのパラメータ

パラメータ	説明
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	読み取り操作のアウトプット・バッファ。

戻り値

なし。

例外

表 17-38 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 17-39 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。

表 17-39 BFILE に関する READ プロシージャの例外

例外	説明
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 17-40 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 17-41 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 17-42 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：「[INSTR ファンクション](#)」(17-36 ページ)、「[READ プロシージャ](#)」(17-44 ページ)

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 17-43 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 17-44 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：「[ERASE プロシージャ](#)」(17-22 ページ)、「[WRITEAPPEND
プロシージャ](#)」(17-54 ページ)

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 17-45 WRITE プロシージャのパラメータ

パラメータ	説明
lob_loc	書込み先の内部 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）。

表 17-45 WRITE プロシージャのパラメータ

パラメータ	説明
offset	書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	書込み用の入力バッファ。

戻り値

なし。

例外

表 17-46 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：[「APPEND プロシージャ」](#)(17-14 ページ)、[「COPY プロシージャ」](#)(17-19 ページ)

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 17-47 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 17-48 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：「[APPEND プロシージャ](#)」(17-14 ページ)、[「COPY プロシージャ](#)」(17-19 ページ)、[「WRITE プロシージャ](#)」(17-52 ページ)

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パー・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。カパー・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） Enterprise Manager のモニター機能は、インスタンスのロック情報を表示するための 2 つのモニターを提供します。Enterprise Manager モニターの詳細は、『Oracle Server Manager ユーザーズ・ガイド』を参照してください。

UTLLOCKT.SQL UTLLOCKT.SQL スクリプトは、文字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 18-1 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 18-2 DBMS_LOCK パッケージのサブプログラム

サブプログラム	説明
ALLOCATE_UNIQUE プロシージャ (18-4 ページ)	名前付きロックに一意のロック ID を割り当てます。
REQUEST ファンクション (18-6 ページ)	特定のモードのロックを要求します。
CONVERT ファンクション (18-7 ページ)	ロックを別のモードに変換します。
RELEASE ファンクション (18-9 ページ)	ロックを解放します。
SLEEP プロシージャ (18-10 ページ)	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、領域回復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 18-3 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、領域を回復できます。

エラー

```
ORA-20000, ORU-10003: Unable to find or insert lock <lockname> into catalog dbms_  
lock_allocated.
```

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します。

パラメータ

表 18-4 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、VS ビューと Enterprise Manager モニターに表示されるときにロックの略称です。 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 18-4 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 18-5 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 18-6 CONVERT ファンクションのパラメータ

パラメータ	説明
id or lockhandle	変更したいロック・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	指定のロックに割り当てる新規モード。 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、VS ビューと Enterprise Manager モニターに表示されるときに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。
timeout	ロック・モードの変更を待つ秒数。 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。

戻り値

表 18-7 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。

表 18-7 CONVERT ファンクションの戻り値

戻り値	説明
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 18-8 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 18-9 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。

表 18-9 RELEASE ファンクションの戻り値

戻り値	説明
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、指定の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 18-10 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割り込む可能性があります。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・データは、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ため、データ回復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回復できます。

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_FILES 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 ビューを設定して、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（メディア回復に適用される順序と同じ）にあるすべての列を戻します。

関連項目：「例」(19-8 ページ)および [第 19 章の「DBMS_LOGMNR」](#)

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。これは、Options パラメータを NEW に設定した ADD_LOGFILE がすでに最低 1 つ起動している場合のみ有効です。
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。これは、ログ・ファイルが以前にリストに追加されていない場合は無効です。

START_LOGMNR オプション・フラグの定数

USER_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必要があります。
-------------	---

ブレース・ホルダ列の使用法

V\$LOGMNR_CONTENTS 表には、ブレース・ホルダ列の複数のセットが含まれています。各ブレース・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。各ブレース・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。ブレース・ホルダ列を割り当てた後は、そのブレース・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、割当て "colmap = SCOTT EMP (1, EMPNO);" は、PH1 ブレース・ホルダ列セットを表と列 (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。

各ブレース・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 ブレース・ホルダ列セットには、EMP 表への変更のみを選択する割当てと EMP 表名を SELECT に追加する割当ての 2 つがあります。

```
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 ph1_name='CUSTID'
AND ph1_redo='12345';
```

logmnr.opt ブレース・ホルダ列の使用法

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 デクシヨナリ・ファイル (UTL_FILE_DIR) と同じデクシヨナ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、割当て済みの列は現行の LogMiner デクシヨナリに対して検証されます。割当て済み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ます)

