Oracle® GoldenGate

Windows and UNIX リファレンス・ガイド

11*g* リリース 2 パッチ・セット 1 (11.2.1.0.1)

B69763-01 (原本部品番号 : E29399-01)

2012 年 11 月

Oracle GoldenGate Windows and UNIX リファレンス・ガイド 11g リリース 2 パッチ・セット 1 (11.2.1.0.1)

B69763-01 (原本部品番号 : E29399-01)

目次

はじめに

Oracle GoldenGate のドキュメントについて

完全な Oracle GoldenGate ドキュメント・セットは、次のマニュアルで構成されています。

HP NonStop プラットフォーム

- 『Oracle GoldenGate *HP NonStop 管理者ガイド*』: NonStop プラットフォームで使用する Oracle GoldenGate レプリケーション・ソリューションの計画、構成および実装方法が記載されています。
- 『Oracle GoldenGate *HP NonStop リファレンス・ガイド*』: NonStop プラットフォームで使用する Oracle GoldenGate のパラメータ、コマンドおよびファンクションに関する詳細情報が記載されています。

Windows、UNIX および Linux プラットフォーム

- *インストレーションおよびセットアップ・ガイド*: Oracle GoldenGate によってサポートされているデータベースごとに、インストレーションおよびセットアップ・ガイドが提供されています。Oracle GoldenGate レプリケーション・ソリューションのインストールのためのシステム要件、インストール前およびインストール後の手順、インストール方法、システム固有のその他の情報が記載されています。
- 『*Oracle GoldenGate Windows and UNIX 管理者ガイド*』: Windows および UNIX プラットフォームで使用する Oracle GoldenGate レプリケーション・ソリューションの計画、構成および実装方法が記載されています。
- 『*Oracle GoldenGate Windows and UNIX リファレンス・ガイド*』: Windows および UNIX プラットフォームで使用する Oracle GoldenGate のパラメータ、コマンドおよびファンクションに関する詳細情報が記載されています。
- 『Oracle GoldenGate *Windows and UNIX トラブルシューティングおよびチューニング・ガイド*』: Oracle GoldenGate レプリケーション・ソリューションのパフォーマンスを改善するための推奨事項と一般的な問題の解決策が記載されています。

その他の Oracle GoldenGate 製品

- 『*Oracle GoldenGate Monitor 管理者ガイド*』: Oracle GoldenGate レプリケーション・コンポーネントを監視するための *Oracle GoldenGate Monitor* のインストール、実行および管理方法が記載されています。
- 『*Oracle GoldenGate Director 管理者ガイド*』: Oracle GoldenGate レプリケーション・コンポーネントの構成、管理、監視およびレポート作成に使用する Oracle GoldenGate Director のインストール、実行および管理方法が記載されています。
- 『*Oracle GoldenGate Veridata 管理者ガイド*』: Oracle GoldenGate Veridata データ比較ソリューションのインストール、実行および管理方法が記載されています。
- 『*Oracle GoldenGate for Java 管理者ガイド*』: Oracle GoldenGate トレイルへの JMS メッセージを取得し、取得したデータをメッセージング・システムやカスタム API に配信する Oracle GoldenGate for Java のインストール、構成および実行方法が記載されています。
- 『*Oracle GoldenGate for Flat File 管理者ガイド*』: Oracle GoldenGate によって取得されたデータを、ETL、固有またはレガシー・アプリケーションへのバッチ入力としてフォーマットする Oracle GoldenGate for Flat File のインストール、構成および実行方法が記載されています。

このマニュアルの表記規則

このマニュアルでは、次の表記規則を使用します。

- パラメータとコマンド引数は、次の例のように大文字で表記されます。
  ```
  CHECKPARAMS
  ```
- ファイル名、表名およびその他の名前は、関連するオペレーティング・システムまたはソフトウェア・アプリケーションで大 / 小文字が区別される場合を除き、次の例のように小文字で表記されます。
  ```
  account_tab
  GLOBALS
  ```
- 変数は、次の例のように山カッコ (< >) 内に表記されます。
  ```
  <group name>
  ```
- 複数の相互排他的な引数の 1 つをを選択する必要がある場合は、次の例のように、選択肢が中カッコで囲まれ、パイプ記号で区切られています。
  ```
  VIEW PARAMS {MGR | <group> | <file name>}
  ```
- オプションの引数は、次の例のように大カッコで囲まれています。
  ```
  CLEANUP EXTRACT <group name> [, SAVE <count>]
  ```
- オプションの引数が多数ある場合は、[<option>] のようなプレースホルダが使用され、次の例のようにオプションが個別にリストおよび説明されています。
  ```
  TRANLOGOPTIONS [<option>]
  ```
- 1 つの引数が複数回渡される場合は、次の例のように省略記号 (...) が使用されています。
  ```
  PARAMS ([<requirement rule>] <param spec> [, <param spec>] [, ...])
  ```
- アンパサンド (&) は、Oracle GoldenGate パラメータ・ファイルで継続文字として使用されています。複数行にわたるパラメータ文の各行の末尾に配置する必要があります。このドキュメントの大半の例では、アンパサンドは適切な位置に配置されていますが、発行フォーマットのスペースの制約により、複数行にわたる文の一部では省略されていることがあります。

Oracle GoldenGate のその他のヘルプ情報の取得

Oracle GoldenGate ドキュメントに加えて、次の方法で Oracle GoldenGate のヘルプ情報を取得できます。

Oracle GoldenGate インタフェースでのヘルプ情報の取得

GGSCI および Oracle GoldenGate Director アプリケーションは、ともにオンライン・ヘルプを提供しています。

GGSCI コマンド

Oracle GoldenGate コマンドのヘルプを取得するには、GGSCI で HELP コマンドを使用します。コマンド・カテゴリの概要を取得するには、オプションを指定せずに HELP コマンドを発行します。特定のコ

マンドのヘルプを取得するには、コマンド名を入力として HELP コマンドを発行します。

```
HELP <command name>
```

例 :

```
HELP ADD EXTRACT
```

ヘルプ・ファイルにコマンドの構文と説明が表示されます。

Oracle GoldenGate Director および Oracle GoldenGate Monitor

Oracle GoldenGate グラフィカル・クライアント・インタフェースのヘルプを取得するには、各アプリケーション内の**「Help」**メニューを使用します。

質問と問題に関するヘルプ情報の取得

トラブルシューティングの詳細は、『*Oracle GoldenGate Windows and UNIX トラブルシューティングおよびチューニング・ガイド*』を参照してください。補足情報は、http://support.oracle.com のナレッジ・ベースで取得できます。質問に対する回答が見つけられない場合は、サポート・サイトからサービス・リクエストをオープンできます。

第 1 章
Oracle GoldenGate GGSCI コマンド

Oracle GoldenGate ソフトウェア・コマンド・インタフェース (GGSCI) は、ユーザーと Oracle GoldenGate 機能コンポーネント間のコマンド・インタフェースです。

コマンドの概要

GGSCI コマンドの概要は次のとおりです。

コマンド・グループ	目的
Manager コマンド	Manager プロセスを起動および管理します。
Extract コマンド	Extract グループを作成および管理します。
Replicat コマンド	Replicat グループを作成および管理します。
ER コマンド	複数の Extract および Replicat グループを 1 つのユニットとして制御します。
トレイル・コマンド	トレイルを Extract グループに関連付け、ファイル管理パラメータを提供します。
パラメータ・コマンド	エディタを実行し、パラメータを定義または変更します。
データベース・コマンド	データベース関連コマンドを発行します。
Trandata コマンド	Oracle GoldenGate が UPDATE 操作をレプリケートするために必要な補足情報をロギングするようにデータベースを構成します。
チェックポイント表コマンド	Oracle GoldenGate チェックポイント表を作成および管理します。
Oracle トレース表コマンド	双方向構成でのデータのループを防止するためのトレース表を作成および管理します。
DDL コマンド	DDL 同期に関連するコマンドです。
その他のコマンド	その他の機能を制御します。

Manager コマンド

Manager コマンドでは、Manager プロセスを制御します。Manager は Oracle GoldenGate の親プロセスで、自身のプロセスおよびファイル、リソース、ユーザー・インタフェース、しきい値およびエラーのレポートを管理します。

コマンドの概要

Manager コマンドの概要は次のとおりです。

INFO MANAGER

SEND MANAGER

START MANAGER

STATUS MANAGER

STOP MANAGER

INFO MANAGER

INFO MANAGER では、Manager プロセスが実行中かどうかを確認します。Manager が実行中の場合は、ポート番号が表示されます。このコマンドは STATUS MANAGER の別名です。

構文

```
INFO MANAGER
```

SEND MANAGER

SEND MANAGER では、アクティブな Manager プロセスのステータスを取得するか、Manager パラメータ・ファイルに構成されている動的ポート情報を取得します。

構文

```
SEND MANAGER
[CHILDSTATUS [DEBUG]]
[GETPORTINFO [DETAIL]
[GETPURGEOLDEXTRACTS]
```

引数	説明
`CHILDSTATUS [DEBUG]`	Manager によって起動されたプロセスに関するステータス情報を取得します。DEBUG は、プロセスに割り当てられたポート番号を返します。
`GETPORTINFO [DETAIL]`	デフォルトでは、プロセスに割り当てられたポート、および対応するプロセス ID の現在のリストを取得します。DETAIL は補足情報を提供します。
`GETPURGEOLDEXTRACTS`	Manager パラメータ・ファイルの PURGEOLDEXTRACTS パラメータで設定されているトレイル・メンテナンス・ルールに関する情報を表示します。PURGEOLDEXTRACTS の詳細は、293 ページを参照してください。

例 1 SEND MANAGER CHILDSTATUS DEBUG は、次のような子プロセス・ステータスを返します。基本の CHILDSTATUS オプションも同様の表示を返しますが、Port 列は含まれません。

```
ID  Group   Process  Retry  Retry Time  Start Time           Port
1   ORAEXT  2400     0      None        2011/01/21 21:08:32  7840
2   ORAEXT  2245     0      None        2011/01/23 21:08:33  7842
```

例 2 SEND MANAGER GETPORTINFO DETAIL は、次のような動的ポート・リストを返します。

```
Entry  Port   Error   Process   Assigned             Program
0      8000   0       2387      2011-01-01 10:30:23
1      8001   0
2      8002   0
```

例 3 SEND MANAGER GETPURGEOLDEXTRACTS は、次のような情報を出力します。

```
PurgeOldExtracts Rules
Fileset                       MinHours MaxHours MinFiles MaxFiles UseCP
S:\GGS\DIRDAT\EXTTRAIL\P4\*          0        0        1        0   Y
S:\GGS\DIRDAT\EXTTRAIL\P2\*          0        0        1        0   Y
S:\GGS\DIRDAT\EXTTRAIL\P1\*          0        0        1        0   Y
S:\GGS\DIRDAT\REPTRAIL\P4\*          0        0        1        0   Y
S:\GGS\DIRDAT\REPTRAIL\P2\*          0        0        1        0   Y
S:\GGS\DIRDAT\REPTRAIL\P1\*          0        0        1        0   Y
OK
Extract Trails
Filename                         Oldest_Chkpt_Seqno  IsTable  IsVamTwoPhaseCommit
S:\GGS\8020\DIRDAT\RT                             3        0        0
S:\GGS\8020\DIRDAT\REPTRAIL\P1\RT                13        0        0
S:\GGS\8020\DIRDAT\REPTRAIL\P2\RT                13        0        0
S:\GGS\8020\DIRDAT\REPTRAIL\P4\RT                13        0        0
S:\GGS\8020\DIRDAT\EXTTRAIL\P1\ET                14        0        0
S:\GGS\8020\DIRDAT\EXTTRAIL\P2\ET                14        0        0
S:\GGS\8020\DIRDAT\EXTTRAIL\P4\ET                14        0        0
```

START MANAGER

START MANAGER では、Manager プロセスを起動します。このコマンドは、非クラスタ環境で使用します。Windows クラスタ環境では、Cluster Administrator から Manager を停止する必要があります。

構文

```
START MANAGER
```

STATUS MANAGER

STATUS MANAGER では、Manager プロセスが実行中かどうかを確認します。Manager が実行中の場合は、ポート番号が表示されます。

構文

```
STATUS MANAGER
```

STOP MANAGER

STOP MANAGER では、Manager を停止します。このコマンドは、非クラスタ環境で使用します。Windows クラスタ環境では、Cluster Administrator から Manager を停止する必要があります。

構文

```
STOP MANAGER [!]
```

引数	説明
!	(感嘆符)Manager をシャットダウンするかどうかを確認するプロンプトをバイパスします。

Extract コマンド

Extract コマンドでは、Extract グループを作成および管理します。Extract プロセスは、構成パラメータに応じて完全なデータ・レコードまたはトランザクション・データの変更を取得した後、ターゲット表に適用されるかロード・ユーティリティなどの別のプロセスによって処理されるデータをターゲット・システムに送信します。

コマンドの概要

ADD EXTRACT

ALTER EXTRACT

CLEANUP EXTRACT

DELETE EXTRACT

INFO EXTRACT

KILL EXTRACT

LAG EXTRACT

REGISTER EXTRACT

SEND EXTRACT

START EXTRACT

STATS EXTRACT

STATUS EXTRACT

STOP EXTRACT

UNREGISTER EXTRACT

ADD EXTRACT

ADD EXTRACT では、Extract グループを作成します。SOURCEISTABLE タスクまたは別名 Extract が指定されている場合を除き、ADD EXTRACT は実行間の処理の継続性を維持するためにチェックポイントを作成

します。Extract グループを作成する前に『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を確認してください。

コマンドの制限

Oracle GoldenGate は、Oracle GoldenGate Manager インスタンス当たり、最大で 5,000 の同時 Extract および Replicat グループをサポートします。サポートされているレベルで、INFO、STATUS などの GGSCI コマンドですべてのグループを完全に制御および表示できます。Oracle GoldenGate では、環境を効果的に管理するために、Extract および Replicat グループの数 (合計) をデフォルト・レベルの 300 以下に保つことが推奨されます。

このコマンドでは、DESC オプションに入力するテキストを含め、すべてのキーワードおよび入力のサイズは、500 バイトを超えることはできません。

構文　標準、パッシブまたはデータ・ポンプ Extract の場合

```
ADD EXTRACT <group name>
{, SOURCEISTABLE |
    , TRANLOG [<bsds name>] |
    , INTEGRATED TRANLOG |
    , VAM |
    , EXTFILESOURCE <file name> |
    , EXTTRAILSOURCE <trail name> |
    , VAMTRAILSOURCE <VAM trail name>}
{, BEGIN {NOW | yyyy-mm-dd [:hh:mi:[ss[.cccccc]]]} |
    , EXTSEQNO <seqno>, EXTRBA <relative byte address> |
    , LOGNUM <log number>, LOGPOS <byte offset> |
    , EOF |
    , LSN <value> |
    , EXTRBA <relative byte address> |
    , EOF | LSN <value> |
    , PAGE <data page>, ROW <row> |
    }
[, THREADS <n>]
[, PASSIVE]
[, PARAMS <parameter file>]
[, REPORT <report file>]
[, DESC "<description>"]
```

構文　別名 Extract の場合

```
ADD EXTRACT <group name>
, RMTHOST {<host name> | <IP address>}
, MGRPORT <port>
[, RMTNAME <name>]
[, DESC "<description>"]
```

引数	説明
<group name>	Extract グループ名。グループの命名規則については、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』を参照してください。

引数	説明
SOURCEISTABLE	Oracle GoldenGate ダイレクト・ロード方法または SQL*Loader へのダイレクト・バルク・ロード方法を使用して、データベースから初期ロードのためにレコード全体を抽出する Extract タスクを作成します。SOURCEISTABLE を指定しない場合、ADD EXTRACT はオンライン変更同期プロセスを作成します。この場合は、他のデータ・ソース・オプションを 1 つ指定する必要があります。SOURCEISTABLE を使用するときは、サービス・オプションを指定しないでください。タスク・パラメータは、パラメータ・ファイルに指定する必要があります。 初期ロード方法の詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。
TRANLOG [<bsds name>]	データ・ソースとしてトランザクション・ログを指定します。このオプションは、Teradata 以外のすべてのデータベースで使用します。TRANLOG を指定するときは、BEGIN オプションを指定する必要があります。 (z/OS 上の DB2) z/OS システム上の DB2 では、<bsds name> オプションを使用して、トランザクション・ログのブートストラップ・データ・セット (BSDS) ファイル名を指定します。指定する BSDS 名が、*Extract プロセスの接続先の DB2 インスタンス*のものであることを確認してください。Oracle GoldenGate では、BSDS の指定の妥当性はチェックされません。 (Oracle) Oracle Standard または Enterprise Edition 11.2.0.2 以降では、このモードは*クラシック・キャプチャ・モード*と呼ばれます。Extract は Oracle REDO ログを直接読み取ります。代替構成については、INTEGRATED TRANLOG も参照してください。
INTEGRATED TRANLOG	統合キャプチャ・モードで、この Extract を追加します。このモードでは、Extract は、論理変更レコード (LCR) を Extract に直接渡すデータベース・ログマイニング・サーバーと統合します。Extract は REDO ログを読み取りません。INTEGRATED TRANLOG を使用する前に、DBLOGIN または MININGDBLOGIN コマンドを使用して、ソース・データベースまたはダウンストリーム・マイニング・データベースにログインしてから、REGISTER EXTRACT コマンドを使用して、Extract をそのデータベースに登録します。統合キャプチャの詳細は、『Oracle GoldenGate *Oracle インストレーションおよびセットアップ・ガイド*』を参照してください。
VAM	(Teradata) *Vendor Access Module* (VAM) と呼ばれる Extract API を Teradata Access Module (TAM) とのインタフェースとして指定します。このオプションは、Teradata データベースで使用します。
EXTFILESOURCE <file name>	データ・ソースとして抽出ファイルを指定します。このオプションは、プライマリ Extract グループとターゲット・システム間の仲介の役割を果たすセカンダリ Extract グループ (データ・ポンプ) とともに使用します。 <file name> には、ファイルの相対パス名または完全修飾パス名を指定します (例 : dirdat\extfile または c:\ggs\dirdat\extfile)。

引数	説明
EXTTRAILSOURCE <trail name>	データ・ソースとしてトレイルを指定します。このオプションは、プライマリ Extract グループとターゲット・システム間の仲介の役割を果たすセカンダリ Extract グループ (データ・ポンプ) とともに使用します。 <trail name> には、トレイルの相対パス名または完全修飾パス名を指定します (例 : dirdat\aa または c:\ggs\dirdat\aa)。
VAMTRAILSOURCE <VAM trail name>	(Teradata)VAM トレイルを指定します。このオプションは、Teradata 最大保護モードを使用しているときに使用します。 <VAM trail name> には、プライマリ Extract グループの書込み先 VAM トレイルの相対パス名または完全修飾パス名を指定します。VAM トレイルの読取り、およびターゲット・システムへのデータの送信には、VAM ソート Extract グループを使用します。
BEGIN {NOW | yyyy-mm-dd [:hh:mi:[ss[.cccccc]]]}	データ・ソース内の処理開始位置のタイムスタンプを指定します。 次の値が有効です。 ◆ NOW ◆ 次のフォーマットの日付と時刻 : yyyy-mm-dd[:hh:mi:[ss[.cccccc]]] **NOW の意味 :** DB2 LUW 以外のすべてのデータベースの場合、NOW では ADD EXTRACT コマンドの発行時刻を指定します。 DB2 LUW の場合 NOW では START EXTRACT を有効にする時刻を指定します。この日時に*ほぼ*一致する最初のレコードが開始位置になります。これは、DB2 LUW ではトランザクションのコミットおよびアボート・レコードがタイムスタンプを含む唯一のログ・レコードで、これらのタイムスタンプとの比較しか開始位置を計算する方法がないためです。これは、Oracle GoldenGate が使用する API の制限によるものです。 データ・ポンプ Extract では、ADD EXTRACT 文より前にトレイルに取得されたデータをバイパスするときを除き、NOW を使用しないでください。 **SQL Server のトランザクション・ログのタイムスタンプによる開始位置の決定** 時刻で開始位置を決定する場合、SQL Server の次の制約によって影響を受けます。 ◆ SQL Server トランザクション・ログに記録されているタイムスタンプは、3.3333 マイクロ秒 (ms) の粒度を使用します。このレベルの粒度では、2 つのトランザクションが同じ 3.333 ミリ秒の時間間隔で開始する場合、2 つのトランザクションの時間差に基づいて開始位置を決定できないことがあります。 ◆ タイムスタンプは、すべての SQL Server ログ・レコードではなく、トランザクションを開始してコミットしたレコード、およびデータが含まれないその他の一部のレコードのみに記録されます。

引数	説明

	◆ SQL Server のタイムスタンプは、システム・クロックではなく、使用中の各プロセッサ固有の内部クロックに基づいています。このクロックは 1 秒間に数回更新されますが、更新から更新までの間にシステム・クロックとの同期が外れることがあります。その場合、時刻で決定する開始位置の正確さがさらに低下します。
	◆ ログ・バックアップ・ファイルに記録されるタイムスタンプは、バックアップ内に記録された時刻と正確に一致しない可能性があります (ただし、不正確さは 1 秒未満です)。
	LSN を使用した位置決めのほうが正確です。<LSN <value> を参照してください。
	