<table> 表名。

<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、すべて大文字である必要があります。

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。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 19-1 DBMS_LOGMNR パッケージのサブプログラム

サブプログラム	説明
ADD_LOGFILE プロシージャ (19-5 ページ)	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。
START_LOGMNR プロシージャ (19-6 ページ)	ログ分析ツールを初期化します。
END_LOGMNR プロシージャ (19-8 ページ)	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 ビューから情報を選択するために、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 19-2 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。 「ADD_LOGFILE オプション・フラグの定数」 (19-2 ページ) を参照してください。

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 19-3 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。
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。 「START_LOGMNR オプション・フラグの定数」 (19-2 ページ) と 「logmnr.opt プレース・ホルダ列の使用方法」 (19-3 ページ) を参照してください。

例外

プロシージャは、次の理由から ORA-1280 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。
5. DictFileName が存在しない場合。
6. REDO ログを生成するデータベースが、DictFilename で指定したディクショナリ・ファイルを生成したデータベースと異なる場合。
7. logmnr.opt ファイルを使用しないで DBMS_LOGMNR.USE_COLMAP を設定した場合。
8. logmnr.opt ファイルで DBMS_LOGMNR.USE_COLMAP が設定されたが、構文エラーがある場合。
9. endScn が startScn より小さい場合。
10. endTime が startTime 以前の場合 (startScn と endScn も指定されていない)。

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

```
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;
```

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・ファイルを生成了と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 20-1 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する必要があります。

BUILD プロシージャは、init.ora で UTIL_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 SERVER OUTPUT 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（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
```

```
SVRMGR> EXECUTE dbms_logmnr_d.build('l_dictionary.ora',  
SVRMGR> '/oracle/database/');
```

関連項目： [第 19 章の「DBMS_LOGMNR」](#)

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使用方法の詳細は、該当の章を参照してください。

サブプログラムの要約

表 21-1 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
BEGIN_INSTANTIATION プロシージャ (21-2 ページ)	レプリケート・オブジェクト・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。
BEGIN_LOAD プロシージャ (21-3 ページ)	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。
END_INSTANTIATION プロシージャ (21-4 ページ)	レプリケート・オブジェクト・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。
END_LOAD プロシージャ (21-5 ページ)	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。
RESUME_SUBSET_OF_MASTERS プロシージャ (21-6 ページ)	レプリケート・オブジェクト・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。

構文

```
DBMS_OFFLINE_OG.BEGIN_INSTANTIATION (  
    gname      IN   VARCHAR2,  
    new_site   IN   VARCHAR2,  
    fname      IN   VARCHAR2);
```

パラメータ

表 21-2 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・グループ名。
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。

表 21-2 BEGIN_INSTANTIATION プロシージャのパラメータ

パラメータ	説明
fname	これは、内部使用のためのシステム・パラメータです。オラクル社カスタマ・サポート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 21-3 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ら、オラクル社カスタマ・サポート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。

構文

```
DBMS_OFFLINE_OG.BEGIN_LOAD (  
  gname      IN   VARCHAR2,  
  new_site   IN   VARCHAR2);
```

パラメータ

表 21-4 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 21-5 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・インスタンス化を完了します。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。

構文

```
DBMS_OFFLINE_OG.END_INSTANTIATION (  
    gname      IN  VARCHAR2,  
    new_site   IN  VARCHAR2);
```

パラメータ

表 21-6 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・グループ名。
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 21-7 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. nomasterdef	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・インスタンス化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。

構文

```
DBMS_OFFLINE_OG.END_LOAD (  
    gname      IN   VARCHAR2,  
    new_site   IN   VARCHAR2,  
    fname      IN   VARCHAR2);
```

パラメータ

表 21-8 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・グループ名。
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	これは、内部使用のためのシステム・パラメータです。オラクル社カスタマ・サポート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 21-9 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ら、オラクル社カスタマ・サポートに連絡してください。
dbms_repcat.flavor_ contains	この例外が発生したら、オラクル社カスタマ・サポート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。

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 21-10 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・グループ名。
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	このパラメータが TRUE の場合、保留中の RepCat 管理要求は無視され、各マスターで可能な限り早く通常のレプリケーション・アクティビティを復元します。これは、緊急の状況でのみ指定してください。 このパラメータが FALSE の場合、マスターで gname に対して保留中の 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・アクティビティを復元します。

例外

表 21-11 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。

サブプログラムの要約

表 22-1 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
BEGIN_LOAD プロシージャ (22-2 ページ)	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。
END_LOAD プロシージャ (22-3 ページ)	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。

構文

```
DBMS_OFFLINE_SNAPSHOT.BEGIN_LOAD (  
    gname           IN   VARCHAR2,  
    sname           IN   VARCHAR2,  
    master_site     IN   VARCHAR2,  
    snapshot_otype  IN   VARCHAR2,  
    storage_c       IN   VARCHAR2 := ' ',  
    comment         IN   VARCHAR2 := ' ',  
    min_communication IN  BOOLEAN  := TRUE);
```

パラメータ

表 22-2 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のスキーマ名。
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_onsame	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領域オプション。
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。列の古い値は、それがキー列の場合または変更された列グループにある列の場合のみ、更新トリガーにより送信されます。

例外

表 22-3 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を指定のマスター・サイトで検索できません。
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実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。

構文

```
DBMS_OFFLINE_SNAPSHOT.END_LOAD (  
    gname          IN  VARCHAR2,  
    sname          IN  VARCHAR2,  
    snapshot_ename IN  VARCHAR2);
```

パラメータ

表 22-4 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のスキーマ名。
snapshot_ename	スナップショット名。

例外

表 22-5 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意思決定支援のための機能を提供します。

関連項目： これらの機能の使用方法は、『Oracle8i 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 23-1 DBMS_OLAP エラー・メッセージ

エラー・コード	エラー・メッセージ
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。
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	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	統計情報がテーブル / 列にありません。
QSM-00509	無効なパラメータ
QSM-00510	統計情報がサマリーにありません。
QSM-00511	無効なファクト表がファクト・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 23-2 DBMS_OLAP パッケージのサブプログラム

サブプログラム	説明
ESTIMATE_SUMMARY_SIZE プロシージャ (23-3 ページ)	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。
EVALUATE_UTILIZATION プロシージャ (23-4 ページ)	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。
EVALUATE_UTILIZATION_W プロシージャ (23-4 ページ)	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。
RECOMMEND_MV プロシージャ (23-5 ページ)	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。
RECOMMEND_MV_W プロシージャ (23-7 ページ)	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。
VALIDATE_DIMENSION プロシージャ (23-8 ページ)	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 23-3 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 23-4 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。

また、このプロシージャにより、[WORK\\$_IDEAL_MVIEW](#) と [WORK\\$_MVIEW_USAGE](#) ビューも作成します。

関連項目：「[DBMS_OLAP インタフェース表](#)」(23-9 ページ)

構文

```
DBMS_OLAP.EVALUATE_UTILIZATION_W;
```


パラメータ

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

パラメータ	I/O	説明
MVIEWS_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・ビューの使用方法的の表。 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、[「RECOMMEND_MV_W プロシージャ」](#) (23-7 ページ) 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 23-6 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・ビューを格納するために使用できる最大記憶領域 (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、保持すべき既存のマテリアライズド・ビュー記憶領域を 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 23-7 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、`WORKS_IDEAL_MVIEW` と `WORKS_MVIEW_USAGE` のビューも作成します。

関連項目：「[DBMS_OLAP インタフェース表](#)」(23-9 ページ)

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 23-8 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領域（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。

表 23-8 RECOMMEND_MV_W プロシージャのパラメータ

パラメータ	説明
retention_pct	実際または仮定されるワークロードに基づき、保持すべき既存のマテリアライズド・ビュー記憶領域を 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 23-9 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 23-10 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 23-11 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 23-12 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。

表 23-12 MVIEW\$_RECOMMENDATION

列名	型	制約	説明
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。
measures_list	VARCHAR2(2000)	アクションが DROP の場合は NULL、CREATE の 場合は NULL 以外	<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。 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。 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。
storage_in_bytes	NUMBER	>= 0	実際または見積りした記憶領域 (バイト)

表 23-12 MVIEW\$_RECOMMENDATION

列名	型	制約	説明
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、<グループ化ファンクション>(<式>) 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。

- pct_performance_gain

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。
- benefit_to_cost_ratio

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、最も改善度が多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 23-13 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、[MVIEWS_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 23-14 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。

表 23-14 WORK\$_IDEAL_MVIEW

列名	型	制約	説明
actual_benefit	NUMBER	>0	このマテリアライズド・ビューの使用によるパフォーマンス上の実際の改善度。

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 24-1 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 の各ファイルは、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る）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 スタアド・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 25-1 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 26-1 DBMS_OUTPUT エラー

エラー	エラー・メッセージ
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 (これらのコールだけが、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(S) 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 26-2 DBMS_OUTPUT パッケージのサブプログラム

サブプログラム	説明
ENABLE プロシージャ (26-3 ページ)	メッセージの出力を使用可能にします。
DISABLE プロシージャ (26-4 ページ)	メッセージの出力を使用禁止にします。
PUT および PUT_LINE プロシージャ (26-4 ページ)	行をバッファに設定します。
PUT および PUT_LINE プロシージャ (26-4 ページ)	行の一部をバッファに設定します。
NEW_LINE プロシージャ (26-6 ページ)	PUT を使用して作成した行を終了します。
GET_LINE および GET_LINES プロシージャ (26-6 ページ)	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 26-3 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 26-4 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設定されます。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);
DBMS_OUTPUT.NEW_LINE;
```