	Sybase のトランザクション・ログのタイムスタンプによる開始位置の決定
	Sybase のみ、BEGIN および COMMIT レコードにタイムスタンプが記録されます。指定した実際のタイムスタンプに関係なく、開始位置は、指定したタイムスタンプに最も近い位置、またはその位置から開始するトランザクションの最初のレコードになります。Extract レポートには、次の位置が表示されます。
	Positioning To: 指定した開始時刻です。たとえば、次のようになります。
	`Positioning to begin time Jan 1, 2011 12:13:33 PM.`
	Positioned To: 指定したタイムスタンプが、BEGIN または COMMIT レコードを含むトランザクション・ログのタイムスタンプ以前である場合、次の例のように "Positioned To Page" が表示されます。
	`2011-01-01 12:13:39 INFO OGG-01516 Positioned to Page #: 0004460243 Row #: 00111, Jan 1, 2011 12:13:38 PM.`
	First Record Position: 次の例のように、**Positioned To** の位置またはそれ以降の位置で見つかった最初の有効なレコードの位置です。
	`2011-01-01 12:13:39 INFO OGG-01517 Position of first record processed Page #: 0004460243 Row #: 00111, Jan 1, 2011 12:13:38 PM.`

`EXTSEQNO <seqno>, EXTRBA <relative byte address>`	Oracle と NonStop SQL/MX のプライマリ Extract、およびデータ・ポンプ Extract に有効です。次のいずれかを指定します。 ◆ Oracle REDO ログの順序番号と、そのログ内のデータ取得開始位置の RBA。 ◆ NonStop SQL/MX TMF オーディット・トレイルの順序番号と、そのファイル内のデータ取得開始位置の相対バイト・アドレス。この 2 つの要素によって、TMF マスター・オーディット・トレイル (MAT) の位置を指定します。 ◆ (データ・ポンプの場合) トレイル内のデータ取得開始ファイル。順序番号を指定しますが、埋め込みのために 0 を使用しないでください。たとえば、トレイル・ファイルが c:\ggs\dirdat\aa000026 の場合は、EXTSEQNO 26 と指定します。このオプションを使用しない場合、処理はデフォルトでトレイルの最初から開始されます。 このオプションを使用する前に Oracle サポートに連絡してください。詳細は、http://support.oracle.com を参照してください。

引数	説明
`EXTRBA <relative byte address>`	z/OS 上の DB2 に有効です。トランザクション・ログ内のデータ取得開始位置の相対バイト・アドレスを指定します。
`EOF`	SQL Server に有効です。ログ・ファイルの末尾 (次のレコードの書込み位置) から処理を開始するように構成します。アクティブなトランザクションは取得されません。

`LSN <value>`	SQL Server に有効です。SQL Server トランザクション・ログ内のデータ取得開始位置の LSN を指定します。ログ・バックアップまたはオンライン・ログ内に存在する LSN を指定する必要があります。このオプションの別名は EXTLSN です。
	
	SQL Server では、データベースが返す方法に応じて、LSN は次のいずれかで構成されます。
	1. 0X00000d7e:0000036b:01bd のように、接頭辞 0X と先行ゼロを付けたコロン区切り 16 進数文字列 (8:8:4)
	2. 000003454:0000000875:00445 のように、先行ゼロを付けたコロン区切り 10 進数文字列 (10:10:5)
	3. 0Xd7e:36b:1bd のように、接頭辞 0X を付けた先行ゼロなしのコロン区切り 16 進数文字列
	4. 3454:875:445 のように、先行ゼロなしのコロン区切り 10 進数文字列
	5. 3454000000087500445 のような 10 進数文字列
	上記で、最初の値は仮想ログ・ファイル番号、2 番目の値は仮想ログ内のセグメント番号、3 番目の値はエントリ番号です。
	次のような問合せを使用すると、名前付きトランザクションの LSN を検索できます。
	`select [Current LSN], [Transaction Name], [Begin Time]`
	`  from fn_dblog(null, null)`
	` where Operation = 'LOP_BEGIN_XACT'`
	`   and [Begin Time] >= <time>`
	
	特定のトランザクションの開始時刻を確認した後、関連する LSN を検索し、開始時刻が同じ 2 つのトランザクションの間に開始位置を指定できます。

`EOF | LSN <value>`	DB2 LUW に有効です。Extract 起動時のトランザクション・ログ内のデータ取得開始位置を指定します。 ◆ EOF では、ログ・ファイル内のアクティブ LSN から処理を開始するように構成します。アクティブ LSN は、ログ・ファイルの末尾位置 (次のレコードの書込み位置) です。アクティブなトランザクションは取得されません。

引数	説明
	◆ LSN <value> では、指定した LSN に有効なログ・レコードが存在する場合に、そこから処理を開始するように構成します。指定した LSN が存在しない場合、Extract は異常終了します。Extract は指定した LSN に配置されますが、Extract がその LSN を最初に処理するとはかぎらないことに注意してください。これは、ログ・ファイル内に、DB2 内部ログ・レコードなど Extract が無視する多数のタイプのレコードが存在するためです。Extract は、実際に最初に処理した LSN を Extract レポート・ファイルにレポートします。
`PAGE <data page>, ROW <row>`	Sybase に有効です。Sybase トランザクション・ログ内の開始位置を定義するデータ・ページと行を指定します。開始位置は、指定した PAGE および ROW に最も近い位置、またはその位置から開始するトランザクションの最初のレコードである必要があるため、Extract レポートには、次の位置が表示されます。 ◆ **Positioning To**: PAGE および ROW で指定されたレコードの位置。 ◆ **Positioned To**: Positioning To の位置またはそれ以降の位置で見つかった最初の BEGIN レコードの位置。 ◆ **First Record Position**: Positioning To の位置またはそれ以降の位置で見つかった最初の有効なレコードの位置。
`PARAMS <parameter file>`	Extract パラメータ・ファイルの保管場所の完全パス名を、Oracle GoldenGate ディレクトリ内のデフォルトの dirprm 以外に指定します。
`REPORT <report file>`	Extract レポート・ファイルの保管場所の完全パス名を、Oracle GoldenGate ディレクトリ内のデフォルトの dirrpt 以外に指定します。
`THREADS <n>`	REDO ログを読み取るために Extract が保持するプロデューサ・スレッドの数を指定します。 ◆ 統合キャプチャの場合、この値は 1 にする必要があります。これは、Oracle ログマイニング・サーバーが送信する論理変更レコードを解析およびフィルタする Extract 構造に、プロデューサ・スレッドを追加し、そのデータを Extract メモリー・キャッシュに送信します。コンシューマ・スレッドは、コミット済トランザクションを取得するためにメモリー・キューを読み取り、フォーマット、フェッチ、チェックポイント、メタデータ、およびトレイルへの書込みを処理します。これらの職責を、メモリー記憶域およびキューイングを持つ 2 つの非同期スレッドに分離することにより、Extract のパフォーマンスは向上します。THREADOPTIONS パラメータの OUTQUEUESIZE オプションを使用して、このキューのサイズを設定できます。 ◆ クラシック・キャプチャ・モードの場合、Oracle RAC 構成で使用して、プロデューサ・スレッドの数を指定します。これらは、様々な RAC ノードで異なる REDO ログを読み取る Extract スレッドです。この値は、REDO データの取得元のノードの数と同じである必要があります。

引数	説明
PASSIVE	この Extract グループをパッシブ・モードで実行し、ターゲット・システム上の別名 Extract グループを起動または停止することによってのみ起動および停止できるように指定します。ソースおよびターゲット間の接続は、このグループではなく、ターゲットの別名 Extract グループによって確立されます。 このオプションは、標準 Extract グループまたはデータ・ポンプ Extract グループに対して使用できます。これは、ネットワークを介してターゲット上のリモート・トレイルにデータを送信するソース・システム上の Extract に対してのみ使用する必要があります。 パッシブおよび別名 Extract グループの構成方法は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。
DESC "<description>"	グループの説明 (`"Extracts account_tab on Serv1"` など) を指定します。この説明は引用符で囲む必要があります。短縮キーワード `DESC` または完全なキーワード `DESCRIPTION` を使用できます。
RMTHOST {<host name> | <IP address>}	このグループを別名 Extract として識別し、リモート・ホストの DSN 名またはその IP アドレスを指定します。
MGRPORT <port>	別名 Extract に対して使用し、Manager が実行中のリモート・システム上のポートを指定します。
RMTNAME <name>	別名 Extract に対して使用します。別名 Extract の名前と異なる場合に、パッシブ Extract の名前を指定します。

ADD EXTRACT の例

例 1 次の例では、トランザクション・ログからデータベースの変更を抽出する、"finance" という名前の Extract グループを作成します。抽出は、ADD EXTRACT によってこのグループが作成された時刻に生成されたレコードから開始されます。

```
ADD EXTRACT finance, TRANLOG, BEGIN NOW
```

例 2 次の例では、Oracle RAC ログからデータベースの変更を抽出する、"finance" という名前の Extract グループを作成します。抽出は、このグループが作成された時刻に生成されたレコードから開始されます。RAC インスタンスが 4 つあるので、Extract スレッド数は 4 になります。

```
ADD EXTRACT finance, TRANLOG, BEGIN NOW, THREADS 4
```

例 3 次の例では、トランザクション・ログからデータベースの変更を抽出する、"finance" という名前の Extract グループを作成します。抽出は、2011 年 1 月 21 日 8 時に生成されたレコードから開始されます。

```
ADD EXTRACT finance, TRANLOG, BEGIN 2011-01-21:08:00
```

例 4 次の例では、統合キャプチャの Extract グループを作成します。

```
ADD EXTRACT finance, INTEGRATED TRANLOG, BEGIN NOW
```

例 5 次の例では、最大パフォーマンス・モードまたは最大保護モードの Teradata TAM とインタフェースを取る、"finance" という名前の Extract グループを作成します。Teradata ソースには、BEGIN ポイントを使用しません。

```
ADD EXTRACT finance, VAM
```

例 6 次の例では、"finance" という名前の VAM ソート Extract グループを作成します。このプロセスは、VAM トレイル /ggs/dirdat/vt から読み取ります。

```
ADD EXTRACT finance, VAMTRAILSOURCE dirdat/vt
```

例 7 この例では、"finance" という名前のデータ・ポンプ Extract グループを作成し、Oracle GoldenGate トレイル c:\ggs\dirdat\lt から読み取ります。

```
ADD EXTRACT finance, EXTTRAILSOURCE dirdat\lt
```

例 8 次の例では、"load" という名前の初期ロード Extract を作成します。

```
ADD EXTRACT load, SOURCEISTABLE
```

例 9 次の例では、トランザクション・ログからデータベースの変更を抽出する、"finance" という名前のパッシブ Extract グループを作成します。

```
ADD EXTRACT finance, TRANLOG, BEGIN NOW, PASSIVE
```

例 10 次の例では、"financeA" という名前の別名 Extract グループを作成します。この別名 Extract は、ソース・システム sysA 上の "finance" という名前のパッシブ Extract に関連付けられます。このシステム上の Manager は、ポート 7800 を使用しています。

```
ADD EXTRACT financeA, RMTHOST sysA, MGRPORT 7800, RMTNAME finance
```

ALTER EXTRACT

ALTER EXTRACT は、次の目的で使用します。

- ADD EXTRACT コマンドで作成した Extract グループの属性を変更する
- 次の順序のトレイル・ファイルに切り替える
- 統合キャプチャの構成にアップグレードする
- 統合キャプチャの構成からダウングレードする

このコマンドを使用する前に、STOP EXTRACT <group name> コマンドで Extract を停止してください。

構文

```
ALTER EXTRACT <group name>
[, <ADD EXTRACT attribute>]
[, UPGRADE INTEGRATED TRANLOG]
[, DOWNGRADE INTEGRATED TRANLOG]
[, THREAD <number>]
[, ETROLLOVER]
```

引数	説明
<group name>	変更する Extract グループの名前。

引数	説明

<ADD EXTRACT attribute>	次の例外を除き、ADD EXTRACT コマンドで指定した任意の属性を変更できます。
	◆ EXTTRAILSOURCE オプションで指定した Extract の変更。
	◆ THREADS オプションで指定した RAC スレッド数の変更。
	このような例外に該当する場合は、対象の Extract グループを削除し、このグループをもう一度追加してください。
	BEGIN オプションを使用している場合は、同じ文に他のオプションを指定しないでください。次の例のように、別の文を発行してください。
	`ALTER EXTRACT finance, BEGIN 2011-01-01`
	`ALTER EXTRACT finance, ETROLLOVER`

UPGRADE INTEGRATED TRANLOG	Extract グループをクラシック・キャプチャから統合キャプチャにアップグレードします。このコマンドを使用する前に、DBLOGIN を使用してソース Oracle データベースに接続するか、MININGDBLOGIN コマンドを使用してダウンストリーム・マイニング・データベースに接続します。このコマンドを発行した後で、REGISTER EXTRACT を DATABASE 引数とともに発行して、Extract をそのデータベースに登録します。アップグレードをサポートするには、オープンしている最も古いトランザクションの開始を含むトランザクション・ログが、ソースまたはダウンストリーム・マイニング・システムで使用可能である必要があります。統合キャプチャの詳細は、Oracle データベースのドキュメントを参照してください。
DOWNGRADE INTEGRATED TRANLOG	Extract グループを統合キャプチャからクラシック・キャプチャにダウングレードします。このコマンドを使用する前に、DBLOGIN を使用してソース Oracle データベースに接続するか、MININGDBLOGIN コマンドを使用してダウンストリーム・マイニング・データベースに接続します。このコマンドを発行した後で、UNREGISTER EXTRACT を DATABASE 引数とともに発行して、Extract をそのデータベースから登録解除します。ダウングレードをサポートするには、オープンしている最も古いトランザクションの開始を含むトランザクション・ログが、ソースまたはダウンストリーム・マイニング・システムで使用可能である必要があります。統合キャプチャの詳細は、Oracle データベースの Oracle GoldenGate のドキュメントを参照してください。
THREAD <number>	Oracle RAC 構成で、指定した REDO スレッドに対してのみ Extract を変更します。指定できるスレッド番号は 1 つのみです。
ETROLLOVER	Extract の再起動時に、次の順序のトレイル・ファイルに切り替えさせます。たとえば、現在のファイルが ET000002 の場合、Extract を再起動すると現在のファイルは ET000003 になります。トレイルは 000001 から 999999 まで増やすことが可能で、順序番号は 000000 から再開始します。

例 1 次の例では、2011 年 1 月 1 日からのデータの処理を開始するように Extract の設定を変更します。

```
ALTER EXTRACT finance, BEGIN 2011-01-01
```

例 2 次の例では、トレイル内の特定の場所から処理を開始するように Extract の設定を変更します。

```
ALTER EXTRACT finance, EXTSEQNO 26, EXTRBA 338
```

例 3 次の例では、Oracle RAC 環境の Extract を変更し、REDO スレッド 4 のみに新しい開始位置を適用します。

```
ALTER EXTRACT accounts, THREAD 4, BEGIN 2011-01-01
```

例 4 次の例では、特定の LSN から処理を開始するように SQL Server 環境の Extract の設定を変更します。

```
ALTER EXTRACT sales, LSN 3454:875:445
```

例 5 次の例では、Extract の設定を変更して、順序の次のトレイル・ファイルに切り替えさせます。

```
ALTER EXTRACT finance, ETROLLOVER
```

例 6 次の例では、統合キャプチャにアップグレードするように Extract の設定を変更します。

```
ALTER EXTRACT finance, UPGRADE INTEGRATED TRANLOG
```

例 7 次の例では、クラシック・キャプチャにダウングレードするように Extract の設定を変更します。

```
ALTER EXTRACT finance, DOWNGRADE INTEGRATED TRANLOG
```

CLEANUP EXTRACT

CLEANUP EXTRACT では、特定の Extract グループの実行履歴を削除します。このコマンドでは、最後の実行レコードがそのまま保持されるので、Extract は停止した場所から処理を再開できます。このコマンドを使用する前に、STOP EXTRACT コマンドを発行して Extract を停止してください。

構文

```
CLEANUP EXTRACT <group name> [, SAVE <count>]
```

引数	説明
`<group name>`	1 つの Extract グループ名、または複数のグループを指定するワイルドカード (*)。たとえば、「T*」と指定すると、名前が T から始まるすべての Extract グループの実行履歴が削除されます。
`SAVE <count>`	指定した数の最新のレコードを削除から除外します。

例 1 次の例では、最新レコード以外のすべてのレコードを削除します。

```
CLEANUP EXTRACT finance
```

例 2 次の例では、最新の 5 レコード以外のすべてのレコードを削除します。

```
CLEANUP EXTRACT *, SAVE 5
```

DELETE EXTRACT

DELETE EXTRACT では、Extract グループを削除します。このコマンドを実行すると、グループに属するチェックポイント・ファイルは削除されますが、パラメータ・ファイルはそのまま残ります。グループの削除後、必要に応じてグループをもう一度作成することも、パラメータ・ファイルを削除することもできます。

DELETE EXTRACT を使用する前に、STOP EXTRACT コマンドを使用して Extract を停止します。

Extract グループに関連付けられたトレイル・ファイルを削除するには、オペレーティング・システムを介して手動で削除します。

構文

```
DELETE EXTRACT <group name> [!]
```

引数	説明
<group name>	1 つの Extract グループ名、または複数のグループを指定するワイルドカード (*)。たとえば、「T*」と指定すると、名前が T から始まるすべての Extract グループが削除されます。
!	(感嘆符) プロンプトを表示せずに、ワイルドカードに一致するすべての Extract グループを削除します。

INFO EXTRACT

INFO EXTRACT では、次の情報を表示します。

- Extract のステータス (STARTING、RUNNING、STOPPED または ABENDED)
- おおよその Extract ラグ
- チェックポイント情報
- プロセス実行履歴
- Extract の書込み先トレイル
- 統合キャプチャのアップグレードまたはダウングレードのステータス

INFO EXTRACT を発行すると、Extract は実行中または停止になります。実行中のプロセスの場合、RUNNING のステータスは、次のいずれかを意味します。

- アクティブ : 実行中および処理中 (または処理可能な) データ。これは、プロセス開始後のプロセスの通常の状態です。
- 一時停止 : プロセスは実行中ですが、EVENTACTIONS SUSPEND アクションにより一時停止しました。一時停止状態では、プロセスはアクティブではなく、データを処理できませんが、現在の実行の状態は保持され、GGSCI で RESUME コマンドを発行すると続行できます。INFO コマンドでの RBA は、一時停止アクションの前の最新のチェックポイント位置を表します。状態がアクティブか一時停止かを確認するには、SEND EXTRACT コマンドを STATUS オプションとともに発行します。

TASKS または ALLPROCESSES 引数なしの基本コマンドでは、オンライン (継続的な)Extract プロセスの情報のみを表示します。タスクは除外されます。

次に例を示します。

図 1 INFO EXTRACT

```
EXTRACT                     EXTCUST Last Started 2011-01-05 16:09 Status RUNNING
Checkpoint Lag           00:01:30 (updated 97:16:45 ago)
Log Read Checkpoint File  /rdbms/data/oradata/redo03a.log
                          2011-01-05 16:05:17  Seqno 2952, RBA 7598080
```

ラグについて

チェックポイント・ラグは、最新のチェックポイントがトレイルに書き込まれた時点での遅延 (秒) を表します。次に例を示します。

- 現在の時刻 = 15:00:00
- 最新のチェックポイント = 14:59:00
- 最新の処理レコードのタイムスタンプ = 14:58:00

この場合、ラグは 00:01:00(1 分、14:58 と 14:59 の差異) とレポートされます。

UNKNOWN というラグ値は、プロセスは実行しているがレコードをまだ処理していないか、(タイム・ゾーンの差ではなくクロックが正確でないために) ソース・システムのクロックがターゲット・システムのクロックよりも進んでいることを示します。

より正確なラグ情報を取得するには、LAG EXTRACT を使用します (34 ページを参照してください)。

詳細情報の表示

次に、DETAIL オプションの出力の例を示します。

図 2 INFO EXTRACT と DETAIL

```
EXTRACT    ORAEXT    Last Started 2011-01-15 16:16   Status STOPPED
Checkpoint Lag       00:00:00 (updated 114:24:48 ago)
Log Read Checkpoint  File C:\ORACLE\ORADATA\ORA920\REDO03.LOG
                     2011-01-15 16:17:53 Seqno 46, RBA 3757568

Target Extract Trails:

Remote Trail Name                     Seqno    RBA   Max MB

  c:\goldengate802\dirdat\xx              0  57465       10
  c:\goldengate802\dirdat\jm              0  19155       10

Extract Source                          Begin             End

C:\ORACLE\ORADATA\ORA920\REDO03.LOG    2011-01-15 16:07 2011-01-15 16:17
C:\ORACLE\ORADATA\ORA920\REDO03.LOG    2011-01-15 15:55  2011-01-15 16:07
C:\ORACLE\ORADATA\ORA920\REDO03.LOG    2011-01-15 15:42  2011-01-15 15:55
C:\ORACLE\ORADATA\ORA920\REDO03.LOG   2011-01-15 15:42  2011-01-15 15:42
  Not Available                         * Initialized *   2011-01-15 15:42

Current directory    C:\GoldenGate802

Report file          C:\GoldenGate802\dirrpt\ORAEXT.rpt
Parameter file       C:\GoldenGate802\dirprm\ORAEXT.prm
Checkpoint file      C:\GoldenGate802\dirchk\ORAEXT.cpe
Process file         C:\GoldenGate802\dirpcs\ORAEXT.pce
Error log            C:\GoldenGate802\ggserr.log
```

チェックポイントの表示

正確なチェックポイントの位置は、データ・ソースの読取りチェックポイントとトレイルの書込みチェックポイントで構成されます。次に、SHOWCH オプションで表示されるチェックポイント情報の例を示します。ここでは、データ・ソースは Oracle RAC データベース・クラスタなので、出力にはスレッド情報も含まれています。過去のチェックポイントを表示するには、表示するチェックポイント番号を SHOWCH エントリに続いて指定します。