パラメータ

表 26-5 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
a	バッファに設定する項目。

エラー

表 26-6 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(S) 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(S) 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 26-7 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。バッファに設定さ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 26-8 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、バッファに設定された単一行の情報を戻します。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 26-9 GET_LINES プロシージャのパラメータ

パラメータ	説明
lines	バッファに設定さ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、**例 2** (26-9 ページ) 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 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 indentation 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 57 章の「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での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 - ハッシュ複合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 - ハッシュ複合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、指定の並列性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,<idx_name>,<tab_name>,FALSE);
```

複合パーティションについて、プロシージャは、その複合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る）のみを再作成します。

```
EXECUTE dbms_pclxutil.build_part_index(4,4,<idx_name>,<tab_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,<idx_name>,<tab_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だけです。
- 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 (特にさまざまな記憶領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 27-1 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	ローカル索引ビルド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状況が発生します)。
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状況が発生します)。
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。

```
SVRMGR> 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
SVRMGR>
```

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ーに存続します。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。

メモリー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ード・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（効果的に）行うことができますため、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）: ボーリングするための待機プロセスを要求せずに、別のプロセスをポストできます。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、現在の値を取得します。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だけのカバー・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 28-1 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 28-2 DBMS_PIPE パッケージのサブプログラム

サブプログラム	説明
CREATE_PIPE ファンクション (28-5 ページ)	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)。
PACK_MESSAGE プロシージャ (28-7 ページ)	ローカル・バッファにメッセージを作成します。
SEND_MESSAGE ファンクション (28-8 ページ)	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。
RECEIVE_MESSAGE ファンクション (28-10 ページ)	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。
NEXT_ITEM_TYPE ファンクション (28-12 ページ)	バッファにある次の項目のデータ型を戻します。
UNPACK_MESSAGE プロシージャ (28-13 ページ)	バッファにある次の項目にアクセスします。
REMOVE_PIPE ファンクション (28-14 ページ)	名前付きパイプを削除します。

表 28-2 DBMS_PIPE パッケージのサブプログラム

サブプログラム	説明
PURGE プロシージャ (28-15 ページ)	名前付きパイプの内容をパージします。
RESET_BUFFER プロシージャ (28-16 ページ)	ローカル・バッファの内容をパージします。
UNIQUE_SESSION_NAME ファンクシ ョン (28-16 ページ)	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 28-3 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、その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 28-4 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 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 28-4 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 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 28-5 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 28-6 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 28-7 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 28-8 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/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、またはパイプが満杯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。 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 28-9 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てから削除されます。

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 28-10 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 28-11 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 28-11 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 28-12 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 28-13 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 28-13 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 28-14 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 28-15 REMOVE_PIPE ファンクションのパラメータ

パラメータ	説明
<code>pipename</code>	削除するパイプ名。

戻り値

表 28-16 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 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 28-16 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。 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 28-17 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、最低使用頻度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パージされるとき、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パージしようとすると、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 28-18 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。

```

SVRMGR> variable rv number
SVRMGR> 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;
    sqlglm(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;
    sqlglm(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ード・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;
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 章の「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ープ・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、インクリメント・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、持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、次の作業が伴います。

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テストします。
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、このプロセスをサポートするために、「実行」という概念をサポートします。1 つの実行には、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。

一般的なセッションには、次の処理が含まれます。

- セッションでプロファイラ・データ収集を開始します。
- プロファイラ・データまたはコード・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。

注意： データ収集を停止すると、収集されたデータはフラッシュ・アウトされます。

アプリケーションの実行時、プロファイラ・データは、セッション期間中持続するメモリ・データ構造に収集されます。セッションの途中で `FLUSH_DATA` ファンクションをコールして、増分データを取得し、割り当てられたプロファイラ・データ構造のためにメモリを解放できます。

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するために提供されています。

関連項目： [「FLUSH_DATA ファンクション」](#) (29-5 ページ)。

他の方法として、集中的に管理された 1 つのスキーマに表を作成できます。プロファイラ・ユーザーは、パブリック・シノニムを作成し、表と順序についての `INSERT/SELECT` 権限を付与して、その表にアクセスできます。

注意： 収集したプロファイラ・データは、セッションの終了時に自動的にフラッシュされません。セッションの終了時にデータをフラッシュするためには、`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。

PL/SQL ユニットを最初に実行する場合など、一部の PL/SQL 操作には、実行する 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・アップを行う必要があります。ウォーム・アップは、プロファイラ・データを収集しないでアプリケーションを 1 回実行します。

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）を作成して分析することができます。

Oracle8i では、PL/SQL デモ用スクリプトに文脈依存のレポート・ライターのサンプルが準備されています。

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。

エラー・コード

ファンクションからの戻り値が 0 の場合は、正常終了を示します。0 以外の戻り値は、エラー状態を示します。潜在的なエラーの戻り値は、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。プロファイラ表は作成されていてアクセス可能か、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インプリメンテーション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、このエラーが発生します。回復する唯一の方法は、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;`

サブプログラムの要約

表 29-1 DBMS_PROFILER パッケージのサブプログラム

サブプログラム	説明
START_PROFILER ファンクション (29-4 ページ)	セッションでプロファイラ・データ収集を開始します。
STOP_PROFILER ファンクション (29-5 ページ)	セッションでプロファイラ・データ収集を停止します。
FLUSH_DATA ファンクション (29-5 ページ)	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。
GET_VERSION プロシージャ (29-6 ページ)	この API のバージョンを取得します。
INTERNAL_VERSION_CHECK ファンクション (29-6 ページ)	このバージョンの DBMS_PROFILER パッケージが、データベース内のインプリメンテーション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、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