図 3 INFO EXTRACT と SHOWCH

```
EXTRACT    JC108XT Last Started 2011-01-01 14:15   Status ABENDED
Checkpoint Lag       00:00:00 (updated 00:00:01 ago)
Log Read Checkpoint  File /orarac/oradata/racq/redo01.log
           2011-01-01 14:16:45  Thread 1, Seqno 47, RBA 68748800
Log Read Checkpoint  File /orarac/oradata/racq/redo04.log
           2011-01-01 14:16:19  Thread 2, Seqno 24, RBA 65657408

Current Checkpoint Detail:

Read Checkpoint #1

    Oracle RAC Redo Log

    Startup Checkpoint(starting position in data source):
        Thread #: 1
        Sequence #: 47
        RBA: 68548112
        Timestamp: 2011-01-01 13:37:51.000000
        SCN: 0.8439720
        Redo File: /orarac/oradata/racq/redo01.log

    Recovery Checkpoint (position of oldest unprocessed transaction in data
    source):
        Thread #: 1
        Sequence #: 47
        RBA: 68748304
        Timestamp: 2011-01-01 14:16:45.000000
        SCN: 0.8440969
        Redo File: /orarac/oradata/racq/redo01.log

    Current Checkpoint (position of last record read in the data source):
        Thread #: 1
        Sequence #: 47
        RBA: 68748800
        Timestamp: 2011-01-01 14:16:45.000000
        SCN: 0.8440969
        Redo File: /orarac/oradata/racq/redo01.log

Read Checkpoint #2

    Oracle RAC Redo Log
```

```
    Startup Checkpoint(starting position in data source):
        Sequence #: 24
        RBA: 60607504
        Timestamp: 2011-01-01 13:37:50.000000
        SCN: 0.8439719
        Redo File: /orarac/oradata/racq/redo04.log

    Recovery Checkpoint (position of oldest unprocessed transaction in data
    source):
        Thread #: 2
        Sequence #: 24
        RBA: 65657408
        Timestamp: 2011-01-01 14:16:19.000000
        SCN: 0.8440613
        Redo File: /orarac/oradata/racq/redo04.log

    Current Checkpoint (position of last record read in the data source):
        Thread #: 2
        Sequence #: 24
        RBA: 65657408
        Timestamp: 2011-01-01 14:16:19.000000
        SCN: 0.8440613
        Redo File: /orarac/oradata/racq/redo04.log

Write Checkpoint #1

    GGS Log Trail

    Current Checkpoint (current write position):

        Sequence #: 2
        RBA: 2142224
        Timestamp: 2011-01-01 14:16:50.567638
        Extract Trail: ./dirdat/eh

    Header:
        Version = 2
        Record Source = A
        Type = 6
        # Input Checkpoints = 2
        # Output Checkpoints = 1

    File Information:
        Block Size = 2048
        Max Blocks = 100
        Record Length = 2048
        Current Offset = 0

    Configuration:
        Data Source = 3
        Transaction Integrity = 1
        Task Type = 0
```

```
Status:
    Start Time = 2011-01-01 14:15:14
    Last Update Time = 2011-01-01 14:16:50
    Stop Status = A
    Last Result = 400
```

Extract 読取りチェックポイントについて

Extract は、データ・ソースに読取りチェックポイントを作成します。

開始チェックポイント

開始チェックポイントは、プロセスの起動時にデータ・ソースに作成される最初のチェックポイントです。この統計は、次のもので構成されます。

- Thread #: Oracle GoldenGate が Oracle RAC 環境で実行されている場合は、このチェックポイントを作成した Extract スレッドの番号。それ以外の環境で実行されている場合、この統計は表示されません。
- Sequence #: チェックポイントが作成されたトランザクション・ログの順序番号。
- RBA: チェックポイントが作成されたレコードの相対バイト・アドレス。
- Timestamp: チェックポイントの作成時のレコードのタイムスタンプ。
- SCN: チェックポイントが作成されたレコードのシステム変更番号。
- Redo File: チェックポイントが作成されたレコードを含むトランザクション・ログのパス名。

リカバリ・チェックポイント

リカバリ・チェックポイントは、Extract によってまだ処理されていない最も古いトランザクションを含むレコードのデータ・ソース内の位置です。この統計のフィールドは、他のタイプの読取りチェックポイントに含まれるフィールドと同じです。

現在のチェックポイント

現在のチェックポイントは、Extract が読み取った最新レコードのデータ・ソース内の位置です。この値は、サマリー、およびオプションなしの基本の INFO EXTRACT コマンドで表示される Log Read Checkpoint 統計と同じになるはずです。この統計のフィールドは、他のタイプの読取りチェックポイントに含まれるフィールドと同じです。

Extract 書込みチェックポイントについて

Extract は、トレイルに書込みチェックポイントを作成します。

現在のチェックポイント

現在のチェックポイントは、Extract が現在書き込んでいるトレイル内の位置です。この統計は、次のもので構成されます。

- Sequence #: チェックポイントが書き込まれたトレイル・ファイルの順序番号。
- RBA: チェックポイントが作成されたトレイル・ファイル内レコードの相対バイト・アドレス。
- Timestamp: チェックポイントの作成時のレコードのタイムスタンプ。
- Extract trail: トレイルの相対パス名。

その他の SHOWCH 情報

SHOWCH 出力の最後に表示される Header、File Information、Configuration および Status 統計は、Oracle サポートのアナリストが使用する情報です。これらの統計には、サポート・ケースの解決に有益な内部情報が含まれています。

構文

```
ALTER EXTRACT <group name>
[, SHOWCH [<n>]]
[, DETAIL]
[, TASKS | ALLPROCESSES]
[, UPGRADE | DOWNGRADE]
```

引数	説明
`<group name>`	1 つの Extract グループ名、または複数のグループを指定するワイルドカード (*)。たとえば、「T*」と指定すると、名前が T から始まるすべての Extract グループの情報が表示されます。
`SHOWCH [<n>]`	基本コマンドでは、現在の Extract チェックポイントの情報を表示します。 <n> には、現在のチェックポイントに加えて表示する過去のチェックポイントの番号を指定します。 注意 : 出力に不規則なインデントおよび空白が表示されることがあります。これは正常な出力で、情報の正確性に影響はありません。
`DETAIL`	次の情報を表示します。 ◆ データ・ソース内の開始位置および停止位置 (時刻で表示) を含む Extract 実行履歴。 ◆ Extract の書込み先トレイル。
`TASKS`	Extract タスクのみを表示します。ワイルドカード引数で指定したタスクは、INFO EXTRACT では表示されません。
`ALLPROCESSES`	タスクを含むすべての Extract グループを表示します。
`UPGRADE | DOWNGRADE`	◆ UPGRADE は、Extract をクラシック・キャプチャ・モードから統合キャプチャ・モードにアップグレードできるかどうかを表示します。 ◆ DOWNGRADE は、Extract を統合キャプチャ・モードからクラシック・キャプチャ・モードにダウングレードできるかどうかを表示します。 Extract をアップグレードまたはダウングレードできない場合、その理由が表示されます。 このオプションでは、ワイルドカードの Extract 名は許可されていません。 このコマンドを使用する前に、DBLOGIN コマンドを発行します。

例 1
```
INFO EXTRACT fin*, SHOWCH
```

例 2
```
INFO EXTRACT *, TASKS
```

例 3
```
INFO EXTRACT finance UPGRADE
```

KILL EXTRACT

KILL EXTRACT では、標準または**パッシブ**モードで実行している Extract プロセスを中断します。このコマンドは、STOP EXTRACT コマンドでプロセスを正常に停止できない場合にのみ使用してください。中断された Extract プロセスは、Manager プロセスによって再起動されません。

構文

```
KILL EXTRACT <group name>
```

引数	説明
<group name>	1 つの Extract グループ名、または複数のグループを指定するワイルドカード (*)。たとえば、「T*」と指定すると、名前が T から始まるすべての Extract グループが中断されます。

例

```
KILL EXTRACT finance
```

LAG EXTRACT

LAG EXTRACT では、Extract とデータ・ソース間の正確なラグ・タイムを確認します。LAG EXTRACT は、トレイル内のチェックポイントの位置を読み取るのでなく、Extract と直接通信するので、INFO EXTRACT よりも正確にラグ・タイムを計算できます。

Extract のラグについて

Extract のラグとは、(システム・クロックに基づく)Extract がレコードを処理した時刻と、データ・ソース内のそのレコードのタイムスタンプとの差 (秒) です。

次に LAG EXTRACT の出力の例を示します。

図 4 LAG EXTRACT 出力

```
Sending GETLAG request to EXTRACT CAPTPCC...
Last record lag: 2 seconds.
At EOF, no more records to process.
```

構文

```
LAG EXTRACT <group name>
```

引数	説明
<group name>	1 つの Extract グループ名、または複数のグループを指定するワイルドカード (*)。たとえば、「T*」と指定すると、名前が T から始まるすべての Extract グループのラグ・タイムを確認できます。

例 1

```
LAG EXTRACT *
```

例 2

```
LAG EXTRACT *fin*
```

REGISTER EXTRACT

REGISTER EXTRACT では、次の目的で、プライマリ Extract グループを Oracle データベースに登録します。

- 統合キャプチャ・モードを有効にする。
- リカバリに必要なアーカイブ・ログを保持するため、Oracle Recovery Manager で動作するように、Extract をクラシック・キャプチャ・モードで有効にする。

REGISTER EXTRACT は、データ・ポンプ Extract には無効です。

Extract グループをデータベースから登録解除するには、UNREGISTER EXTRACT コマンドを使用します (52 ページを参照してください)。

構文

```
REGISTER EXTRACT <group name>
{DATABASE | LOGRETENTION}
```

引数	説明
`<group name>`	登録する Extract グループの名前。ワイルドカードは使用しないでください。
`DATABASE`	Extract グループに対して統合キャプチャを有効にします。このモードでは、Extract は、論理変更レコード (LCR) の形式で変更データを受信するデータベース・ログマイニング・サーバーと統合します。Extract は REDO ログを読み取りません。Extract は、取得処理、フィルタリング、変換およびその他の要件を実行します。サポート情報および構成手順は、Oracle データベースの Oracle GoldenGate のドキュメントを参照してください。 REGISTER EXTRACT を DATABASE とともに使用する前に、dbms_goldengate_auth.grant_admin_privilege プロシージャを使用して付与された権限で、DBLOGIN または MININGDBLOGIN コマンドを使用します。追加の要件は、これらのコマンドを参照してください。 REGISTER EXTRACT を使用した後で、ADD EXTRACT を INTEGRATED TRANLOG オプションとともに使用して、同じ名前の Extract グループを作成します。 以前に、この Extract に対して REGISTER EXTRACT が LOGRETENTION とともに使用されていた場合、ログ・ファイルは統合キャプチャの一部として管理されるため、ログ保持の目的で、これは自動的に登録解除されます。
`LOGRETENTION`	Extract がリカバリに必要とするログを保持するため、Oracle Recovery Manager (RMAN) で動作するように、Extract グループをクラシック・キャプチャ・モードで有効にします。LOGRETENTION は、次の特徴を持つ、基盤の Oracle Streams 取得プロセスを作成します。 ◆ Extract グループに専用で、ログ保持の目的でのみ使用される ◆ 類似した名前を持つ Extract グループが統合キャプチャ用に構成されている場合、LOGRETENTION は無視されます。

引数	説明
	ログは、現在のデータベース SCN に基づいて、REGISTER EXTRACT が発行された時点から保持されます。ログ保持機能は、TRANLOGOPTIONS パラメータの LOGRETENTION オプションで制御されます。 REGISTER EXTRACT を LOGRETENTION とともに使用する前に、DBLOGIN コマンドを 78 ページの表 1 で示した権限で発行します。

例 1 REGISTER EXTRACT sales LOGRETENTION

例 2 REGISTER EXTRACT sales DATABASE

SEND EXTRACT

SEND EXTRACT では、実行中の Extract プロセスと通信します。このリクエストは、Extract がユーザーからのコマンドを受け入れる準備ができるとすぐに処理されます。

構文

```
SEND EXTRACT <group name>, {
BR {BRINTERVAL <interval> |
    BRSTART |
    BRSTOP |
    BRCHECKPOINT {IMMEDIATE | IN <N>{H|M} | AT YYYY-MM-DD HH:MM[:SS]}} |
CACHEMGR {CACHESTATS | CACHEQUEUES | CACHEPOOL} |
FORCESTOP |
FORCETRANS <Transaction ID> [THREAD <n>] [FORCE] |
GETLAG |
GETTCPSTATS |
LOGEND |
LOGSTATS |
REPORT |
RESUME |
ROLLOVER |
SHOWTRANS [<Transaction ID>] [THREAD <n>] [COUNT <n>]
    [DURATION <duration><unit>] [TABULAR]
    [FILE <name> [DETAIL]] |
SKIPTRANS <Transaction ID> [THREAD <n>] [FORCE] |
STATUS |
STOP |
TRACE[2] <tracefile> |
TRACE[2] OFF |
TRACE OFF <file name> |
TRACEINIT |
TRANLOGOPTIONS {PURGEORPHANEDTRANSACTIONS | NOPURGEORPHANEDTRANSACTIONS} |
TRANLOGOPTIONS TRANSCLEANUPFREQUENCY <minutes> |
VAMMESSAGE "<Teradata command>" |
VAMMESSAGE {"ARSTATS" | "INCLUDELIST [filter]" | "EXCLUDELIST [filter]"} |
VAMMESSAGE "OPENTRANS"
}
```

引数	説明
`<group name>`	1 つの Extract グループ名、または複数のグループを指定するワイルドカード (*)。たとえば、「T*」と指定すると、グループ名が T から始まるすべての Extract プロセスにこのコマンドが送信されます。Extract が実行中でない場合はエラーが返されます。
`BR {BRINTERVAL <interval> | BRSTART | BRSTOP | BRSTATS | BRCHECKPOINT {IMMEDIATE | IN <N>{H|M} | AT YYYY-MM-DD HH:MM[:SS]}}`	Extract の Bounded Recovery モードに影響を及ぼすコマンドを送信します。 BRINTERVAL <interval> Bounded Recovery チェックポイント間の時間を設定します。有効な値は 20 分から 96 時間です。分は M で、時間は H で指定します。デフォルト間隔は 4 時間です。 BRSTART Bounded Recovery を開始します。このコマンドは、Oracle サポートの指示の下でのみ使用します。 BRSTOP Bounded Recovery の実行およびリカバリを停止します。このオプションを使用する前に Oracle サポートに連絡してください。通常の場合、Bounded Recovery に問題があると、Bounded Recovery は自身を無効化します。 BRCHECKPOINT {IMMEDIATE | IN <N>{H|M} | AT YYYY-MM-DD HH:MM[:SS]}} Bounded Recovery チェックポイントがいつ作成されるかを設定します。 ◆ IMMEDIATE では、Extract は、SEND EXTRACT が発行されるとすぐにチェックポイントを発行します。 ◆ IN <N>{H | M} では、Extract は、SEND EXTRACT が発行されてから、指定された時間数の経過後にチェックポイントを発行します。 ◆ AT YYYY-MM-DD HH:MM[:SS]}} では、Extract は、指定された時間にチェックポイントを発行します。
`CACHEMGR {CACHESTATS | CACHEQUEUES | CACHEPOOL <n>}`	Oracle GoldenGate メモリー・キャッシュ・マネージャについての統計を返します。 ◆ CACHESTATS は、仮想メモリーの使用量とファイル・キャッシュに関する CACHEMGR 統計を返します。 ◆ CACHEQUEUESCACHEMGR は、空きキューに関する統計のみを返します。 ◆ CACHEPOOL <n> は、指定されたオブジェクト・プールに関する統計のみを返します。 CACHESTATS は、Oracle サポートから指示が明示的にあった場合にのみ使用します。

引数	説明
FORCESTOP	すべての通知をバイパスし、Extract を強制的に停止します。このコマンドを発行すると、プロセスは即座に停止します。
FORCETRANS <Transaction ID> [THREAD <n>] [FORCE]	(MySQL、Oracle、SQL Server、Sybase) Extract に、トランザクション ID 番号で指定したトランザクションをコミット済トランザクションとしてトレイルに書き込ませます。トランザクション ID 番号は、SHOWTRANS コマンドまたは Extract ランタイム・メッセージから取得してください。Extract は、このコマンドの発行後にトランザクションに追加されたデータはすべて無視します。FORCE が使用されていない場合、確認プロンプトに応える必要があります。FORCETRANS を使用するには、SHOWTRANS コマンドで表示されるトランザクション・リスト内の最も古いトランザクションを指定する必要があります。 オプション : ◆ THREAD <n> では、異なるスレッドに重複するトランザクション ID がある場合に、Oracle RAC 環境でこのトランザクションを生成したスレッドを指定します。 ◆ FORCE では、確認プロンプトをバイパスします。 FORCETRANS では、ソース・データベースに対するトランザクションのコミットは行いません。Replicat が (暗黙的なコミットで) 処理できるように、既存データをトレイルのみに書き込ませます。 このコマンドは、古い順に他のトランザクションに対して繰り返し実行できます。 FORCETRANS を使用した後、SEND EXTRACT コマンドを FORCESTOP とともに発行する場合は、少なくとも 5 分待機してください。待機しないと、トランザクションはそのまま残ります。 Extract の起動直後に FORCETRANS を使用する場合、待機してからコマンドを再度実行するように求めるエラー・メッセージを受信することがあります。これは、Extract がまだ他のトランザクションを処理していないことを意味しています。Extract が他のトランザクションの処理を開始すると、トランザクションをトレイルに書き込ませることができます。
GETLAG	Extract とデータ・ソース間の正確なラグ・タイムを確認します。LAG EXTRACT(34 ページを参照してください) と同じ結果が返されます。
GETTCPSTATS	Extract とターゲット・システム間のネットワーク・アクティビティについての統計を表示します。次の統計が含まれます。 ◆ ローカルおよびリモート IP アドレス ◆ インバウンドおよびアウトバウンド・メッセージ (バイトおよびバイト / 秒) ◆ 受信(インバウンド)および送信(アウトバウンド)数。受信は、インバウンド・メッセージ当たり少なくとも 2 つ (長さについて 1 つ、データについて 1 つ以上) あります。

引数	説明
	◆ 送信および受信当たりの平均バイト数 ◆ 送信および受信の待機時間 : 送信の待機時間とは、TCP への書込み完了にかかる時間です。送信の待機時間が短いほど、ネットワークのパフォーマンスが優れています。受信の待機時間とは、読取り完了にかかる時間です。送信および受信の待機時間によって、おおよそのネットワークのラウンド・トリップ時間を把握できます。これらの値は、マイクロ秒で表されます。 ◆ データ圧縮のステータス (有効または無効) ◆ 圧縮前バイト数と圧縮後バイト数 : 圧縮前と圧縮後を比較すると圧縮率 (圧縮前と圧縮後のバイトの差) を把握できます。この圧縮率と、1 秒当たりの圧縮バイト数を比較することで、リソースおよびネットワーク消費の観点から圧縮率がコストに見合うかどうか判断できます。 RMTHOST および RMTHOSTOPTIONS の TCPBUFSIZE オプションでは、圧縮前データの TCP バッファ・サイズを制御します。圧縮が有効の場合、ネットワークに送信されるデータはこのサイズよりも小さくなります。GETTCPSTATS では、圧縮後のスループットを表示します。
LOGEND	Extract がデータ・ソース内のすべてのレコードを処理したかどうかを確認します。
LOGSTATS	(Oracle) Extract に、Oracle REDO ログ・ファイルからのデータの処理に関連する統計についてのレポートを発行するように指示します。ログ・ファイル上で追加の I/O なしにデータが使用できるように、Extract は、Extract が処理中の現在のレコードの先を読み取る非同期ログ・リーダーを使用します。次の統計があります。 ◆ AsyncReader.Buffers*n*: 取得された REDO データを含む各バッファ・キューに対して、このようなフィールドがあります。サイズ、その中のレコード数、データを処理するまでの待機時間の長さを示します。これらの統計は、キューでの書込み操作および読取り操作に対して指定されます。 ◆ REDO **先読みバッファ** : 非同期で先読みするのに使用されているバッファの数。 ◆ REDO **先読みバッファ・サイズ** : 各バッファのサイズ。 ◆ **現在の** REDO **の先読み** REDO **バイト** : 現在の REDO ログ・ファイルに対して先読みモードがオンかオフかどうか (値は ON または OFF)。 ◆ **読取り** REDO **バイト** : この Extract のインスタンスに関連付けられているすべての REDO ログ・ファイルから読み取られたバイト数。

引数	説明
	◆ **先読み REDO バイト**：先読みメカニズムによって処理されたバイト数。 ◆ **未使用 REDO バイト**：Extract 位置変更や失効読取りの結果、その後に削除された先読みバイト数。 ◆ **解析 REDO バイト**：有効なログ・データとして処理されたバイト数。 ◆ **出力 REDO バイト**：トレイル・ファイルに書き込まれたバイト数 (内部 Oracle GoldenGate のオーバーヘッドは含まない)。
`REPORT`	Extract レポート・ファイルに一時的な統計レポートを生成します。RESETREPORTSTATS | NORESETREPORTSTATS とともに使用する場合、表示される統計は、STATOPTIONS パラメータの構成によって異なります。336 ページを参照してください。
`RESUME`	EVENTACTIONS SUSPEND イベントによって一時停止されたプロセスを再開 (アクティブに) します。プロセスは、一時停止された位置から通常の処理を再開します。
`ROLLOVER`	Extract の再起動時に、順序の次のトレイル・ファイルに切り替えさせます。たとえば、現在のファイルが ET000002 の場合、このコマンドの実行後、現在のファイルは ET000003 になります。トレイルは 000001 から 999999 まで増やすことが可能で、順序番号は 000000 から再開始します。