```
DBMS_PROFILER.START_PROFILER (  
    run_comment IN VARCHAR2 := sysdate)  
RETURN BINARY_INTEGER;
```

パラメータ

表 29-2 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。

STOP_PROFILER ファンクション

このファンクションは、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、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。

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 29-3 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、データベース内のインプリメンテーション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、組み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 30-1 DBMS_RANDOM パッケージのサブプログラム

サブプログラム	説明
INITIALIZE プロシージャ (30-2 ページ)	シード値を使用して、パッケージを初期化します。
SEED プロシージャ (30-3 ページ)	シードをリセットします。
RANDOM ファンクション (30-3 ページ)	乱数を取得します。
TERMINATE プロシージャ (30-3 ページ)	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 30-2 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 30-3 SEED 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。

サブプログラムの要約

表 31-1 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
DIFFERENCES プロシージャ (31-2 ページ)	2 つの表の違いを判断します。
RECTIFY プロシージャ (31-5 ページ)	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 31-2 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。

表 31-2 DIFFERENCES プロシージャのパラメータ

パラメータ	説明
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。
where_clause	この制約を満たす行だけが比較のために選択されます。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。
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だスキーマの名前。
missing_rows_oname1	MISSING_ROWS_SITE にある表の名前で、この表には REFERENCE サイトの表にあって COMPARISON サイトの表にはない行、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_oname2	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_oname 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_oname 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、継続できなくなる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。

表 31-2 DIFFERENCES プロシージャのパラメータ

パラメータ	説明
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 31-3 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。
cannotbnull	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 31-4 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しないでください。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 31-4 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、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 31-5 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。

サブプログラムの要約

表 32-1 DBMS_REFRESH パッケージのサブプログラム

サブプログラム	説明
ADD プロシージャ (32-2 ページ)	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。
CHANGE プロシージャ (32-3 ページ)	スナップショット・グループのリフレッシュ間隔を変更します。
DESTROY プロシージャ (32-5 ページ)	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。
MAKE プロシージャ (32-6 ページ)	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。
REFRESH プロシージャ (32-8 ページ)	リフレッシュ・グループを手動でリフレッシュします。
SUBTRACT プロシージャ (32-9 ページ)	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： 追加情報は、『Oracle8i レプリケーション API リファレンス』の「ADD OBJECTS TO REFRESH GROUP」を参照してください。また、『Oracle8i レプリケーション・ガイド』マニュアルの「スナップショットの概念およびアーキテクチャ」も参照してください。

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 32-2 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、スナップショット・グループのリフレッシュ間隔を変更します。

関連項目： 追加情報は、『Oracle8i レプリケーション・ガイド』マニュアルの「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 32-3 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	グループにあるスナップショットの次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。
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見えることがあります。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。

表 32-3 CHANGE プロシージャのパラメータ

パラメータ	説明
purge_option	<p>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 = アグレッシブ・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は、アグレッシブ・パージに設定してキューを減らします。</p> <p>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アグレッシブ・パージに設定して PUSH を実行してください。</p>