`SHOWTRANS [<Transaction ID>]`
`[THREAD <n>]`
`[COUNT <n>]`
`[DURATION <duration><unit>]`
`[TABULAR] |`
`[FILE <name> [DETAIL]]`
	(MySQL、Oracle、SQL Server、Sybase) オープンしているトランザクションについての情報を表示します。SHOWTRANS は、データベース・タイプに応じて次のいずれかを表示します。
	◆ プロセス・チェックポイント (Extract が再起動する場合に、トランザクション処理を継続する必要がある最も古いログを示します)
	◆ トランザクション ID
	◆ Extract グループ名
	◆ REDO スレッド番号
	◆ (トランザクションの実際の開始時刻でなく)Oracle GoldenGate がトランザクションから抽出した最初の操作のタイムスタンプ
	◆ システム変更番号 (SCN)
	◆ REDO ログ番号と RBA
	◆ ステータス (Pending COMMIT または Running)。Pending COMMIT は、FORCETRANS 発行後のトランザクション書込み中に表示されます。

引数	説明
	オプションを指定しない場合 SHOWTRANS では、使用可能なバッファに収まるオープンしているすべてのトランザクションが表示されます。出力をさらに制御するには、次のオプションを参照してください。 SHOWTRANS オプション : ◆ <Transaction ID> では、コマンド出力を特定のトランザクションに制限します。 ◆ THREAD <n> では、コマンド出力を特定の Oracle RAC スレッドに対してオープンしているトランザクションに制限します。<n> には、Extract に認識されている RAC スレッド番号を指定します。 ◆ COUNT <n> では、コマンド出力を指定した数のオープンしているトランザクションに制限します (最も古いトランザクションから開始します)。有効な値は 1 ～ 100,000 です。 ◆ DURATION <duration><unit> では、コマンド出力を指定した時間より長くオープンしているトランザクションに制限します。 <duration> には、時間の長さを整数で指定します。 <unit> には、second、minutes、hours または days を完全なスペリングまたは短縮形式で指定します。 S | SEC | SECS | SECOND | SECONDS M | MIN | MINS | MINUTE | MINUTES H | HOUR | HOURS D | DAY | DAYS 各レコードにタイムスタンプを付けない Sybase の場合、duration は必ずしも正確ではなく、トランザクション・ログに格納されている BEGIN レコードと COMMIT レコードの時間情報に依存します。 ◆ (Oracle) TABULAR では、SQL*Plus のデフォルトの表出力のような表フォーマットで出力を生成します。デフォルトは field-per-row です。 ◆ FILE <name> では、Extract に、指定したファイルにトランザクション情報を書き込ませます。コンソールへの出力はありません。 ◆ (Oracle) FILE <name> DETAIL は、16 進数およびプレーン文字のデータ・ダンプを書き込みます。これにより、メモリーからファイルにトランザクション全体がダンプされます。このデータを確認することにより、トランザクションをスキップするかトレイルに強制的に書き込むかを判別できます。 **注意 :** 基本の詳細情報は、WARNLONGTRANS CHECKINTERVAL パラメータに指定されている間隔でレポート・ファイルに自動的に書き込まれます。

引数	説明
SKIPTRANS <Transaction ID> [THREAD <n>] [FORCE]	(MySQL、Oracle、SQL Server、Sybase) Extract に、指定したトランザクションをスキップさせます。これにより、メモリーから現在のすべてのデータが削除され、後続のすべてのデータが無視されます。FORCE が使用されていない場合、確認プロンプトに応える必要があります。 ◆ <Transaction ID> は、トランザクション ID 番号です。ID 番号は、SHOWTRANS コマンドまたは Extract ランタイム・メッセージから取得してください。SKIPTRANS を使用するには、SHOWTRANS コマンドで表示されるトランザクション・リスト内の最も古いトランザクションを指定する必要があります。このコマンドは、古い順に他のトランザクションに対して繰り返し実行できます。 ◆ (Oracle) THREAD <n> では、重複するトランザクション ID がある場合に、Oracle RAC 環境でこのトランザクションを生成したスレッドを指定します。SKIPTRANS は、実際のスレッド番号ではなく、チェックポイント索引番号を指定します。正しいスレッドを指定するには、INFO EXTRACT <group> SHOWCH コマンドを発行してから、スキップするスレッド番号に対応する READ チェックポイント索引番号を指定します。詳細は、例を参照してください。 ◆ FORCE では、確認プロンプトをバイパスします。 SKIPTRANS を使用した後、SEND EXTRACT コマンドを FORCESTOP とともに発行する場合は、少なくとも 5 分待機してください。待機しないと、トランザクションはそのまま残ります。
STATUS	処理状況の詳細なステータス (現在の位置およびアクティビティを含む) を返します。 Current status 行には、次のような処理ステータス・メッセージが表示されます。 ◆ Delaying - 追加のデータを待機中 ◆ Suspended - 再開を待機中 ◆ Processing data - データを処理中 ◆ Starting initial load - 初期ロード・タスクを開始中 ◆ Processing source tables - 初期ロード・タスクのデータを処理中 ◆ Reading from data source - データ・ソース (ソース表、トランザクション・ログなど) から読取り中 ◆ Adding record to transaction list - ファイル・メモリー・トランザクション・リストにレコードを追加中 ◆ At EOF (end of file) - 処理するレコードなし

引数	説明
	異常終了イベント後の Extract リカバリ中は、上記のステータスに加え、次のようなステータスの注釈が表示されます。Extract はリカバリ中にログ読取り位置を継続的に変更するので、リカバリの進捗状況を追跡できます。 ◆ In recovery[1] - Extract は、トランザクション・ログのチェックポイントにリカバリ中です。 ◆ In recovery[2] - Extract は、チェックポイントからトレイルの末尾にリカバリ中です。 ◆ Recovery complete - リカバリが終了し、正常な処理が再開されます。 **DB2 LUW** ◆ グループ名およびプロセス ID ◆ 処理ステータス ◆ LSN ◆ タイムスタンプ **z/OS 上の DB2** ◆ グループ名およびプロセス ID ◆ 処理ステータス ◆ ログ RBA ◆ タイムスタンプ ◆ BSDS **Oracle および Oracle RAC** ◆ グループ名およびプロセス ID ◆ 処理ステータス ◆ REDO スレッド番号 (RAC のみ) ◆ REDO ログ順序番号 ◆ REDO ログの RBA ◆ タイムスタンプ ◆ SCN(RAC のみ) ◆ REDO ログ名 **SQL Server** ◆ グループ名およびプロセス ID ◆ 処理ステータス ◆ タイムスタンプ **Teradata、プライマリ Extract** ◆ グループ名およびプロセス ID ◆ 処理ステータス ◆ タイムスタンプ

引数	説明
	Teradata VAM ソート Extract ◆ 処理ステータス ◆ VAM トレイル順序番号 ◆ VAM トレイルの RBA ◆ タイムスタンプ ◆ VAM トレイル名 **すべてのデータベース、SOURCEISTABLE Extract タスク** ◆ Extract 名およびプロセス ID ◆ RMTTASK ◆ レコード番号 ◆ タイムスタンプ ◆ 表名

`STOP`	Extract を停止します。
	(WARNLONGTRANS パラメータに基づいて) 長時間におよぶトランザクションがある場合は、次のメッセージが表示されます。
	
	`Sending GETLAG request to EXTRACT CAPTPCC...`
	`There are open, long-running transactions. Before you stop Extract, make the archives containing data for those transactions available for when Extract restarts. To force Extract to stop, use the SEND EXTRACT <group>, FORCESTOP command.`
	`Oldest redo log file necessary to restart Extract is: Redo Thread 1, Redo Log Sequence Number 150, SCN 31248005, RBA 2912272.`

`TRACE[2] {<tracefile> | OFF}`	トレースの有効化と無効化を切り替えます。トレースは、処理のボトルネックを明らかにする情報を、指定したファイルに取得します。 ◆ TRACE では、ステップバイステップの処理情報を取得します。 ◆ TRACE2 では、具体的なステップでなく、コード・セグメントを特定します。 ◆ OFF ではトレースを無効化します。 トレースがすでに実行中の場合は、既存のトレース・ファイルが閉じ、<tracefile> で指定した新しいファイルにトレースが再開されます。 トレースによって重大な処理のボトルネックが明らかになった場合は、Oracle サポートに連絡してください。詳細は、http://support.oracle.com を参照してください。
`TRACE OFF <file name>`	指定したトレース・ファイルに対してのみ、トレースをオフにします。

引数	説明
TRACEINIT	トレースの統計を 0 にリセットし、統計の累積をもう一度開始します。このオプションは、履歴でなく現在の処理動作をトレースするときに使用します。
TRANLOGOPTIONS {PURGEORPHANEDTRANSACTIONS | NOPURGEORPHANEDTRANSACTIONS}	Oracle RAC に有効です。ノードに障害が発生し、Extract がロールバックを取得できないときに発生する孤立トランザクションのパージを有効化または無効化します。386 ページの「TRANLOGOPTIONS」も参照してください。
TRANLOGOPTIONS TRANSCLEANUPFREQUENCY <minutes>	Oracle RAC に有効です。Oracle GoldenGate が、孤立トランザクションのスキャン、再スキャンによる確認、および削除の一連の動作を実行する間隔 (分) を指定します。有効な値は 1 ～ 43200 分です。デフォルトは 10 分です。386 ページの「TRANLOGOPTIONS」も参照してください。

VAMMESSAGE "<Teradata command>"
 	
VAMMESSAGE {
"ARSTATS" |
"INCLUDELIST [filter]" |
"EXCLUDELIST [filter]"
}
 	
VAMMESSAGE "OPENTRANS"
	Extract が使用している取得 API にコマンドを送信します。
	<Teradata command> に指定可能な値 :
	◆ "control:terminate"
	レプリケーション・グループを停止します。Teradata のレプリケーション・グループを削除または変更する前に実行する必要があります。
	◆ "control:suspend"
	レプリケーション・グループを一時停止します。Oracle GoldenGate をアップグレードするときに使用できます。
	◆ "control:resume"
	一時停止したレプリケーション・グループを再開します。
	◆ "control:copy <database>.<table>"
	ソース・データベースからターゲット・データベースに表をコピーします。
	
	指定可能な SQL/MX コマンド :
	◆ "ARSTATS"
	TMF オーディット読取り統計を表示します。
	◆ "INCLUDELIST [filter]"
	Extract がオーディット・トレイル内でデータ・レコードを検出した、TABLE パラメータの選択基準に一致する表のリストを表示します。[filter] オプションでは、ワイルドカード・パターンを使用して返される表のリストをフィルタできます。

引数	説明
	◆ "EXCLUDELIST [filter]" Extract がオーディット・トレイル内でデータ・レコードを検出した、TABLE パラメータの選択基準に一致しない表のリストを表示します。[filter] オプションでは、ワイルドカード・パターンを使用して返される表のリストをフィルタできます。暗黙的に除外されている特定のシステム表は、常に除外表のリストに含まれます。 このモジュールは、GGSCI にレスポンスを返します。レスポンスは ERROR または OK で、レスポンス・メッセージとともに返されます。 **指定可能な SQL Server コマンド :** "OPENTRANS" オープンしているトランザクションのリスト (トランザクション ID、開始時刻、最初の LSN、含まれている操作数を含む) を出力します。

例 1 `SEND EXTRACT finance, ROLLOVER`

例 2 `SEND EXTRACT finance, STOP`

例 3 `SEND EXTRACT finance, VAMMESSAGE "control:suspend"`

例 4 `SEND EXTRACT finance, TRANLOGOPTIONS TRANSCLEANUPFREQUENCY 20`

例 5 次の例では、SKIPTRANS を説明します。スレッド 2 が Read Checkpoint #3 にあることを示す、次の SHOWCH 出力から開始します。

```
GGSCI> INFO <extract> SHOWCH
Read Checkpoint #3
Oracle RAC Redo Log
Startup Checkpoint (starting position in the data source):
Thread #: 2
Sequence #: 17560
RBA: 65070096
Timestamp: 2011-07-30 20:04:47.000000
SCN: 1461.3499051750 (6278446271206)
Redo File: RAC4REDO/sss11g/onlinelog/group_4.292.716481937
```

そのため、SKIPTRANS は SKIPTRANS <xid> THREAD 3 にする必要があります。

例 6 `SEND EXTRACT finance, SHOWTRANS COUNT 2`

例 7 次に SHOWTRANS の出力を示します。

SHOWTRANS のデフォルト出力

```
Oldest redo log file necessary to restart Extract is:
Redo Thread 1, Redo Log Sequence Number 148, SCN 30816254, RBA 17319664
------------------------------------------------------------
XID                : 5.15.52582
Items              : 30000
Extract            : JC108XT
Redo Thread        : 1
Start Time         : 2011-01-18:12:51:27
SCN                : 20634955
Redo Seq           : 103
Redo RBA           : 18616848
Status             : Running
------------------------------------------------------------
XID                : 7.14.48657
Items              : 30000
Extract            : JC108XT
Redo Thread        : 1
Start Time         : 2011-01-18:12:52:14
SCN                : 20635145
Redo Seq           : 103
Redo RBA           : 26499088
Status             : Running
```

TABULAR が有効なときの SHOWTRAN 出力 (右側を切り捨てた表示)

```
XID         Items  Extract   Redo Thread  Start Time
5.15.52582  30000  JC108XT   1            2011-01-18:12:52:14

SHOWTRANS FILE <name> DETAIL
Dumping transaction memory at 2011-01-21 13:36:54.
Record #1:
Header (140 bytes):
       0: 0000 0A4A 0000 FFFF 0000 0000 0057 6C10        ...J.........Wl.
      16: 02FF 3F50 FF38 7C40 0303 4141 414E 5A77        ..?P.8|@..AAANZw
      32: 4141 4641 4141 4B6F 4941 4144 0041 4141        AAFAAAKoIAAD.AAA
      48: 4E5A 7741 4146 4141 414B 6F49 4141 4400        NZwAAFAAAKoIAAD.
      64: 4141 414E 5A77 414A 2F41 4142 7A31 7741        AAANZwAJ/AABz1wA
      80: 4141 0041 4141 4141 4141 4141 4141 4141        AA.AAAAAAAAAAAA
      96: 4141 4141 4100 0000 0140 FF08 0003 0000        AAAAA....@......
     112: 0000 0000 0000 70FF 0108 FFFF 0001 4A53        ......p.......JS
     128: 554E 2E54 4355 5354 4D45 5200                  UN.TCUSTMER.

Data (93 bytes):
       0: 2C00 0400 0400 0000 0100 0200 0300 0000        ,...............
      16: 0000 0000 0800 0000 1800 0000 2000 0400        ............ ...
      32: 1000 0600 0200 0000 284A 414E 456C 6C6F        ........(JANEllo
      48: 6352 4F43 4B59 2046 4C59 4552 2049 4E43        cROCKY FLYER INC
      64: 2E44 454E 5645 5220 6E43 4F20 7365 7400        .DENVER nCO set.
      80: 0000 0000 0000 0C00 0000 0000 00              .............
```

SHOWTRANS の出力のサマリーを分析するときは、データベース上で現在実行中の (事前定義済バッファに収まる) すべてのトランザクションが表示されることを理解してください。Oracle GoldenGate の構成に含まれる表の操作が、今後別のトランザクションに追加されかどうかわからないため、Extract は構成されている表の操作を含むトランザクションだけでなく、オープンしているすべてのトランザクションを追跡する必要があります。

SHOWTRANS の Items フィールドの出力は、トランザクション内の操作数の合計ではなく、これまでに Oracle GoldenGate によって取得されたトランザクション内の操作数です。取得された操作に、構成に含まれる表への操作が含まれていない、またはその一部しか含まれていない場合は、Items の値は 0、または操作数の合計よりも少ない値になります。

Start Time フィールドには、トランザクション自体の実際の開始時刻ではなく、Oracle GoldenGate がトランザクションから抽出した最初の操作のタイムスタンプが表示されます。

注意 **Oracle GoldenGate ソフトウェアは継続的に拡張されるため、実際のコマンド出力はこの例とは若干異なる場合があります。**

START EXTRACT

START EXTRACT では、Extract プロセスを起動します。Extract が起動したことを確認するには、INFO EXTRACT または STATUS EXTRACT コマンドを使用します。

Extract は、特定の同期構成ではオペレーティング・システムのコマンドラインからも起動できます。目的に適した構成および起動方法の詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。

構文

```
START EXTRACT <group name>
```

引数	説明
`<group name>`	1 つの Extract グループ名、または複数のグループを指定するワイルドカード (*)。たとえば、「T*」と指定すると、名前が T から始まるすべての Extract グループが起動されます。

例

```
START EXTRACT finance
```

STATS EXTRACT

STATS EXTRACT では、1 つ以上の Extract グループの統計を表示します。出力には、Oracle GoldenGate 構成に含まれる DML および DDL 操作が含まれます。

最も正確な 1 秒当たりの処理操作数を取得するには、次のことを行います。

1. STATS EXTRACT コマンドを RESET オプションとともに発行します。
2. STATS EXTRACT REPORTRATE コマンドを発行します。LATEST STATISTICS フィールドに 1 秒当たりの操作数が表示されます。

図 5 LATEST および REPORTFETCH オプションを使用した出力の例

```
Sending STATS request to EXTRACT GGSEXT...

Start of Statistics at 2011-01-08 11:45:05.

DDL replication statistics (for all trails):
*** Total statistics since extract started     ***
    Operations                                    3.00
    Mapped operations                             3.00
    Unmapped operations                           0.00
    Default operations                            0.00
    Excluded operations                           0.00

Output to ./dirdat/aa:

Extracting from JDADD.EMPLOYEES to JDADD.EMPLOYEES:
*** Latest statistics since 2011-01-08 11:36:55 ***
    Total inserts                               176.00
    Total updates                                 0.00
    Total deletes                                40.00
    Total discards                                0.00
    Total operations                            216.00

Extracting from JDADD.DEPARTMENTS to JDADD.DEPARTMENTS:
*** Latest statistics since 2011-01-08 11:36:55 ***
No database operations have been performed.
End of Statistics.
```

注意 DB2 データベース上で実行された実際の DML 操作数は、Oracle GoldenGate がレポートした抽出 DML 操作数と一致しないことがあります。DB2 は、行を物理的に変更しない UPDATE 文のログを取らないので、Oracle GoldenGate はこのような文を検出することや、統計に反映させることができません。

Oracle GoldenGate が最大保護モードで構成されている Teradata ソース・システムで正確な統計を取得するには、プライマリ Extract ではなく VAM ソート Extract に STATS EXTRACT を発行します。プライマリ Extract には、ロールバックされるコミットされていないトランザクションの統計が含まれることがありますが、VAM ソート Extract は、コミットされていないトランザクションの統計のみレポートします。

構文

```
STATS EXTRACT <group name>
[, <statistic>]
[, TABLE <table>]
[, TOTALSONLY <table spec>]
[, REPORTFETCH | NOREPORTFETCH]
[, REPORTRATE <time units>]
[, ... ]
```

引数	説明
`<group name>`	1 つの Extract グループ名、または複数のグループを指定するワイルドカード (*)。たとえば、「T*」と指定すると、名前が T から始まるすべての Extract グループの統計が返されます。
`<statistic>`	表示する統計。それぞれをコンマで区切ることによって複数の統計を指定できます (例： STATS EXTRACT finance, TOTAL, DAILY)。 次の値が有効です。 TOTAL プロセス起動からの合計を表示します。 DAILY 現在の日付の開始からの合計を表示します。 HOURLY 現在の時間の開始からの合計を表示します。 LATEST 最後の RESET コマンド発行後の合計を表示します。 RESET LATEST 統計フィールドのカウンタをリセットします。
`TABLE <table>`	指定した表、またはワイルドカード (*) で指定した表グループのみの統計を表示します。
`TOTALSONLY <table spec>`	指定した表、またはワイルドカード (*) で指定した表グループの統計サマリーを表示します。
`REPORTFETCH | NOREPORTFETCH`	フェッチ操作についての統計を出力に含めるかどうかを制御します。デフォルトは NOREPORTFETCH です。336 ページの「STATOPTIONS」も参照してください。
`REPORTRATE <time units>`	統計を絶対値ではなく処理レートとして表示します。 次の値が有効です。 ◆ HR ◆ MIN ◆ SEC

例

次の例では、特定の表について、合計と 1 分当たりの統計を表示し、最新の統計をリセットし、フェッチ統計を出力します。

```
STATS EXTRACT finance, TOTAL, HOURLY, TABLE acct,
REPORTRATE MIN, RESET, REPORTFETCH
```

STATUS EXTRACT

STATUS EXTRACT では、Extract が実行中かどうかを確認します。RUNNING のステータスは、次のいずれかを意味します。

- アクティブ : 実行中および処理中 (または処理可能な) データ。これは、プロセス開始後のプロセスの通常の状態です。
- 一時停止 : プロセスは実行中ですが、EVENTACTIONS SUSPEND アクションにより一時停止しました。一時停止状態では、プロセスはアクティブではなく、データを処理できませんが、現在の実行の状態は保持され、GGSCI で RESUME コマンドを発行すると続行できます。INFO コマンドでの RBA は、一時停止アクションの前の最新のチェックポイント位置を表します。状態がアクティブか一時停止かを確認するには、SEND EXTRACT コマンドを STATUS オプションとともに発行します。