parallelism	0 = シリアル伝播、 $n > 0 = n$ 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のために、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を Oracle が自動的に計算します。オラクル社カスタマ・サポート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： 追加情報は、『Oracle8i レプリケーション・ガイド』マニュアルの「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 32-4 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： 追加情報は、『Oracle8i レプリケーション API リファレンス』の「リフレッシュ・グループの作成」を参照してください。また、『Oracle8i レプリケーション・ガイド』マニュアルの「スナップショットの概念およびアーキテクチャ」も参照してください。

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 32-5 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、現行データベースにある必要があります。

表 32-5 MAKE プロシージャのパラメータ

パラメータ	説明
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 N 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、N + 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 値と共に使用されます。 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 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。
push_deferred_rpc	更新可能なスナップショットでのみ使用されます。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見えることがあります。
refresh_after_errors	更新可能なスナップショットでのみ使用されます。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 に設定します。

表 32-5 MAKE 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、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 = アグレッシブ・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、アグレッシブ・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アグレッシブ・パージに設定して PUSH を実行してください。
parallelism	0 = シリアル伝播、n > 0 = 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のために、同時に検査されるトランザクションの最大数。Oracle は、最適なパフォーマンスのためのデフォルト設定を自動的に計算します。オラクル社カスタマ・サポート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： 追加情報は、『Oracle8i レプリケーション・ガイド』マニュアルの「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 32-6 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： 追加情報は、『Oracle8i レプリケーション・ガイド』マニュアルの「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 32-7 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が、リスト内のすべてのスナップショットは、現行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 N 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、N + 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は、このパラメータを FALSE に設定します。

C

catproc.sql スクリプト, 1-2
CLOB データ型
 NCLOB, 17-3
CREATE PACKAGE BODY コマンド, 1-3
CREATE PACKAGE コマンド, 1-3

D

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_DDL パッケージ, 6-1
DBMS_DEBUG パッケージ, 7-1
DBMS_DEFER_QUERY パッケージ, 9-1
DBMS_DEFER_SYS パッケージ, 10-1
DBMS_DEFER パッケージ, 8-1
DBMS_DESCRIBE パッケージ, 11-1
DBMS_DISTRIBUTED_TRUST_ADMIN パッケージ,
 12-1
DBMS_HS_PASSTHROUGH パッケージ, 14-1
DBMS_HS パッケージ, 13-1
DBMS_IOT パッケージ, 15-1
DBMS_JOB パッケージ, 16-1
DBMS_LOB パッケージ, 17-1
DBMS_LOCK パッケージ, 18-1
DBMS_LOGMNR_D パッケージ, 20-1
DBMS_LOGMNR パッケージ, 19-1
DBMS_MVIEW パッケージ
 DBMS_SNAPSHOT パッケージ, 45-1
DBMS_OFFLINE_OG パッケージ, 21-1
DBMS_OFFLINE_SNAPSHOT パッケージ, 22-1
DBMS_OLAP パッケージ, 23-1

DBMS_ORACLE_TRACE_AGENT パッケージ, 24-1
DBMS_ORACLE_TRACE_USER パッケージ, 25-1
DBMS_OUTPUT パッケージ, 26-1
DBMS_PCLXUTIL パッケージ, 27-1
DBMS_PIPE パッケージ, 28-1
DBMS_PROFILER パッケージ, 29-1
DBMS_RANDOM パッケージ, 30-1
DBMS_RECTIFIER_DIFF パッケージ, 31-1
DBMS_REFRESH パッケージ, 32-1
DBMS_REPAIR パッケージ, 33-1
DBMS_REPCAT_ADMIN パッケージ, 35-1
DBMS_REPCAT_INSTANTIATE パッケージ, 36-1
DBMS_REPCAT_RGT パッケージ, 37-1
DBMS_REPCAT パッケージ, 34-1
DBMS_REPUTIL パッケージ, 38-1
DBMS_RESOURCE_MANAGER_PRIVS パッケージ,
 40-1
DBMS_RESOURCE_MANAGER パッケージ, 39-1
DBMS_RLS パッケージ, 41-1
DBMS_ROWID パッケージ, 42-1
DBMS_SESSION パッケージ, 43-1
DBMS_SHARED_POOL パッケージ, 44-1
DBMS_SNAPSHOT パッケージ
 DBMS_MVIEW パッケージ, 45-1
DBMS_SPACE_ADMIN パッケージ, 47-1
DBMS_SPACE パッケージ, 46-1
DBMS_SQL パッケージ
 エラーの位置, 48-31
DBMS_STATS パッケージ, 49-1
DBMS_TRACE パッケージ, 50-1
DBMS_TRANSACTION パッケージ, 51-1
DBMS_TTS パッケージ, 52-1
DBMS_UTILITY パッケージ, 53-1
DEBUG_EXPTOC パッケージ, 54-1
DefDefaultDest 表

宛先の削除, 10-4
宛先の追加, 10-3
DefError 表
トランザクションの削除, 10-5
DESC_TAB データ型, 48-29

L

LOB
DBMS_LOB パッケージ, 17-1

O

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, 55-1

P

PL/SQL
データ型, 11-6