構文

```
STATUS EXTRACT <group name> [, TASKS | ALLPROCESSES]
```

引数	説明
`<group name>`	1 つの Extract グループ名、または複数のグループを指定するワイルドカード (*)。たとえば、「T*」と指定すると、名前が T から始まるすべての Extract グループのステータスが返されます
`TASKS`	Extract タスクのステータスのみを表示します。デフォルトでは、(ワイルドカードを使用せずに) 単一の Extract グループを指定している場合を除き、タスクは表示されません。
`ALLPROCESSES`	すべての Extract グループのステータス (タスクを含む) を表示します。

例 1

```
STATUS EXTRACT finance
```

例 2

```
STATUS EXTRACT fin*
```

STOP EXTRACT

STOP EXTRACT では、Extract を正常に停止します。このコマンドを実行すると、Extract の次の起動に備えて同期のステータスが保持され、Extract は Manager に自動的に起動されません。

STOP EXTRACT の発行時に長時間オープンしているトランザクションがある場合、Extract を再起動すると、このトランザクションに必要な最も古いトランザクション・ログ・ファイルを通知されることがあります。SEND EXTRACT の SHOWTRANS オプションを使用すると、このようなトランザクションの詳細とデータを表示できます。必要な場合には、SKIPTRANS または FORCETRANS オプションを使用し、トランザクションをスキップするか、コミット済トランザクションとしてトレイルに強制的に書き込みます。36 ページを参照してください。

構文

```
STOP EXTRACT <group name>
```

引数	説明
<group name>	1 つの Extract グループ名、または複数のグループを指定するワイルドカード (*)。たとえば、「T*」と指定すると、名前が T から始まるすべての Extract グループが停止されます。

例 STOP EXTRACT finance

UNREGISTER EXTRACT

UNREGISTER EXTRACT では、Extract グループの登録を Oracle データベースから削除します。UNREGISTER EXTRACT は、プライマリ Extract グループにのみ有効です。データ・ポンプ Extract には、これを使用しないでください。

Extract グループをデータベースに登録するには、REGISTER EXTRACT コマンドを使用します。

Extract をクラシック・キャプチャ・モードから統合キャプチャ・モードにアップグレードするには、ALTER EXTRACT コマンドを使用します。

構文
```
UNREGISTER EXTRACT <group name>
{DATABASE | LOGRETENTION}
```

引数	説明
<group name>	データベースから登録解除する Extract グループの名前。ワイルドカードは使用しないでください。このグループは、現在データベースに登録されている必要があります。
DATABASE	Extract グループに対して統合キャプチャ・モードを無効にします。このコマンドでは、Extract グループと同じ名前のデータベース取得 (マイニング) サーバーを削除します。Extract のキャプチャ・モードのサポートおよび構成の詳細は、Oracle データベースの Oracle GoldenGate のドキュメントを参照してください。 UNREGISTER EXTRACT を DATABASE とともに使用する前に、次を実行します。 1. STOP EXTRACT <group> コマンドを使用して、Extract を停止します。 2. dbms_goldengate_auth.grant_admin_privilege プロシージャで付与された権限で、DBLOGIN または MININGDBLOGIN コマンドを使用して、マイニング・データベースにログインします。マイニング・データベースがダウンストリーム・データベースである場合、MININGDBLOGIN を使用します。それ以外の場合は、DBLOGIN を使用します。 3. DELETE EXTRACT を使用して、Extract グループを削除します。

引数	説明
LOGRETENTION	指定した Extract グループのログ保持を無効にし、基盤の Oracle Streams 取得プロセスを削除します。この Extract グループで変更を取得しない場合のみ、UNREGISTER EXTRACT を LOGRETENTION とともに使用します。ログ保持機能は、TRANLOGOPTIONS パラメータの LOGRETENTION オプションで制御されます。 UNREGISTER EXTRACT を LOGRETENTION とともに使用する前に、STOP EXTRACT <group> コマンドを使用して Extract を停止します。次に、DBLOGIN コマンドを 78 ページの表 1 で示した権限で発行します。

例 1 `UNREGISTER EXTRACT sales LOGRETENTION`

例 2 `UNREGISTER EXTRACT sales DATABASE`

Replicat コマンド

Replicat コマンドでは、Replicat グループを作成および管理します。Replicat プロセスは、Extract プロセスによって抽出されたデータを読み取り、このデータをターゲット表に適用するか、ロード・アプリケーションなどの他のアプリケーションがこのデータを使用できるように準備を整えます。

コマンドの概要

ADD REPLICAT

ALTER REPLICAT

CLEANUP REPLICAT

DELETE REPLICAT

INFO REPLICAT

KILL REPLICAT

LAG REPLICAT

SEND REPLICAT

START REPLICAT

STATS REPLICAT

STATUS REPLICAT

STOP REPLICAT

ADD REPLICAT

ADD REPLICAT では、Replicat グループを作成します。SPECIALRUN が指定されている場合を除き、ADD REPLICAT は実行間の処理の継続性を維持するためにチェックポイントを作成します。Replicat グループを作成する前に、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を確認してください。

コマンドの制限

このコマンドでは、DESC オプションに入力するテキストを含め、すべてのキーワードおよび入力のサイズは、500 バイトを超えることはできません。

Oracle GoldenGate は、Oracle GoldenGate Manager インスタンス当たり、最大で 5,000 の同時 Extract および Replicat グループをサポートします。サポートされているレベルで、INFO、STATUS などの GGSCI コマンドですべてのグループを完全に制御および表示できます。Oracle GoldenGate では、環境を効果的に管理するために、Extract および Replicat グループの数 (合計) をデフォルト・レベルの 300 以下に保つことが推奨されます。

構文

```
ADD REPLICAT <group name>
{, SPECIALRUN |
    , EXTFILE <file name> |
    , EXTTRAIL <trail name>}
[, BEGIN {NOW | yyyy-mm-dd:hh:mm[:ss[.cccccc]]} |
    , EXTSEQNO <seqno>, EXTRBA <rba>]
[, CHECKPOINTTABLE <owner.table> | NODBCHECKPOINT]
[, PARAMS <parameter file>]
[, REPORT <report file>]
[, DESC "<description>"]
```

引数	説明
`<group name>`	Replicat グループ名。グループの命名規則については、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』を参照してください。
`SPECIALRUN`	Replicat 特別実行をタスクとして作成します。SPECIALRUN、EXTFILE または EXTTRAIL を指定する必要があります。Extract が SPECIALRUN モードの場合は、GGSCI の START REPLICAT コマンドで Replicat を起動しないでください。
`EXTFILE` `<full path name>`	Extract パラメータ・ファイルの RMTFILE で指定されている抽出ファイルの相対名または完全修飾名を指定します。
`EXTTRAIL` `<full path name>`	ADD RMTTRAIL または ADD EXTTRAIL コマンドで作成されたトレイルの相対名または完全修飾名を指定します。
`BEGIN <start point>`	トレイルの初期チェックポイントを定義します。 ◆ ADD REPLICAT でグループが作成されたときに変更のレプリケートを開始するには、NOW 引数を使用します。 ◆ 特定の時刻から変更の抽出を開始するには、日付および時刻フォーマットとして yyyy-mm-dd:hh:mm[:ss[.cccccc]] を使用します。

引数	説明
EXTSEQNO <seqno>	トレイル内のデータ処理開始ファイルの順序番号を指定します。順序番号を指定しますが、埋め込みのために 0 を使用しないでください。たとえば、トレイル・ファイルが c:\ggs\dirdat\aa000026 の場合は、EXTSEQNO 26 と指定します。 このオプションを使用しない場合、処理はデフォルトでトレイルの最初から開始されます。EXTSEQNO を使用するには、EXTRBA も使用する必要があります。このオプションを使用する前に Oracle サポートに連絡してください。詳細は、http://support.oracle.com を参照してください。
EXTRBA <rba>	EXTSEQNO で指定したトレイル・ファイル内の相対バイト・アドレスを指定します。このオプションを使用する前に Oracle サポートに連絡してください。詳細は、http://support.oracle.com を参照してください。
CHECKPOINTTABLE <owner.table>	この Replicat グループが、データベース内の指定した表にチェックポイントを書き込むように指定します。所有者と表名を指定します (例 : hr.hr_checkpoint)。この引数は、GLOBALS ファイルのすべてのデフォルトの CHECKPOINTTABLE 指定よりも優先されます。ADD CHECKPOINTTABLE コマンドを使用して、この表を追加する必要があります。
NODBCHECKPOINT	この Replicat グループがチェックポイント表にチェックポイントを書き込まないように指定します。この引数は、GLOBALS ファイルのすべてのデフォルトの CHECKPOINTTABLE 指定よりも優先されます。この引数は、作成中の Replicat グループとともにチェックポイントを使用しないときに指定する必要があります。
PARAMS <parameter file>	パラメータ・ファイルの保管場所を、Oracle GoldenGate ディレクトリ内のデフォルトの dirprm 以外に指定します。完全修飾パス名を指定します。
REPORT <report file>	処理レポート・ファイルの保管場所の完全パス名を、Oracle GoldenGate ディレクトリ内のデフォルトの dirrpt 以外に指定します。
DESC "<description>"	グループの説明 ("Loads account_tab on Serv2" など) を指定します。この説明は引用符で囲む必要があります。短縮キーワード DESC または完全なキーワード DESCRIPTION を使用できます。

例 `ADD REPLICAT sales, EXTTRAIL dirdat\rt`

ALTER REPLICAT

ALTER REPLICAT では、ADD REPLICAT コマンドで作成した Replicat グループの属性を変更します。このコマンドを使用する前に、STOP REPLICAT <group name> コマンドを発行して Replicat を停止してください。

構文 `ALTER REPLICAT <group name> , <option> [, ...]`

引数	説明
<group name>	変更する Replicat グループの名前。
<option>	ADD REPLICAT のオプションの 1 つを指定します。CHECKPOINT および NODBCHECKPOINT オプションを除き、ADD REPLICAT コマンドで構成した説明または任意のサービス・オプションを変更できます。

例 1 `ALTER REPLICAT finance, EXTSEQNO 53`

例 2 `ALTER REPLICAT finance, EXTRBA 0`

例 3 `ALTER REPLICAT finance, BEGIN 2011-01-07:08:00:00`

CLEANUP REPLICAT

CLEANUP REPLICAT では、特定の Replicat グループの実行履歴を削除します。このコマンドでは、最後の実行レコードがそのまま保持されるので、Replicat は停止した場所から処理を再開できます。

このコマンドを使用する前に、STOP REPLICAT <group name> コマンドを発行して Replicat を停止してください。

構文 `CLEANUP REPLICAT <group name> [, SAVE <count>]`

引数	説明
<group name>	1 つの Replicat グループ名、または複数のグループを指定するワイルドカード (*)。たとえば、「T*」と指定すると、名前が T から始まるすべての Replicat グループの実行履歴が削除されます。
SAVE <count>	指定した数の最新のレコードを削除から除外します。

例 1 次の例では、最新レコード以外のすべてのレコードを削除します。

```
CLEANUP REPLICAT finance
```

例 2 次の例では、最新の 5 レコード以外のすべてのレコードを削除します。

```
CLEANUP EXTRACT *, SAVE 5
```

DELETE REPLICAT

DELETE REPLICAT では、Replicat グループを削除します。このコマンドを実行すると、チェックポイント・ファイルは削除されますが、パラメータ・ファイルはそのまま残ります。グループの削除後、必要に応じてグループをもう一度作成することも、パラメータ・ファイルを削除することもできます。このコマンドを実行すると、(他のプロセスがファイルを読み取っていない場合) 削除されたグループが使用していたチェックポイントが削除されるので、Manager によってトレイル・ファイルをパージできます。

DELETE REPLICAT を使用する前に、次のことを行います。

1. Replicat を停止します。

   ```
   STOP REPLICAT <group name>
   ```

2. このグループがデータベースのチェックポイント表を使用している場合、チェックポイント表からチェックポイントを削除できるように、DBLOGIN コマンドを使用してデータベースにログインします。

構文 `DELETE REPLICAT <group name> [!]`

引数	説明
`<group name>`	1 つの Replicat グループ名、または複数のグループを指定するワイルドカード (*)。たとえば、「T*」と指定すると、名前が T から始まるすべての Replicat グループが削除されます。
!	このオプションは、データベースのチェックポイント表からではなく、ディスク上のチェックポイント・ファイルからこのグループのチェックポイントを削除するときに使用します。このオプションを使用すると、ワイルドカードで複数のグループを指定するときに発生するプロンプトを無視できます。

例 `DELETE REPLICAT finance`

INFO REPLICAT

INFO REPLICAT では、Replicat グループの処理履歴を取得します。このコマンド出力には次の情報が含まれます。

- Replicat のステータス (STARTING、RUNNING、STOPPED または ABENDED)
- おおよその Replicat ラグ
- Replicat が読み取るトレイル
- トレイル内のチェックポイントを含む Replicat 実行履歴
- Replicat 環境に関する情報

TASKS または ALLPROCESSES 引数なしの基本コマンドでは、オンライン (継続的な)Replicat プロセスの情報のみを表示します。タスクは除外されます。

INFO REPLICAT を発行すると、Replicat は停止または実行中になります。実行中のプロセスの場合、RUNNING のステータスは、次のいずれかを意味します。

- アクティブ : 実行中および処理中 (または処理可能な) データ。これは、プロセス開始後のプロセスの通常の状態です。
- 一時停止 : プロセスは実行中ですが、EVENTACTIONS SUSPEND アクションにより一時停止しました。一時停止状態では、プロセスはアクティブではなく、データを処理できませんが、現在の実行の状態は保持され、GGSCI で RESUME コマンドを発行すると続行できます。INFO コマンドでの RBA は、一時停止アクションの前の最新のチェックポイント位置を表します。状態がアクティブか一時停止かを確認するには、SEND REPLICAT コマンドを STATUS オプションとともに発行します。

ラグについて

Checkpoint Lag は、最新のチェックポイントがトレイルに書き込まれた時点での遅延 (秒) です。次に例を示します。

- 現在の時刻 = 15:00:00
- 最新のチェックポイント = 14:59:00

- 最新の処理レコードのタイムスタンプ = 14:58:00

この場合、ラグは 00:01:00(1 分、14:58 と 14:59 の差異) とレポートされます。

ラグの値が UNKNOWN の場合は、Replicat は実行しているがレコードをまだ処理していないか、(タイム・ゾーンの差ではなくクロックが正確でないために) ソース・システムのクロックがターゲット・システムのクロックよりも進んでいることを示します。より正確なラグ情報を表示するには、LAG REPLICAT を使用します (61 ページを参照してください)。

詳細情報の表示

詳細情報を表示するには、DETAIL オプションを使用します。次に出力の例を示します。

図 6 詳細な INFO REPLICAT 出力

```
REPLICAT   DELTPCC          Last Started 2011-01-21 11:40 Status RUNNING
Checkpoint Lag              00:00:00 (updated 232:39:41 ago)
Log Read Checkpoint File    C:\GGS\DIRDAT\RT000000
                            2011-01-21 18:54:33.000000 RBA 4735245

Extract Source                          Begin             End
C:\GGS\DIRDAT\RT000000     2011-01-21 18:54           2011-01-21 18:54
C:\GGS\DIRDAT\RT000000     * Initialized *            2011-01-21 18:54

Current directory          C:\GGS
Report file                C:\GGS\dirrpt\DELTPCC.rpt
Parameter file             dirprm\DELTPCC.prm
Checkpoint file            C:\GGS\dirchk\DELTPCC.cpr
Checkpoint table           GG.CHECKPT
Process file               C:\GGS\dirpcs\DELTPCC.pcr
Error log                  C:\GGS\ggserr.log
```

チェックポイントの表示

Replicat は、最新の読取り位置をマーク付けするために、トレイル・ファイルにチェックポイントを作成します。プロセス・チェックポイントを表示するには、SHOWCH オプションを使用します。基本コマンドでは、現在のチェックポイントを表示します。特定の番号の過去のチェックポイントを表示するには、番号の値を SHOWCH エントリに続いて入力します。

図 7 INFO REPLICAT、SHOWCH

```
REPLICAT   JC108RP   Last Started 2011-01-12 13:10   Status RUNNING
Checkpoint Lag       00:00:00 (updated 111:46:54 ago)
Log Read Checkpoint  File ./dirdat/eh000000
                     First Record  RBA 3702915
Current Checkpoint Detail:
    Read Checkpoint #1
    GGS Log Trail
    Startup Checkpoint(starting position in data source):
        Sequence #: 0
        RBA: 3702915
        Timestamp: Not Available
        Extract Trail: ./dirdat/eh
    Current Checkpoint (position of last record read in the data source):
        Sequence #: 0
        RBA: 3702915
        Timestamp: Not Available
        Extract Trail: ./dirdat/eh
    Header:
        Version = 2
        Record Source = A
        Type = 1
        # Input Checkpoints = 1
        # Output Checkpoints = 0
    File Information:
        Block Size = 2048
        Max Blocks = 100
        Record Length = 2048
        Current Offset = 0
    Configuration:
        Data Source = 0
        Transaction Integrity = -1
        Task Type = 0
    Status:
        Start Time = 2011-01-12 13:10:13
        Last Update Time = 2011-01-12 21:23:31
        Stop Status = A
        Last Result = 400
```

Replicat チェックポイントについて

Extract はトレイルにチェックポイントを作成します。

開始チェックポイント

開始チェックポイントは、プロセスの起動時にトレイルに作成される最初のチェックポイントです。この統計は次の情報で構成されます。

- Sequence #: チェックポイントが書き込まれたトレイル・ファイルの順序番号。
- RBA: チェックポイントが作成されたレコードの相対バイト・アドレス。
- Timestamp: チェックポイントの作成時のレコードのタイムスタンプ。
- Extract Trail: トレイルの相対パス名。

現在のチェックポイント

現在のチェックポイントは、Replicat が読み取った最新レコードのトレイル内の位置です。この値は、サマリー、およびオプションなしの基本の INFO REPLICAT コマンドに表示される Log Read Checkpoint 統計と同じになるはずです。この統計のフィールドは、Startup Checkpoint に含まれるフィールドと同じです。

その他の SHOWCH 情報

SHOWCH 出力の最後に表示される Header、File Information、Configuration および Status 統計は、Oracle サポートのアナリストが使用する情報です。これらの統計には、サポート・ケースの解決に有益な内部情報が含まれています。

構文

```
INFO REPLICAT <group name>
[, DETAIL]
[, SHOWCH [<n>]]
[, TASKS | ALLPROCESSES]
```

引数	説明
`<group name>`	1 つの Replicat グループ名、または複数のグループを指定するワイルドカード (*)。たとえば、「T*」と指定すると、名前が T から始まるすべての Replicat グループが表示されます。
`DETAIL`	詳細情報を表示します。
`SHOWCH`	チェックポイント・ファイルに記録済のチェックポイント、チェックポイント表に記録済のチェックポイント、使用中のチェックポイントなど、現在のチェックポイントの詳細を表示します。データベース・チェックポイントの表示には、表名、ハッシュ・キー (一意の識別子)、作成時のタイムスタンプが含まれます。 <n> には、現在のチェックポイントに加えて表示する過去のチェックポイントの番号を指定します。
`TASKS`	Replicat タスクのみを表示します。ワイルドカード引数で指定したタスクは、INFO REPLICAT では表示されません。
`ALLPROCESSES`	タスクを含むすべての Replicat グループを表示します。

例 1
```
INFO REPLICAT *, DETAIL, ALLPROCESSES
```

例 2
```
INFO REPLICAT *, TASKS
```

例 3
```
INFO REPLICAT finance, SHOWCH
```

KILL REPLICAT

KILL REPLICAT では、Replicat グループを中断します。プロセスを中断すると、最新のチェックポイントが記録され、現在のトランザクションはデータベースによってロールバックされます。したがって、データを失うことなくプロセスを再開できます。中断された Replicat プロセスは、Manager プロセスによって再起動されません。このコマンドは、STOP REPLICAT コマンドで Replicat を正常に停止できない場合にのみ使用してください。

構文　　`KILL REPLICAT <group name>`

引数	説明
`<group name>`	1 つの Replicat グループ名、または複数のグループを指定するワイルドカード (*)。たとえば、「T*」と指定すると、名前が T から始まるすべての Replicat グループが中断されます。

例　　`KILL REPLICAT finance`

LAG REPLICAT

LAG REPLICAT では、Replicat とトレイル間の正確なラグ・タイムを確認します。LAG REPLICAT は、チェックポイントの位置を読み取るのでなく、Replicat と直接通信するので、INFO REPLICAT よりも正確にラグ・タイムを計算できます。

Replicat のラグについて

Replicat のラグとは、(システムクロックに基づく)Replicat が最後のレコードを処理した時刻と、トレイル内のそのレコードのタイムスタンプとの差 (秒) です。

構文　　`LAG REPLICAT <group name>`

引数	説明
`<group name>`	1 つの Replicat グループ名、または複数のグループを指定するワイルドカード (*)。たとえば、「T*」と指定すると、名前が T から始まるすべての Replicat グループのラグが表示されます。

例 1　　`LAG REPLICAT *`

例 2　　`LAG REPLICAT *fin*`

SEND REPLICAT

SEND REPLICAT では、起動中または実行中の Replicat プロセスと通信します。このリクエストは、Replicat がユーザーからのコマンドを受け入れる準備ができるとすぐに処理されます。

構文

```
SEND REPLICAT <group name>,{
FORCESTOP |
GETLAG |
HANDLECOLLISIONS [<table spec>] |
NOHANDLECOLLISIONS [<table spec>] |
REPORT [HANDLECOLLISIONS [<table spec>]] |
RESUME |
STATUS |
STOP |
TRACE[2] [DDLINCLUDE | DDL[ONLY]] [FILE] <file name> |
TRACE[2] OFF |
TRACE OFF <file name> |
TRACEINIT
}
```

引数	説明
`<group name>`	Replicat グループ名。この Replicat が実行中でない場合はエラーが返されます。
`FORCESTOP`	すべての通知をバイパスし、Replicat を強制的に停止します。このコマンドは、アクティブなトランザクションをロールバックし、このプロセスを即座に停止します。
`GETLAG`	Replicat とトレイル間の正確なラグ・タイムを表示します。ラグ・タイムは、Replicat によって最後のレコードが処理された時刻と、トレイル内のそのレコードのタイムスタンプとの差 (秒) です。この結果は、LAG REPLICAT と同じです。
`HANDLECOLLISIONS [<table spec>]`	HANDLECOLLISIONS パラメータを有効にします。このオプションを使用するかわりに、Replicat パラメータ・ファイルに HANDLECOLLISIONS パラメータを指定することもできます。HANDLECOLLISIONS は、ソース・データベースがアクティブな間に初期データ・ロードを実行するときに発生するエラーを自動処理するために使用します。初期ロードを完了し、ターゲット表にオンライン・データ変更を適用したら、(SEND REPLICAT またはパラメータ・ファイルからこのパラメータを削除することによって) 必ず HANDLECOLLISIONS を無効にしてください。 <table spec> では、HANDLECOLLISIONS の適用を特定のターゲット表、または標準ワイルドカード (*) で指定したターゲット表グループに制限します。
`NOHANDLECOLLISIONS [<table spec>]`	HANDLECOLLISIONS パラメータを無効にします。ただしパラメータ・ファイルからは削除しません。次に Replicat を起動するときに HANDLECOLLISIONS を有効化しないようにするには、パラメータ・ファイルからこのパラメータを削除する必要があります。 <table spec> では、NOHANDLECOLLISIONS の適用を特定のターゲット表、または標準ワイルドカード (*) で指定したターゲット表グループに制限します。

引数	説明
REPORT [HANDLECOLLISIONS [<table spec>]]	Extract レポート・ファイルに一時的な統計レポートを生成します。RESETREPORTSTATS | NORESETREPORTSTATS とともに使用する場合、表示される統計は、STATOPTIONS パラメータの構成によって異なります。368 ページを参照してください。 HANDLECOLLISIONS では、HANDLECOLLISIONS が有効な表を表示します。<table spec> では、出力を特定のターゲット表、または標準ワイルドカード (*) で指定したターゲット表グループに制限します。
RESUME	EVENTACTIONS SUSPEND イベントによって一時停止されたプロセスを再開 (アクティブに) します。プロセスは、一時停止された位置から通常の処理を再開します。
STATUS	トレイル内の現在の位置、および現在のトランザクションに関する情報を返します。次のフィールドが出力されます。 ◆ 処理ステータス ◆ トレイル・ファイル内の位置 ◆ トレイル連続番号 ◆ トレイルの RBA ◆ トレイル名 次のような処理ステータス・メッセージが表示されます。 ◆ Delaying - 追加のデータを待機中 ◆ Suspended - 再開を待機中 ◆ Waiting on deferred apply –DEFERAPPLYINTERVAL パラメータ基づいて処理待機中 ◆ Processing data - データを処理中 ◆ Skipping current transaction - START REPLICAT と SKIPTRANSACTION を使用している場合 ◆ Searching for START ATCSN <csn> - START REPLICAT と ATCSN を使用している場合 ◆ Searching for START AFTERCSN <csn> - START REPLICAT と AFTERCSN を使用している場合 ◆ Performing transaction timeout recovery - 現在の未完了のトランザクションを中止し、新しいトランザクション開始のために再配置中 (TRANSACTIONTIMEOUT パラメータを参照してください) ◆ Waiting for data at logical EOF after transaction timeout recovery - TRANSACTIONTIMEOUT 終了後に未完了のソース・トランザクションの残りの部分の受信を待機中 ◆ At EOF (end of file) - 処理するレコードなし
STOP	Replicat を正常に停止します。

引数	説明

TRACE[2] [DDLINCLUDE | DDL[ONLY]] [FILE] <tracefile>	トレースの有効化と無効化を切り替えます。トレースは、処理のボトルネックを明らかにする情報を、指定したファイルに取得します。
	◆ TRACE では、ステップバイステップの処理情報を取得します。
	◆ TRACE2 では、具体的なステップでなく、コード・セグメントを特定します。
	トレースがすでに実行中の場合は、既存のトレース・ファイルが閉じ、<tracefile> で指定したファイルにトレースが再開されます。
	トレースによって重大な処理のボトルネックが明らかになった場合は、Oracle サポートに連絡してください。詳細は、http://support.oracle.com を参照してください。
	トレースは、Replicat パラメータの TRACE および TRACE2 でも有効化できます。
	
	DDLINCLUDE | DDLONLY
	(Replicat のみ) DDL トレースを有効化し、トレース・レポートに DDL トレースを取得する方法を指定します。
	◆ DDLINCLUDE では、トランザクション・データ処理の通常のトレースに加えて DDL トレースを取得します。
	◆ DDL[ONLY] では、トランザクション・データ処理の通常のトレースを除外し、DDL のみをトレースします。このオプションは、DDL に短縮できます。
	
	[FILE] <file name>
	Oracle GoldenGate のトレース情報の記録先ファイルの相対名または完全修飾名。FILE キーワードはオプションですが、次の例のように、ファイル名の後に他のオプションを指定する場合は使用する必要があります。
	`SEND REPLICAT <group> TRACE FILE <file name> DDLINCLUDE`
	次の例のように、ファイル名の後に他のオプションを指定しない場合は、FILE キーワードを省略できます。
	`SEND REPLICAT <group> TRACE DDLINCLUDE <file name>`

TRACE[2] OFF	トレースを無効にします。
TRACE OFF <file name>	指定したトレース・ファイルに対してのみ、トレースをオフにします。このオプションは、複数の EVENTACTIONS 文によって複数のトレース・ファイルを使用できる、EVENTACTIONS 機能をサポートしています。
TRACEINIT	トレースの統計を 0 にリセットし、統計の累積をもう一度開始します。このオプションは、履歴でなく現在の処理動作をトレースするときに使用します。

例 1 `SEND REPLICAT finance, HANDLECOLLISIONS`

例 2 `SEND REPLICAT finance, REPORT HANDLECOLLISIONS fin_*`

例 3 `SEND REPLICAT finance, GETLAG`

START REPLICAT

START REPLICAT では、Replicat を起動します。Replicat が起動したことを確認するには、INFO REPLICAT または STATUS REPLICAT コマンドを使用します。

Replicat の処理開始オプションについて

通常の開始位置

START REPLICAT(オプションなし) では、Replicat はデータ整合性を維持するために次のいずれかのポイントから処理を開始します。

- 正常または異常終了後: 前回の実行で処理されなかった最後のトランザクション(現在のチェックポイントが示す位置)
- グループ作成後の最初の起動時 : アクティブなトレイル・ファイルの初め (順序番号 0、RBA0)

別の開始位置

START REPLICAT の SKIPTRANSACTION、ATCSN および AFTERCSN オプションを使用すると、Replicat は通常の開始位置以外のトレイル内のトランザクションから処理を開始します。これらのオプションは、次の目的で使用します。

- Replicat のトレイル内の処理進行を妨げるエラー以降の論理リカバリ位置を指定する。問題のあるトランザクションをスキップする位置に Replicat を配置できますが、このようなトランザクションのデータがターゲットに適用されないことを認識して使用してください。
- 初期ロード実行中にレプリケートされたトランザクション変更の適用を開始する位置を指定する。トランザクションによってデータベース内のデータが変更されるたびに、データベース・エンジンはその時点でのデータの状態を表す変更識別子を割り当てます。Oracle GoldenGate の用語で*コミット順序番号* (CSN) と呼ばれるこの識別子により、データベースは様々なトランザクションにわたるデータの状態の変化を追跡できます。バックアップ完了に対応する CNS を知っていれば、Replicat を起動し、レプリケートされたトランザクションをその位置以降から適用できます。したがって Replicat は、バックアップ以前の状態でレプリケートされた変更をバイパスできます。古いデータ変更をスキップすることで、重複レコードおよび行方不明レコードのエラーを回避できます。

> **注意** トランザクションをスキップするか、特定の CSN またはそれ以降から開始する場合は、適切なトランザクション・レコードに到達するまでに読み取る必要があるデータ量に応じて、Replicat の処理開始が通常より遅くなることがあります。処理開始の進捗状況を表示するには、SEND REPLICAT コマンドと STATUS オプションを使用します。

コマンドラインからの Replicat の起動

Replicat は、特定の同期構成ではオペレーティング・システムのコマンドラインからも起動できます。目的に適した構成および起動方法の詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。

構文

```
START REPLICAT <group name>
[SKIPTRANSACTION | ATCSN <csn> | AFTERCSN <csn>]
```

引数	説明
<group name>	1 つの Replicat グループ名、または複数のグループを指定するワイルドカード (*)。たとえば、「T*」と指定すると、名前が T から始まるすべての Replicat グループが起動されます。
SKIPTRANSACTION	Replicat に、起動時の位置以降の 1 番目のトランザクションをスキップさせます。この 1 番目のトランザクション内の操作は、すべて除外されます。 この Replicat に対して MAXTRANSOPS パラメータも使用している場合は、トランザクションの途中からトレイル・ファイルの読取りが開始されることがあります。この場合、この不完全なトランザクションの残りの部分はスキップされ、Replicat はファイル内の次の開始トランザクション・レコードから通常の処理を再開します。DISCARDFILE パラメータを使用している場合は、スキップされたレコードは破棄ファイルに書き込まれます。このパラメータを使用していない場合は、次のようなメッセージがレポート・ファイルに書き込まれます。 `User requested START SKIPTRANSACTION. The current transaction will be skipped. Transaction ID <txid>, position Seqno <seqno>, RBA <rba>` **制約事項:** ◆ Replicat の読取り先トレイルが、(チェックポイントを使用した) オンライン変更同期構成に含まれている場合にのみ有効です。タスクタイプの初期ロード (ADD REPLICAT で SPECIALRUN を使用する場合) では有効ではありません。
ATCSN <csn> | AFTERCSN <csn>	◆ ATCSN <csn> では、指定したコミット順序番号 (CSN) を持つ開始トランザクション・インジケータを見つけるまで、Replicat にトレイル内のトランザクションをスキップさせます。このトランザクションおよび後続のトランザクションがターゲットに適用されます。指定した CSN より小さい番号を持つトランザクションはすべてスキップされます。 ◆ AFTERCSN <csn> では、指定した CSN を持つトランザクション以降の最初のトランザクションを見つけるまで、Replicat にトレイル内のトランザクションをスキップさせます。開始トランザクション・レコードに、指定した CSN 以下の番号が含まれているトランザクションは、すべてスキップされます。 <csn> については、付録「コミット順序番号について」を参照してください。CSN は、データベースにネイティブなフォーマットで指定する必要があり、それ以外のフォーマットの場合、Replicat は異常終了し、レポート・ファイルにメッセージを書き込みます。 使用に適した CSN を確認するには、GGSCI の VIEW REPORT <group> コマンドで Replicat レポート・ファイルを表示します。正しい CSN の確認にさらに詳しい調査が必要な場合、経験が豊富な Oracle GoldenGate ユーザーは Logdump ユーティリティを使用できます。Logdump 使用の詳細は、『Oracle GoldenGate *Windows and UNIX トラブルシューティングおよびチューニング・ガイド*』を参照してください。 ATCSN または AFTERCSN を使用する場合は、次のようなメッセージがレポート・ファイルに書き込まれます。 `User requested start at commit sequence number (CSN) <csn-string>` または `User requested start at commit sequence number (CSN) <csn-string>`

引数	説明
	制約事項 : ◆ Replicat の読取り先トレイルが、(チェックポイントを使用した) オンライン変更同期構成に含まれている場合にのみ有効です。タスクタイプの初期ロード (ADD REPLICAT で SPECIALRUN を使用する場合) では有効ではありません。特定の CSN またはその直後からの開始をサポートするには、トレイルは、CSN がファイル・ヘッダーに格納される Oracle GoldenGate リリース 10.0.0 以降のものである必要があります。それ以前のリリースのトレイルの場合に、AFTERCSN を指定して Replicat を起動すると、Replicat は異常終了し、トレイル・フォーマットがサポートされていないことを通知するエラーをレポートに書き込みます。

注意 CSN で指定されたレコードが見つからない場合、Replicat はチェックポイントを発行するので、次のチェックポイントより前に発生する再開プロセスは、リクエストされた CSN よりも前の地点ではなく、リクエストされた場所から処理を開始できます。

例 1
```
START REPLICAT finance
```

例 2 次の例では、Replicat を起動し、Oracle 固有の CSN から処理を開始します。
```
START REPLICAT finance, ATCSN 6488359
```

例 3 次の例では、Replicat を起動し、SQL Server 固有の CSN から処理を開始します。
```
START REPLICAT finance, AFTERCSN 0X000004D2:0000162E:0009
```

STATS REPLICAT

STATS REPLICAT では、1 つ以上の Replicat グループの統計を表示します。

構文
```
STATS REPLICAT <group name>
[, <statistic>]
[, TABLE <table>]
[, TOTALSONLY <table spec>]
[, REPORTCDR]
[, REPORTDETAIL | NOREPORTDETAIL]
[, REPORTRATE <time units>]
[, ... ]
```

引数	説明
<group name>	1 つの Replicat グループ名、または複数のグループを指定するワイルドカード (*)。たとえば、「T*」と指定すると、名前が T から始まるすべての Replicat グループの統計が表示されます。
<statistic>	表示する統計。コンマで区切ることによって複数の統計を指定できます (例 : STATS REPLICAT finance, TOTAL, DAILY)。

引数	説明
	有効な値 : TOTAL プロセス起動からの合計を表示します。 DAILY 現在の日付の開始からの合計を表示します。 HOURLY 現在の時間の開始からの合計を表示します。 LATEST 最後の RESET コマンド発行後の合計を表示します。 RESET LATEST 統計フィールドのカウンタをリセットします。
`TABLE <table>`	指定した表、またはワイルドカード (*) で指定した表グループのみの統計を表示します。
`TOTALSONLY <table spec>`	指定した表、またはワイルドカード (*) で指定した表グループの統計サマリーを表示します。
`REPORTCDR`	競合の検出および解決のための統計を表示します。次の統計があります。 ◆ 合計 CDR 競合 ◆ 成功した CDR 解決 ◆ 失敗した CDR 解決 ◆ CDR INSERTROWEXISTS 競合 ◆ CDR UPDATEROWEXISTS 競合 ◆ CDR DELROWEXISTS 競合 ◆ CDR DELROWMISSING 競合
`REPORTDETAIL | NOREPORTDETAIL`	衝突エラーが原因でレプリケートされなかった操作を出力に含めるかどうかを制御します。出力を有効にすると、このような操作が通常の統計 (実行された INSERT、UPDATE および DELETE 操作)、および詳細な表示の統計でレポートされます。たとえば、10 レコードが INSERT 操作で、キーの重複が原因でこれらすべてのレコードが無視された場合、レポートには INSERT 操作数 10、衝突が原因で破棄された操作数 10 と出力されます。デフォルトは REPORTDETAIL です。336 ページの「STATOPTIONS」も参照してください。
`REPORTRATE <time units>`	統計を絶対値ではなく処理レートとして表示します。 次の値が有効です。 ◆ HR ◆ MIN ◆ SEC

例 次の例では、特定の表について、プロセス起動後および現在の時間の開始後の 1 分当たりの統計を表示しし、最新の統計をリセットします。破棄された操作の統計はレポートされません。

```
STATS REPLICAT finance, TOTAL, HOURLY, TABLE acct,
REPORTRATE MIN, RESET, NOREPORTDETAIL
```

STATUS REPLICAT

STATUS REPLICAT では、Replicat が実行中かどうかを確認します。RUNNING のステータスは、次のいずれかを意味します。

- アクティブ : 実行中および処理中 (または処理可能な) データ。これは、プロセス開始後のプロセスの通常の状態です。
- 一時停止 : プロセスは実行中ですが、EVENTACTIONS SUSPEND アクションにより一時停止しました。一時停止状態では、プロセスはアクティブではなく、データを処理できませんが、現在の実行の状態は保持され、GGSCI で RESUME コマンドを発行すると続行できます。INFO コマンドでの RBA は、一時停止アクションの前の最新のチェックポイント位置を表します。状態がアクティブか一時停止かを確認するには、SEND REPLICAT コマンドを STATUS オプションとともに発行します。

構文

```
STATUS REPLICAT <group name>
[, TASKS]
[, ALLPROCESSES]
```

引数	説明
`<group name>`	1 つの Replicat グループ名、または複数のグループを指定するワイルドカード (*)。たとえば、「T*」と指定すると、名前が T から始まるすべての Replicat グループのステータスが表示されます。
`TASKS`	Replicat タスクのステータスのみを表示します。デフォルトでは、(ワイルドカードを使用せずに) 単一の Replicat グループを指定している場合を除き、タスクは表示されません。
`ALLPROCESSES`	すべての Replicat グループのステータス (タスクを含む) を表示します。

例 1

```
STATUS REPLICAT finance
```

例 2

```
STATUS REPLICAT fin*
```

STOP REPLICAT

STOP REPLICAT では、Replicat を正常に停止します。このコマンドを実行する場合は、Replicat の次の起動に備えて同期のステータスが保持され、Replicat は Manager に自動的に起動されません。

構文

```
STOP REPLICAT <group name> [!]
```

引数	説明
`<group name>`	1 つの Replicat グループ名、または複数のグループを指定するワイルドカード (*)。たとえば、「T*」と指定すると、名前が T から始まるすべての Replicat グループが停止されます。

引数	説明
!	(感嘆符)Replicat を即座に停止します。トランザクションが中止され、プロセスは終了します。

例 `STOP REPLICAT finance`

ER コマンド

ER コマンドでは、複数の Extract および Replicat グループを 1 つのユニットとして制御します。ワイルドカードとともに使用して、その条件を満たすすべての Extract および Replicat グループにコマンドを適用します。

構文 `<command> ER <group wildcard specification>`

引数	説明

`<command>`	次のいずれかを指定できます。
	`INFO`
	`KILL`
	`LAG`
	`SEND`
	`START`
	`STATS`
	`STATUS`
	`STOP`
	これらのコマンドの説明およびオプションのパラメータの詳細は、この章に記載されている同等の Extract または Replicat コマンドを参照してください。

`<group wildcard specification>`	コマンドを適用するグループのワイルドカード指定。Oracle GoldenGate は、自動的に内部記憶域を拡大し、最大で 100,000 のワイルドカード・エントリを追跡します。

例 次の例では、名前に X が含まれる Extract および Replicat グループを起動した後、このグループを停止します。

```
GGSCI (ggs3) > START ER *X*
GGSCI (ggs3) > STOP ER *X*
```

トレイル・コマンド

トレイル・コマンドでは、Oracle GoldenGate トレイルを作成および管理します。トレイルは、ターゲットの場所に適用されるまで、Oracle GoldenGate が抽出データを一時的に保管する、ディスク上の一連のファイルです。

コマンドの概要

ADD EXTTRAIL

ADD EXTTRAIL

ADD EXTTRAIL では、ローカル・システム上にオンライン処理用のトレイルを作成し、次のことも行います。

- このトレイルに Extract グループを関連付ける
- 最大ファイル・サイズを指定する

構文

```
ADD EXTTRAIL <trail name>, EXTRACT <group name>
[, MEGABYTES <n>]
[SEQNO <n>]
```

引数	説明
`<trail name>`	トレイルの相対パス名または完全修飾パス名。トレイル名に含められる文字数は 2 文字のみです。Oracle GoldenGate は、新しいファイルが作成されるたびに、この名前に 6 桁の順序番号を追加します。たとえば、名前が dirdat/tr のトレイルのファイル名は、dirdat/tr000001、dirdat/tr000002 のようになります。
`<group name>`	トレイルに関連付ける Extract グループ名。1 つのトレイルにデータを書き込めるのは、1 つの Extract プロセスのみです。
`MEGABYTES <n>`	トレイル内のファイルの最大サイズ (MB)。デフォルトは 100 です。
`SEQNO <n>`	トレイル内の最初のファイルのトレイル順序番号を、指定した数字から開始させます。先行ゼロは付けないようにしてください。たとえば、"tr" という名前のトレイルを 3 番から開始するには、SEQNO 3 と指定します。実際のファイル名は、/ggs/dirdat/tr000003 になります。このオプションは、トラブルシューティングのために、Replicat の読取り位置を特定の順序番号のトレイルに変更する必要があるときに使用できます。これにより、所定の順序番号を読み取るように Replicat の設定を変更する必要がなくなります。

例

```
ADD EXTTRAIL dirdat\aa, EXTRACT finance, MEGABYTES 200
```

例

```
ADD EXTTRAIL /ggs/dirdat/tr000003
```

ADD RMTTRAIL

ADD RMTTRAIL では、リモート・システム上にオンライン処理用のトレイルを作成し、次のことも行います。

- 最大ファイル・サイズを指定する
- このトレイルに Extract グループを関連付ける

パラメータ・ファイルでは、リモート・システムおよび Manager プロセス用の TCP/IP ポートを識別するために、RMTTRAIL エントリの前に RMTHOST エントリを指定します。

構文

```
ADD RMTTRAIL <trail name>, EXTRACT <group name>
[, MEGABYTES <n>]
[, SEQNO <n>]
```