数値コード, 11-8

R

RepCatLog ビュー
削除, 34-60
RepColumn_Group 表
更新, 34-23
RepGroup ビュー
更新, 34-25
RepObject 表
更新, 34-27
RepPriority_Group 表
更新, 34-24
RepResolution_Statistics 表
削除, 34-61
RepResolution 表
更新, 34-28
RepSite ビュー
更新, 34-26
ROWID のデータ型
拡張形式, 42-12
DBMS_ROWID パッケージ, 42-1

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
SQL*Plus
順序番号の作成, 1-5
SQL 文
32KB を超える, 48-12

T

TimeScale パッケージ, 1-16
TimeSeries パッケージ, 1-17
TSTools パッケージ, 1-20

U

UTL_COLL パッケージ, 56-1
UTL_FILE パッケージ, 57-1
UTL_HTTP パッケージ, 58-1
UTL_PG パッケージ, 1-21
UTL_RAW パッケージ, 59-1
UTL_REF パッケージ, 60-1

V

Vir_Pkg パッケージ, 1-22

あ

アドバンスド・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

インポート
オブジェクト・グループ
オフライン・インスタンスেশション, 21-3,
21-5
状態チェック, 34-67
スナップショット
オフライン・インスタンスেশション, 22-2,
22-3

え

エラー

- DBMS_ALERT パッケージが戻すエラー , 15-3
- 動的 SQL 内での位置 , 48-31

お

オフライン・インスタンスーション

- スナップショット , 22-2 , 22-3
- レプリケート・オブジェクト・グループ , 21-2 , 21-3 , 21-4 , 21-5 , 21-6

か

カーソル

- DBMS_SQL パッケージ , 48-4

各国語サポート

- NCLOB , 17-3

カレンダー・パッケージ , 1-13

き

キャラクタ・セット

- ANY_CS , 17-3

キューイング

- DBMS_AQADM パッケージ , 5-1

休止中

- レプリケート・スキーマ , 34-72

競合

解消

- 追加方法 , 34-10
- 統計 , 34-22 , 34-64

こ

コレクション

- 表項目 , 48-14

さ

サイト優先順位

- 変更 , 34-19

サイト優先順位グループ

- 削除 , 34-50
- 作成
 - 構文 , 34-41

- メンバーの削除 , 34-50

- メンバーの追加 , 34-9

削除

- RepCatLog 表 , 34-60
- サイト優先順位グループ , 34-50
- スナップショット・サイト , 34-51
- 統計 , 34-61
- マスター・サイト , 34-66
- 優先順位グループ , 34-47
- 列グループ
 - 構文 , 34-42
- レプリケート・オブジェクト
 - グループ , 34-44
 - スナップショット・サイト , 34-52
 - マスター・サイト , 34-46

作成

- サイト優先順位グループ
 - 構文 , 34-41
- スナップショット・サイト
 - 構文 , 34-35
- パッケージ , 1-3
- 優先順位グループ , 34-39
- リフレッシュ・グループ , 32-6
- 列グループ
 - 構文 , 34-38 , 34-59
- レプリケート・オブジェクト
 - 構文 , 34-32
 - サポートの生成 , 34-56
 - スナップショット・サイト , 34-36
- レプリケート・オブジェクト・グループ
 - 構文 , 34-31

し

使用禁止

- 伝播 , 10-17

実行フロー

- 動的 SQL , 48-4

状態

- 伝播 , 10-6

ジョブ

- キュー
 - ジョブの削除 , 10-19

す

ストアド・アウトライン

- OUTLN_PKG パッケージ, 55-1
- スナップショット
 - オフライン・インスタンスーション, 22-2, 22-3
 - リフレッシュ, 45-7, 45-9, 45-10
- スナップショット・サイト
 - 削除, 34-51
 - 作成
 - 構文, 34-35
 - マスターの変更, 34-73
 - マスターへの変更内容の伝播, 10-15
 - リフレッシュ
 - 構文, 34-62
- スナップショット・ログ
 - マスター表
 - 削除, 45-4, 45-5

せ

- 生成
 - スナップショット・サポート, 34-57

ち

- 遅延トランザクション
 - DefDefaultDest 表
 - 宛先の削除, 10-4
 - 宛先の追加, 10-3
 - 開始, 8-5
 - キューからの削除, 10-5
 - 再実行, 10-8
 - スケジュールの実行, 10-15
 - 遅延リモート・プロシージャ・コール (RPC)
 - 作成, 8-2
 - 即時実行, 10-11
 - 引数, 8-4
 - 引数型, 9-3
 - 引数値, 9-6

- 違い
 - 複数の表, 31-2
 - 調整, 31-5

- 調整
 - 表, 31-5

て

- データ型
 - DBMS_DESCRIBE, 11-4

- DESC_TAB, 48-29
- NCLOB, 17-3
- PL/SQL
 - 数値コード, 11-8
- ROWID, 42-1
- データ定義言語 (DDL)
 - 非同期の提供, 34-55
- 伝播
 - 使用禁止, 10-17
 - 状態, 10-6
 - 変更内容
 - 伝播方法の変更, 34-14

と

- 統計
 - 収集, 34-64
- 動的 SQL
 - DBMS_SQL ファンクション、使用方法, 48-2
 - エラー、位置, 48-31
 - 実行フロー, 48-4
 - 無名ブロック, 48-2

は

- 配列
 - BIND_ARRAY プロシージャ, 48-6
 - DBMS_SQL を使用した一括 DML, 48-14
- はじめに
 - 表記規則の例の一覧, xxxiv
- 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

ひ

- 比較
 - 表, 31-2
- 非同期
 - DDL, 34-55
- 表
 - 調整, 31-5
 - 配列としての表項目, 48-14
 - 比較, 31-2

ふ

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, 41-1
プランの安定性, 55-1

へ

変更

伝播方法, 34-14, 34-21
優先順位レベル, 34-17
レプリケート・オブジェクト, 34-15
変更内容の伝播
伝播方法の変更, 34-21

ま

マスター・サイト
削除, 34-66
作成, 34-7
変更内容の伝播, 10-15
マスター定義サイト
再配置, 34-65

む

無名 PL/SQL ブロック
動的 SQL, 48-2

ゆ

優先順位グループ
サイト優先順位グループ
メンバーの追加, 34-9
削除, 34-47
作成, 34-39
メンバーの削除, 34-47, 34-48
メンバーの追加, 34-8
メンバーの変更
値, 34-18
優先順位, 34-17

り

リフレッシュ
スナップショット, 45-7, 45-9, 45-10
スナップショット・サイト

構文, 34-62
リフレッシュ・グループ
削除, 32-5
新規作成, 32-6
メンバーを削除, 32-9
メンバーを追加, 32-2
リフレッシュ
手動, 32-8
リフレッシュ間隔
変更, 32-3

れ

列グループ
削除
構文, 34-42
作成
構文, 34-38, 34-59
メンバーの削除
構文, 34-43
メンバーの追加
構文, 34-5
レプリケーション・アクティビティの再開, 34-68
レプリケート・オブジェクト
DROP_MASTER_REPOBJECT, 34-46
削除
スナップショット・サイト, 34-52
作成
スナップショット・サイト, 34-36
マスター・サイト, 34-32
サポートの生成, 34-56
変更, 34-15
マスター・グループの作成, 34-31
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, 21-2, 21-3, 21-4, 21-5, 21-6
削除, 34-44