引数	説明
`<trail name>`	トレイルの相対パス名または完全修飾パス名。実際のトレイル名に含められる文字数は 2 文字のみです。Oracle GoldenGate は、新しいファイルが作成されるたびに、この名前に 6 桁の順序番号を追加します。たとえば、名前が ./dirdat/tr のトレイルのファイル名は、./dirdat/tr000001、./dirdat/tr000002 のようになります。
`<group name>`	トレイルに関連付ける Extract グループ名。1 つのトレイルにデータを書き込めるのは、1 つの Extract プロセスのみです。
`MEGABYTES <n>`	トレイル内のファイルの最大サイズ (MB)。デフォルトは 100 です。
`SEQNO <n>`	トレイル内の最初のファイルのトレイル順序番号を、指定した数字から開始させます。先行ゼロは付けないようにしてください。たとえば、"tr" という名前のトレイルを 3 番から開始するには、SEQNO 3 と指定します。実際のファイル名は、/ggs/dirdat/tr000003 になります。このオプションは、トラブルシューティングのために、Replicat の読取り位置を特定の順序番号のトレイルに変更する必要があるときに使用できます。これにより、所定の順序番号を読み取るように Replicat の設定を変更する必要がなくなります。

例

```
ADD RMTTRAIL dirdat\aa, EXTRACT finance, MEGABYTES 200
```

例

```
ADD RMTTRAIL /ggs/dirdat/tr000003
```

ALTER EXTTRAIL

ALTER EXTTRAIL では、ADD EXTTRAIL コマンドで作成した (ローカル・システム上の) トレイルの属性を変更します。この変更は、次に Extract を起動するときに有効になります。

構文

```
ALTER EXTTRAIL <trail name>, EXTRACT < group name>
[, MEGABYTES <n>]
```

引数	説明
`<trail name>`	トレイルの相対パス名または完全修飾パス名 (例 : dirdat\aa)。

引数	説明
<group name>	トレイルに関連付ける Extract グループ名。
MEGABYTES <n>	ファイルの最大サイズ (MB)。デフォルトは 100 です。このオプションの使用後、ROLLOVER オプションを指定して SEND EXTRACT コマンドを発行し、現在のファイルを閉じて新しいファイルを開きます。

例 `ALTER EXTTRAIL dirdat\aa, EXTRACT finance,MEGABYTES 200`

ALTER RMTTRAIL

ALTER RMTTRAIL では、ADD RMTTRAIL コマンドで作成した (リモート・システム上の) トレイルの属性を変更します。この変更は、次に Extract を起動するときに有効になります。

構文

```
ALTER RMTTRAIL <trail name>, EXTRACT <group name>
[, MEGABYTES <n>]
```

引数	説明
<trail name>	トレイルの相対パス名または完全修飾パス名 (例 : dirdat\aa)。
<group name>	トレイルに関連付ける Extract グループ名。
MEGABYTES <n>	ファイルの最大サイズ (MB)。デフォルトは 100 です。このオプションの使用後、ROLLOVER オプションを指定して SEND EXTRACT コマンドを発行し、現在のファイルを閉じて新しいファイルを開きます。

例

```
ALTER RMTTRAIL dirdat\aa, EXTRACT finance,
MEGABYTES 200
```

DELETE EXTTRAIL

DELETE EXTTRAIL では、ローカル・システム上のトレイルに関連付けられているチェックポイント・レコードを削除します。チェックポイントは、Oracle GoldenGate ディレクトリの dirchk サブディレクトリにある、グループと同じ名前のファイルに保持されています。

このコマンドでは、指定したトレイルに対するチェックポイント・ファイルからの参照のみが削除されます。トレイル・ファイル自体は削除されません。トレイル・ファイルを削除するには、オペレーティング・システムの標準のファイル削除コマンドを使用してください。

構文 `DELETE EXTTRAIL <trail name>`

引数	説明
<trail name>	2 文字の接頭辞を含むトレイルの相対パス名または完全修飾パス名。

例 DELETE EXTTRAIL dirdat/et

DELETE RMTTRAIL

DELETE RMTTRAIL では、リモート・システム上のトレイルに関連付けられているチェックポイントのレコードを削除します。チェックポイントは、Oracle GoldenGate ディレクトリの dirchk サブディレクトリにある、グループと同じ名前のファイルに保持されています。

このコマンドでは、指定したトレイルに対するチェックポイント・ファイルからの参照のみが削除されます。トレイル・ファイル自体は削除されません。トレイル・ファイルを削除するには、オペレーティング・システムの標準のファイル削除コマンドを使用してください。

構文 DELETE RMTTRAIL <trail name>

引数	説明
<trail name>	2 文字の接頭辞を含むトレイルの相対パス名または完全修飾パス名。

例 DELETE RMTTRAIL dirdat/et

INFO EXTTRAIL

INFO EXTTRAIL では、ローカル・トレイルの構成情報を取得します。トレイル名、書込み元の Extract、最新のデータ処理位置、割り当てられた最大ファイル・サイズが表示されます。

図 8 INFO EXTTRAIL の出力例

```
    Extract Trail: c:\gg_81\dirdat\md
          Extract: GGSEXT8
            Seqno: 2
              RBA: 51080
        File Size: 100M
```

構文 INFO EXTTRAIL <trail name>

引数	説明
<trail name>	トレイルの相対パス名または完全修飾パス名、または複数のトレイルを指定するワイルドカード。

例 1 `INFO EXTTRAIL dirdat\aa`

例 2 `INFO EXTTRAIL *`

INFO RMTTRAIL

INFO RMTTRAIL では、リモート・トレイルの構成情報を取得します。トレイル名、書込み元の Extract、最新のデータ処理位置、割り当てられた最大ファイル・サイズが表示されます。

図 9 INFO RMTTRAIL の出力例

```
    Extract Trail: /gg_81/dirdat/rt
          Extract: GGSEXT
            Seqno: 4
              RBA: 78066
        File Size: 100M
```

構文 `INFO RMTTRAIL <trail name>`

引数	説明
`<trail name>`	トレイルの相対パス名または完全修飾パス名、または複数のトレイルを指定するワイルドカード。

例 1 `INFO RMTTRAIL dirdat\aa`

例 2 `INFO RMTTRAIL *`

パラメータ・コマンド

パラメータ・コマンドでは、Oracle GoldenGate パラメータ・ファイルの表示および管理を行います。

コマンドの概要

EDIT PARAMS

SET EDITOR

VIEW PARAMS

EDIT PARAMS

EDIT PARAMS では、パラメータ・ファイルを作成または変更します。デフォルトでは、このコマンドによって Windows システムのメモ帳または UNIX システムの vi エディタが起動されます。エディタを変更するには、SET EDITOR コマンドを使用します。

警告 ローカル・オペレーティング・システムの文字セット以外の文字セットの、既存のパラメータ・ファイル (異なる文字セットを指定するために CHARSET オプションを使用したものなど) を表示または編集するために、このコマンドを使用しないでください。内容が破損することがあります。GGSCI 外部からパラメータ・ファイルを表示してください。

構文 `EDIT PARAMS {MGR | <group> | <file name>}`

引数	説明
`MGR`	Manager プロセスのパラメータ・ファイルを開きます。
`<group>`	指定した Extract または Replicat グループのパラメータ・ファイルを開きます。
`<file name>`	指定したファイルを開きます。GGSCI の EDIT PARAMS を使用してパラメータ・ファイルを作成すると、Oracle GoldenGate ディレクトリの dirprm サブディレクトリに保存されます。完全パス名を指定することによって、dirprm 以外のディレクトリにパラメータ・ファイルを作成することもできますが、プロセス・グループの作成時に、ADD EXTRACT または ADD REPLICAT コマンドの PARAMS オプションを使用して、完全パス名を指定する必要があります。

例 1 `EDIT PARAMS finance`

例 2 `EDIT PARAMS c:\lpparms\replp.prm`

SET EDITOR

SET EDITOR では、現在の GGSCI セッションのデフォルト・テキスト・エディタを変更します。Windows のデフォルト・エディタはメモ帳、UNIX は vi です。パラメータ・ファイルを作成するためなどの GGSCI 入力では、ローカル・オペレーティング・システムの文字セットを取得します。

構文 `SET EDITOR <program name>`

引数	説明
`<program name>`	任意のテキスト・エディタ。

例 次の例では、デフォルト・エディタをワードパッドに変更します。

```
SET EDITOR wordpad
```

VIEW PARAMS

VIEW PARAMS では、パラメータ・ファイルの内容を表示します。

警告 **ローカル・オペレーティング・システムの文字セット以外の文字セットのパラメータ・ファイル (異なる文字セットを指定するために CHARSET オプションを使用したものなど) を表示するために、このコマンドを使用しないでください。内容が破損することがあります。GGSCI 外部からパラメータ・ファイルを表示してください。**

構文 `VIEW PARAMS {MGR | <group> | <file name>}`

引数	説明
`MGR`	Manager パラメータ・ファイルを表示します。

引数	説明
<group>	指定した Extract または Replicat グループのパラメータ・ファイルを表示します。
<file name>	指定したファイルを表示します。パラメータ・ファイルが dirprm 以外のディレクトリに存在する場合、完全パス名を指定します。

例 1 `VIEW PARAMS finance`

例 2 `VIEW PARAMS c:\lpparms\replp.prm`

データベース・コマンド

データベース・コマンドでは、データベースと通信します。

コマンドの概要

DBLOGIN

DBLOGIN では、データベースを操作する他の Oracle GoldenGate コマンドの発行準備として、データベースへの接続を確立します。DBLOGIN を発行するユーザーは、これらのコマンドによって成立する機能を実行するための適切なデータベース権限を持っている必要があります。GGSCI コマンドに必要なその他の特別な権限は、そのコマンドの参照ドキュメントでリストされています。

Oracle 統合キャプチャ・モードの構成時の要件

DBLOGIN を使用して REGISTER EXTRACT を発行し、Oracle データベースに対して統合キャプチャを開始する場合、DBLOGIN を発行するユーザーは、次を満たす必要があります。

- Oracle dbms_goldengate_auth.grant_admin_privilege プロシージャを使用して付与された権限がある。
- このDBLOGINに関連付けられているExtractグループのUSERIDパラメータで指定されているユーザーである。
- Extract が統合キャプチャ・モードである間は変更されない。

クラシック・キャプチャ・モードでログ保持を使用するための特別なデータベース権限

Oracle データベースのクラシック・キャプチャ・モードでログ保持機能を有効にするには、REGISTER EXTRACT を LOGRETENTION オプションで使用する前に、DBLOGIN を特別な権限で発行する必要があります。Oracle GoldenGate のインストール時に適切な権限がユーザーに付与されている場合、簡単にするた

めに、Extract データベース・ユーザーとしてログインできます。それ以外の場合は、次の権限を持つユーザーとしてログインします。

表 1　　TRANLOG オプションの Oracle EE 10.2 以降の権限

Oracle EE バージョン	権限

10.2	1. Oracle Streams 管理権限を付与するパッケージを実行します。
	`exec dbms_streams_auth.grant_admin_privilege('<user>')`
	2. INSERT を logmnr_restart_ckpt$ に付与します。
	`grant insert on system.logmnr_restart_ckpt$ to <user>;`
	3. UPDATE を streams$_capture_process に付与します。
	`grant update on sys.streams$_capture_process to <user>;`
	4. 'become user' 権限を付与します。
	`grant become user to <user>;`

11.1 および 11.2.0.1	1. Oracle Streams 管理権限を付与するパッケージを実行します。
	`exec dbms_streams_auth.grant_admin_privilege('<user>')`
	2. 'become user' 権限を付与します。
	`grant become user to <user>;`

11.2.0.2 以降	Oracle Streams 管理権限を付与するパッケージを実行します。 `exec dbms_goldengate_auth.grant_admin_privilege('<user>')`

構文

```
DBLOGIN {
[SOURCEDB <dsn>] |
[, <database>@<host>:<port>] |
USERID {/ | <user id>}[, PASSWORD <password>]
[<algorithm> ENCRYPTKEY {<keyname> | DEFAULT}] [SYSDBA | SQLID <sqlid>]
[SESSIONCHARSET <character set>]
}
```

引数	説明
`SOURCEDB <dsn>`	データソース名。Sybase、MySQL、および ODBC を使用するデータベースで必要です。
`<database>@<host>:<port>`	(MySQL) データベース名、ホスト名およびデータベース・ポート番号を含む接続文字列を指定します。データベース構成で指定されているデフォルト以外のポートを指定するのに使用できます。
`USERID <user id>`	データベースの資格証明が必要な場合に使用します。データベース・ユーザー名、スキーマ (SQL/MX) または SQL*Net 接続文字列 (Oracle) を指定します。

引数	説明
PASSWORD <password>	データベース・ユーザーのパスワードを指定するために認証が必要な場合に使用します。パスワードが ENCRYPT PASSWORD コマンドによって暗号化されている場合は、暗号化されたパスワードを指定します。それ以外の場合は、クリアテキストのパスワードを指定します。パスワードに大文字と小文字の区別がある場合は、そのとおりに入力します。 PASSWORD 句を省略すると、パスワードの入力が求められ、パスワードはエコーされません。 ユーザー ID またはパスワードのいずれかが変更されると、必要に応じて、パスワードの再暗号化など、Oracle GoldenGate パラメータ・ファイルの変更を行う必要があります。
<algorithm>	ENCRYPT PASSWORD を使用してパスワードが暗号化された場合、使用した暗号化アルゴリズムを次の中から指定します。 AES128 AES192 AES256 BLOWFISH
ENCRYPTKEY {<keyname> | DEFAULT}	ENCRYPT PASSWORD で指定した暗号化鍵を指定します。 ◆ ENCRYPTKEY <keyname> は、ENCKEYS 参照ファイル内のユーザー作成の暗号化鍵の論理名を指定します。ENCRYPT PASSWORD が KEYNAME <keyname> オプションとともに使用された場合に使用します。 ◆ ENCRYPTKEY DEFAULT は、Oracle GoldenGate に、ランダム鍵を使用するように指示します。ENCRYPT PASSWORD が KEYNAME DEFAULT オプションとともに使用された場合に使用します。
SYSDBA	(Oracle) ユーザーが sysdba としてログインするように指定します。
SQLID <sqlid>	(z/OS 上の DB2)USERID のログインが (適切な場合は PASSWORD を使用して) 完了した後に、SQL コマンド SET CURRENT SQLID = 'sqlid' を発行します。SET コマンドが失敗すると、DBLOGIN コマンド全体がユニットとして失敗します。
SESSIONCHARSET <database character set>	(Sybase、Teradata および MySQL) GGSCI をデータベースに接続するために、データベース・セッションの文字セットを設定します。後続のすべてのコマンドは、指定したセッションの文字セットを使用します。このコマンド・オプションは、GLOBALS ファイルで指定されるすべての SESSIONCHARSET より優先されます。

例 1
```
DBLOGIN USERID ogg@ora1.ora, PASSWORD AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC
AES128, ENCRYPTKEY securekey1
```

例 2
```
DBLOGIN SOURCEDB msqldb@host1:3305, USERID ogg@ora1.ora, PASSWORD
AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC AES128, ENCRYPTKEY securekey1
```

例 3

```
DBLOGIN SOURCEDB msqldb@host1:3305, USERID ogg@ora1.ora, PASSWORD
AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC AES128, ENCRYPTKEY securekey1,
SESSIONCHARSET ISO-8859-11
```

ENCRYPT PASSWORD

ENCRYPT PASSWORD では、Oracle GoldenGate パラメータ・ファイルまたはコマンドで使用されるパスワードを暗号化します。

構文

```
ENCRYPT PASSWORD <password>
[AES128 | AES192 | AES256 | BLOWFISH]
ENCRYPTKEY {<keyname> | DEFAULT}
```

引数	説明
<password>	ログイン・パスワード。パスワードは引用符で*囲みません*。パスワードに大文字と小文字の区別がある場合は、そのとおりに入力します。
AES128 | AES192 | AES256 | BLOWFISH	使用する暗号化アルゴリズムを指定します。 ◆ AES128 は、キー・サイズが 128 ビットの AES-128 暗号を使用します。 ◆ AES192 は、キー・サイズが 192 ビットの AES-192 暗号を使用します。 ◆ AES256 は、キー・サイズが 256 ビットの AES-256 暗号を使用します。 ◆ BLOWFISH は、ブロック・サイズが 64 ビットで可変長のキー・サイズが 32 ビットから 128 ビットの Blowfish 暗号化を使用します。BLOWFISH は、以前の Oracle GoldenGate バージョンとの下位互換性を維持するためにのみ使用します。 アルゴリズムを指定しない場合、z/OS 上の DB2 および NonStop SQL/MX 以外のすべてのデータベース・タイプでは AES128 がデフォルトになり、z/OS 上の DB2 および NonStop SQL/MX では BLOWFISH がデフォルトになります。 すべての AES 暗号のブロック・サイズは 128 ビットです。これらの暗号の詳細は、Advanced Encryption Security のオンライン出版を参照してください。 Oracle 以外のデータベースに AES 暗号化を使用するには、プロセスを開始する前に、次のように、環境変数として、Oracle GoldenGate インストール・ディレクトリの lib サブディレクトリのパスを指定する必要があります。 ◆ UNIX: LD_LIBRARY_PATH または SHLIB_PATH 変数への入力として、パスを指定します。次に例を示します。 `setenv LD_LIBRARY_PATH ./lib:$LD_LIBRARY_PATH` ◆ Windows: PATH 変数にパスを追加します。 SETENV パラメータを使用して、プロセスのセッション変数として設定できます。

引数	説明
ENCRYPTKEY {<keyname> | DEFAULT}	暗号化鍵を指定します。 ◆ <keyname> は、ローカルの ENCKEYS 参照ファイル内のユーザー作成の暗号化鍵の論理名を指定します。鍵名は ENCKEYS ファイル内の実際の鍵を参照するのに使用されます。ユーザー作成の鍵および関連付けられた ENCKEYS ファイルは、AES 暗号化を使用する際には必須です。Blowfish 暗号化の場合はオプションですが、使用することををお薦めします。<keyname> を使用するには、KEYGEN または別のユーティリティを使用して鍵を生成し、それをソースおよびターゲット・システムの ENCKEYS ファイルに保管します。詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』のセキュリティ・ガイドラインを参照してください。 ◆ DEFAULT は、Oracle GoldenGate に、ダウンストリーム・プロセスによって復号化を実行できるようトレイル内に格納されるランダム鍵を生成するように指示します。この鍵のタイプは安全ではないため、本番環境では使用しないでください。このオプションは、BLOWFISH を指定した場合のみ使用します。DEFAULT が AES アルゴリズムで使用された場合、ENCRYPT PASSWORD はエラーを返します。

例 1 `ENCRYPT PASSWORD ny14072 AES192 ENCRYPTKEY superkey2`

例 2 `ENCRYPT PASSWORD ny14072 BLOWFISH ENCRYPTKEY superkey3`

例 3 `ENCRYPT PASSWORD ny14072 BLOWFISH ENCRYPTKEY DEFAULT`

FLUSH SEQUENCE

FLUSH SEQUENCE は、初期同期化中または再同期化中の最初に Extract を起動した直後に使用します。このコマンドは、Extract がトランザクション・データの取得を開始した時点で初期 REDO レコードが使用できるように、Oracle 順序を更新します。通常は、現在のキャッシュが使い果たされるまで、REDO は生成されません。フラッシュは、ターゲット・システムで正しい順序値に同期するために使用する初期の開始位置を、Replicat に提供します。それ以降は、Extract は、順序値の通常のキャッシュ予約に関連付けられた REDO を使用できます。

FLUSH SEQUENCE の使用方法

1. FLUSH SEQUENCE は、次の Oracle プロシージャを使用します。

表 2 FLUSH SEQUENCE をサポートするプロシージャ

データベース	プロシージャ	ユーザーおよび権限
ソース	updateSequence	Oracle GoldenGate DDL オブジェクトの所有者、または DDL サポートを使用しない場合は他の選択したユーザーに、EXECUTE を付与します。
ターゲット	replicateSequence	Oracle GoldenGate Replicat ユーザーに EXECUTE を付与します。

sequence.sql スクリプトによって、これらのプロシージャはインストールされます。通常、このスクリプトは Oracle GoldenGate インストール・プロセスの一部として実行されますが、FLUSH

SEQUENCE を使用する前に、これが実行されたことを確認してください。sequence.sql が実行されていないと、フラッシュは失敗し、次に示すようなエラー・メッセージが生成されます。

```
Cannot flush sequence {0}. Refer to the Oracle GoldenGate for Oracle documentation
for instructions on how to set up and run the sequence.sql script. Error {1}.
```

2. GLOBALS ファイルには、プロシージャをインストールするスキーマを指定する GGSCHEMA パラメータが含まれている必要があります。このユーザーは、CONNECT、RESOURCE および DBA 権限を持っている必要があります。
3. FLUSH SEQUENCE を使用する前に、updateSequence プロシージャで EXECUTE 権限を持つデータベース・ユーザーとして、DBLOGIN コマンドを発行します。

注意 FLUSH SEQUENCE をサポートするための Oracle GoldenGate の構成に関する詳細な方法は、『Oracle GoldenGate *Oracle インストレーションおよびセットアップ・ガイド*』を参照してください。

構文

```
FLUSH SEQUENCE <owner.sequence>
```

引数	説明
`<owner.sequence>`	Oracle 順序の所有者および名前。スキーマ名は NULL にすることはできません。順序名にはアスタリスク (*) ワイルドカードを使用できますが、所有者名には使用できません。

例

```
FLUSH SEQUENCE scott.seq*
```

LIST TABLES

LIST TABLES では、コマンド引数の指定に一致するデータベース内のすべての表を一覧表示します。このコマンドを使用する前に、DBLOGIN コマンドを使用してデータベース接続を確立してください。

構文

```
LIST TABLES <table>
```

引数	説明
`<table>`	表名、またはワイルドカード (*) で指定した表グループ。

例 次に LIST TABLES コマンドおよび出力例を示します。

```
GGSCI (sysa) 3> list tables tcust*
TCUSTMER
TCUSTORD
```

MININGDBLOGIN

MININGDBLOGIN では、REGISTER EXTRACT など、データベースを操作する他の Oracle GoldenGate コマンドの発行準備として、ダウンストリーム Oracle データベース・ログマイニング・サーバーへの接続を確立します。このコマンドは、統合キャプチャ・モードで Extract を確立する場合のみ使用します。

データベース・ログマイニング・サーバーとして機能する、ソース Oracle データベースにログインするには、DBLOGIN コマンドを使用します。MININGDBLOGIN は、ダウンストリーム・マイニング・データベースへのログイン用に予約されます。

MININGDBLOGIN を発行するユーザーは、次を満たす必要があります。

- Oracle dbms_goldengate_auth.grant_admin_privilege プロシージャを使用して付与された権限がある。
- このMININGDBLOGINに関連付けられているExtractグループのTRANLOGOPTIONS MININGUSERパラメータで指定されているユーザーである。
- Extract が統合キャプチャ・モードである間は変更されない。

統合キャプチャのサポートおよび構成情報は、『Oracle GoldenGate *Oracle インストレーションおよびセットアップ・ガイド*』を参照してください。

構文

```
MININGDBLOGIN {
USERID {/ | <user id>}[, PASSWORD <password>]
[<algorithm> ENCRYPTKEY {<keyname> | DEFAULT}] [SYSDBA]
}
```

引数	説明
`<user id>`	データベース・ユーザー名または SQL*Net 接続文字列を指定します。
`<password>`	データベース・ユーザーのパスワードを指定するために認証が必要な場合に使用します。パスワードが ENCRYPT PASSWORD コマンドによって暗号化されている場合は、暗号化されたパスワードを指定します。それ以外の場合は、クリアテキストのパスワードを指定します。パスワードに大文字と小文字の区別がある場合は、そのとおりに入力します。 PASSWORD 句を省略すると、パスワードの入力が求められ、パスワードはエコーされません。 ユーザー ID またはパスワードのいずれかが変更されると、必要に応じて、パスワードの再暗号化など、Oracle GoldenGate パラメータ・ファイルの変更を行う必要があります。ただし、このユーザーは一定のままにすることをお薦めします。
`<algorithm>`	ENCRYPT PASSWORD を使用してパスワードが暗号化された場合、使用した暗号化アルゴリズムを次の中から指定します。 AES128 AES192 AES256 BLOWFISH
`ENCRYPTKEY {<keyname> | DEFAULT}`	ENCRYPT PASSWORD で指定した暗号化鍵を指定します。 ◆ ENCRYPTKEY <keyname> は、ENCKEYS 参照ファイル内のユーザー作成の暗号化鍵の論理名を指定します。ENCRYPT PASSWORD が KEYNAME <keyname> オプションとともに使用された場合に使用します。 ◆ ENCRYPTKEY DEFAULT は、Oracle GoldenGate に、ランダム鍵を使用するように指示します。ENCRYPT PASSWORD が KEYNAME DEFAULT オプションとともに使用された場合に使用します。

引数	説明
SYSDBA	ユーザーが sysdba としてログインするように指定します。

例

```
MININGDBLOGIN USERID ogg@ora2.ora, PASSWORD
AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC AES128, ENCRYPTKEY securekey1
```

Trandata コマンド

Trandata コマンドでは、適切なデータベース・コンポーネントを構成して、ソース・データ操作をレプリケートするために Oracle GoldenGate が必要とするトランザクション情報を提供します。

コマンドの概要

ADD SCHEMATRANDATA

ADD TRANDATA

DELETE SCHEMATRANDATA

DELETE TRANDATA

INFO SCHEMATRANDATA

INFO TRANDATA

ADD SCHEMATRANDATA

ADD SCHEMATRANDATA では、Oracle 表のスキーマレベルのサプリメンタル・ロギングを有効化します。ADD SCHEMATRANDATA は、行の識別に Oracle GoldenGate が必要とする使用可能なキーのスーパーセットを自動的に記録するために指定したスキーマの、現在および将来のすべての表で機能します。

ADD SCHEMATRANDATA は、次のことを行います。

- CREATE TABLE で作成された新しい表に Oracle サプリメンタル・ロギングを有効化する。
- 列を追加または削除する ALTER TABLE の影響を受けた表のサプリメンタル・ロギングを更新する。
- 名前が変更された表のサプリメンタル・ロギングを更新する。
- 一意キーまたは主キーが追加または削除された表のサプリメンタル・ロギングを更新する。

ADD SCHEMATRANDATA は、次に示す優先順位で、表のキー列を記録します。

- 主キー
- 主キーがない場合、無効、使用不可または非表示のキーを含む、表のすべての一意キー。ADT メンバー列を含む一意キーも記録されます。仮想列 (ファンクションベース索引) の一意キーのみ記録されません。

- 上記のいずれも存在しない場合、表のすべてのスカラー列が記録されます。(システム生成の行 OID は常に記録されます。)

ADD SCHEMATRANDATA を使用するタイミング

ADD SCHEMATRANDATA は、次の場合に使用する必要があります。

- 統合キャプチャ用に構成する Extract グループの一部であるすべての表に対して。ADD SCHEMATRANDATA は、すべてのキーを記録することにより、適切なキーが記録されていることを保証します。
- DDL レプリケーションがアクティブで、DML が新しい表を作成するかキー列を変更する DDL と同時である場合。DDL がオブジェクトで発行された直後に DML をオブジェクトに適用できるシナリオを最適に処理します。ADD SCHEMATRANDATA により、各 DDL 操作で適切なキー値が REDO ログにアトミックに記録されるため、Extract 処理でのラグに関係なく、ログから取得される際の DML のメタデータの継続性が保証されます。

ADD SCHEMATRANDATA を使用する場合の追加の要件

- Oracle GoldenGate が主キーおよび連鎖行に対する更新を処理するためには、*データベース・レベル*で最小サプリメンタル・ロギングを有効にする必要があります。これは、Oracle GoldenGate ではなく、データベース・インタフェースを通じて行う必要があります。次の DDL 文を発行することによって、最小サプリメンタル・ロギングを有効にできます。

  ```
  SQL> alter database add supplemental log data;
  ```

  データベース・レベルでサプリメンタル・ロギングが有効化されていることを検証するために、次の文を発行してください。

  ```
  SELECT SUPPLEMENTAL_LOG_DATA_MIN FROM V$DATABASE;
  ```

  この問合せの出力が YES または IMPLICIT になる必要があります。他の LOG_DATA オプションを設定していても自動的に有効化されないため、LOG_DATA_MIN を明示的に設定する必要があります。

- ADD SCHEMATRANDATA を使用する前に、DBLOGIN コマンドを発行します。このコマンドを発行するユーザーは、Oracle Streams 管理者権限が付与されている必要があります。

  ```
  SQL> exec dbms_streams_auth.grant_admin_privilege('<user>')
  ```

ADD SCHEMATRANDATA を使用する場合の追加の考慮事項

- DDL レプリケーションが有効化されていない場合、ADD TRANDATA コマンドのかわりに ADD SCHEMATRANDATA を使用できます。ただし、表に主キーがなく、複数の一意キーがある場合、ADD SCHEMATRANDATA によって、データベースはすべての一意キーを記録することに注意してください。このような場合、ADD SCHEMATRANDATA によって、データベースは ADD TRANDATA の場合よりも多くの REDO データを記録します。余分なロギングを回避するには、可能な場合、一意キーのうちの 1 つを主キーとして指定します。
- 1 つの主キーがあるか、1 つの一意キーがあるか、キーがない表の場合、ADD SCHEMATRANDATA では、ADD TRANDATA と比較して、追加のロギング・オーバーヘッドはありません。詳細は、86 ページの「ADD TRANDATA」を参照してください。
- FILTER 文、および TABLE パラメータと MAP パラメータの KEYCOLS 句に必要な列など、Oracle GoldenGate で使用するために特定の表の追加の非キー列を記録する必要がある場合、これらの列に対して ADD TRANDATA コマンドを発行します。このコマンドには、列の表レベルのサプリメンタル・ロギングを発行するための COLS オプションがあり、ADD SCHEMATRANDATA とともに使用できます。

構文　　ADD SCHEMATRANDATA <schema>

引数	説明
<schema>	キーの追加情報を記録するスキーマ。ワイルドカードは使用しないでください。

例　　ADD SCHEMATRANDATA SCOTT

ADD TRANDATA

ADD TRANDATA では、Oracle GoldenGate が必要とするトランザクション情報をトランザクション・レコードから取得します。このコマンドを使用する前に、DBLOGIN コマンドを使用してデータベース接続を確立してください。

ADD TRANDATA は、次に示すデータベースでのみ有効です。サポートされている他のデータベースでは、この機能がすでに存在しているか、またはデータベース・インタフェースを使用して構成する必要があります。トランザクション情報を使用可能にするための特別な要件の詳細は、ご使用のデータベース用の Oracle GoldenGate インストレーション・ガイドを参照してください。

DB2 データベース

ADD TRANDATA では、指定した表の DATA CAPTURE CHANGES を有効化します。このコマンドは、DB2 LUW および DB2 z/OS をサポートしています。デフォルトでは、ADD TRANDATA はデータベースに次のいずれかのコマンドを発行します。

DB2 z/OS:

```
ALTER TABLE <name> DATA CAPTURE CHANGES;
```

DB2 LUW:

```
ALTER TABLE <name> DATA CAPTURE CHANGES INCLUDE LONGVAR COLUMNS;
```

DB2 LUW では、ADD TRANDATA と EXCLUDELONG オプションを使用することで、LONGVAR 句を除外できます。

SQL Server データベース

ADD TRANDATA では、Oracle GoldenGate が SQL 操作を再構成するために必要とするサプリメンタル・ロギング情報を有効化します。SQL Server トランザクション・ログは、デフォルトでは十分な情報を提供しません。

Sybase データベース

ADD TRANDATA では、Sybase の sp_setreptable および sp_setrepcol システム・プロシージャを実行して、Sybase 表をレプリケーション対象としてマーク付けします。ADD TRANDATA オプションでは、データベース機能を使用して、データベースが指定した表の LOB データを伝播する方法を制御します。ADD TRANDATA オプションのリストを参照してください。

注意　　ADD TRANDATA コマンドは、表に対して現在設定されている LOB 設定を上書きします。

Oracle データベース

ADD TRANDATA は、デフォルトで表レベルのサプリメンタル・ロギングを有効化します。このコマンドは、表に対して定義されている一意制約のタイプ (または一意制約のない場合) に適した、ADD SUPPLEMENTAL LOG DATA 句を含む ALTER TABLE コマンドを発行します。

KEYCOLS 句が TABLE 文または MAP 文で使用されていない場合、Oracle GoldenGate は、次の優先順序で使用する行識別子を選択します。

1. 主キー
2. 英数字順で最初の一意キー (仮想列、UDT、ファンクションベース列および Null 値可能列を含まない)。キーには、不可視索引の一部となっている列を含めることはできません。
3. 英数字順で最初の一意キー (仮想列、UDT、ファンクションベース列を含まないが、Null 値可能列を含める)。キーには、不可視索引の一部となっている列を含めることはできません。
4. (表に他のタイプのキーが定義されているとしても) 上記のいずれのタイプのキーも存在しない場合、Oracle GoldenGate は、仮想列、UDT、ファンクションベース列、および Oracle GoldenGate 構成から明示的に除外されているすべての列を除く、データベースに主キーでの使用が許可されているすべての列で疑似キーを作成します。

> **注意** **他の使用不能なキーが表にある場合、または表にキーがない場合、Oracle GoldenGate は適切なメッセージをレポート・ファイルに出力します。すべての列からキーを作成すると、ソース・システムの Oracle GoldenGate のパフォーマンスが低下します。ターゲットでは、このキーは、Replicat であまり効率的でないより大きい WHERE 句が使用される原因となります。**

Oracle GoldenGate の DDL レプリケーション機能を使用*していない*場合のみ、ADD TRANDATA を使用します。Oracle GoldenGate の DDL レプリケーション機能を使用している場合、ADD SCHEMATRANDATA コマンドを使用して、必要な補足データを記録します。DDL サポートが有効なときは ADD TRANDATA を使用できますが、次のいずれかを保証できる場合のみです。

- ユーザーまたはアプリケーションが表で DDL を実行する前に、任意およびすべての表で DML アクティビティを停止できる。
- DDL が発生する前に DML アクティビティを停止できるが、次のことを保証できる。
  - ユーザーまたはアプリケーションは、TABLE または MAP 文で明示的またはワイルドカード使用の指定を満たす名前の新しい表を追加する DDL を発行する可能性がない。
  - ユーザーまたはアプリケーションは、Oracle GoldenGate 構成にすでに存在する表のキー定義を変更する DDL を発行する可能性がない。

ADD SCHEMATRANDATA では、レプリケーションの継続性により、DDL が実行されたオブジェクトで DML が発生することを保証します。詳細は、84 ページの「ADD SCHEMATRANDATA」を参照してください。

TABLE および MAP パラメータで FILTER 文および KEYCOLS 句に必要な列など、非キー列を記録するために COLS オプションを使用する必要がある場合は、ADD SCHEMATRANDATA を使用していても ADD TRANDATA を使用できます。

Oracle GoldenGate が主キーおよび連鎖行に対する更新を処理するためには、表レベルのロギングに加えて、データベース・レベルで最小サプリメンタル・ロギングを有効にする必要があります。これは、Oracle GoldenGate ではなく、データベース・インタフェースを通じて行う必要があります。次の DDL 文を発行することによって、最小サプリメンタル・ロギングを有効にできます。

```
SQL> alter database add supplemental log data;
```

データベース・レベルでサプリメンタル・ロギングが有効化されていることを検証するために、次の文を発行してください。

```
SELECT SUPPLEMENTAL_LOG_DATA_MIN FROM V$DATABASE;
```

この問合せの出力が YES または IMPLICIT になる必要があります。他の LOG_DATA オプションを設定していても自動的に有効化されないため、LOG_DATA_MIN を明示的に設定する必要があります。

Oracle の ADD TRANDATA サプリメンタル・ロギングには、次の追加のオプションがあります。

- COLS オプションでは、KEYCOLS 句のため、または要件のフィルタリングや操作のために必要な列など、非キー列を必要に応じて記録します。KEYCOLS 句は、処理の開始時にチェックされ、主キーまたは一意キーがないことが判断された場合に ADD TRANDATA が表のすべての列を記録することを防止します。
- NOKEY オプションでは、必要に応じてキー列のロギングを抑止します。

Oracle データベースに対して ADD TRANDATA を使用する際は、次の事項を考慮してください。

- Oracle GoldenGate がデータの抽出を開始した後にロギングの詳細設定を変更した場合、データが変更される前に、影響を受ける表からデータを読取り中の Extract プロセスを停止し、起動しなおす必要があります。
- ADD TRANDATA を使用してサプリメンタル・ログ・グループを作成する場合、Oracle GoldenGate は接頭辞 GGS_ に、表名、アンダースコアおよびオブジェクト ID を追加します。Oracle ではオブジェクト名が 30 文字までに制限されているため、Oracle GoldenGate は接頭辞とオブジェクト ID をオブジェクト名に含めるために、必要に応じて長い表名を切り捨てます。

構文

```
ADD TRANDATA <owner.table>
[, COLS (<column list>)]
[, INCLUDELONG | EXCLUDELONG]
[, LOBSNEVER | LOBSALWAYS | LOBSIFCHANGED | LOBSALWAYSNOINDEX]
[, NOKEY]
```

引数	説明
<owner.table>	トランザクション・データをロギングする表またはファイルの所有者および名前。ワイルドカードは、オブジェクト名には使用できますが所有者名には使用できません。ワイルドカードとともに使用すると、ADD TRANDATA は、システム・オブジェクトの名前と一致する名前を除外します。システム・オブジェクトではないが、ワイルドカード・パターンでシステム・オブジェクト名と一致する名前のオブジェクトに対して ADD TRANDATA を使用するには、ワイルドカードを使用せずに ADD TRANDATA を発行します。
COLS (<column list>)	サプリメンタル・ロギングに特定の非キー列を追加します。KEYCOLS 句に指定した列、およびフィルタ処理や操作のために必要となる列のロギングに使用でき、TABLE 文の FETCHCOLS 句を使用するときよりも効率的に列の値をフェッチできる場合があります。カンマを使用して複数の列を区切ります (例 : NAME, ID, DOB)。

引数	説明

INCLUDELONG | EXCLUDELONG	(DB2 LUW)ADD TRANDATA によって発行される ALTER TABLE に "INCLUDE LONGVAR COLUMNS" 属性を含めるかどうかを制御します。デフォルトは INCLUDELONG です。ADD TRANDATA をこのオプションとともに発行すると、Oracle GoldenGate は次の文を発行します。
	`ALTER TABLE <name> DATA CAPTURE CHANGES INCLUDE LONGVAR COLUMNS;`
	EXCLUDELONG を使用すると、次のコマンドが発行されます。
	`ALTER TABLE <name> DATA CAPTURE CHANGES;`
	EXCLUDELONG を使用する場合、Oracle GoldenGate では LONGVAR 列を含む表のビフォア・イメージを必要とする機能がサポートされません。この機能の例として、GETUPDATEBEFORES、NOCOMPRESSUPDATES および NOCOMPRESSDELETES パラメータがあります。この機能をサポートするには、トランザクション・ログの LONGVAR 列の設定を変更し、列値のビフォアおよびアフター・イメージ両方を含めるようにする必要があります。

LOBSNEVER | LOBSALWAYS | LOBSIFCHANGED | LOBSALWAYSNOINDEX	(Sybase) データベースが、指定した表の LOB データを伝播する方法を制御します。 **注意 :** ADD TRANDATA コマンドは、表に対して現在設定されている LOB 設定を上書きします。その後に設定を変更するには、sp_setrepcol スクリプトを使用する必要があります。 ◆ LOBSNEVER を使用すると、LOB データは伝播されません。ただし例外があり、LOB 列に NULL 値が挿入されるか、LOB 列が INSERT 操作でスキップされた場合、Extract はトレイル内のこの列に NULL データを書き込みます。 ◆ LOBSALWAYSでは2つのことを行います: sp_setrepcolを使用してLOBレプリケーションをALWAYS_REPLICATE(トランザクション内で変更されたかどうかにかかわらず、常に LOB データをレプリケートする) に設定し、(sp_setreptable の USE_INDEX オプションを使用して) レプリケーションで索引を使用するように表をマーク付けします。単一のトランザクション内で LOB がレプリーション対象としてマーク付けされると長時間かかる可能性があるので、USE_INDEX を使用して各 LOB にグローバル・ノンクラスタード索引を作成してその時間を短縮します。グローバル・ノンクラスタード・インデックスの作成中、共有表ロックは保持されます。 ◆ LOBSIFCHANGED では、トランザクション中にLOBデータが変更された場合にLOBデータのみをレプリケートします。これにより、レプリケーションのオーバーヘッドは低減されますが、レプリケーション環境外のターゲットで発生する可能性がある不整合は防止できません。これはデフォルトです。 ◆ LOBSALWAYSNOINDEX では、LOB レプリケーションを ALWAYS_REPLICATE(トランザクション内で変更されたどうかにかかわらず、常に LOB データをレプリケートする) に設定します。これによりオーバーヘッドは追加されますが、レプリケーション環境外のターゲットで発生する可能性がある不整合は防止できます。LOBSALWAYSNOINDEX では、レプリケーションの索引を使用するように表をマーク付けしません。この設定のメリットは、ADD TRANDATA の実行中にロックが保持されないことです。LOBSALWAYSNOINDEX は、バージョン 15 より前の Sybase データベースのデフォルトです。

引数	説明
	注意：ALWAYS_REPLICATE オプションを使用しているときに、特定の LOB 列に NULL 値が含まれている場合、表内の (LOB でない) 他の列が更新されると、ALWAYS_REPLICATE を有効化していても、この LOB は取得されません。 表の LOB 設定を確認するには、その表に対して ADD TRANDATA を使用した後に、INFO TRANDATA コマンドを使用します。すべての LOB 列の LOB 設定が表示されます。必要に応じて、Sybase システム・プロシージャを使用して特定の列の LOB 設定を変更できます。
NOKEY	主キー列のサプリメンタル・ロギングを抑止します。NOKEY を使用する場合は、COLS オプションを使用してキーとして機能できる別の列をロギングし、TABLE または MAP パラメータの KEYCOLS オプションを使用してこれらの列を代替キーとして指定します。

例 1 次の例では、Oracle 表の主キーのロギング、SQL Server 表の補足データのロギング、またはレプリケーション対象の Sybase 表のマーク付けのいずれかを行います。

```
ADD TRANDATA finance.acct
```

例 2 次の Oracle の例では、主キー、および非キー列 name および address をロギングします。

```
ADD TRANDATA finance.acct, COLS (name, address)
```

例 3 次の Oracle の例では、主キーはロギングせず、かわりに非キー列の name と pid をロギングします。

```
ADD TRANDATA finance.acct, NOKEY, COLS (name, pid)
```

例 4 次の Sybase の例では、レプリケーション対象の acct 表にマーク付けし、トランザクション中に変更された場合にのみ LOB データをロギングするように指定します。

```
ADD TRANDATA finance.acct, LOBSIFCHANGED
```

DELETE SCHEMATRANDATA

DELETE SCHEMATRANDATA では、ADD SCHEMATRANDATA コマンドで追加された Oracle のスキーマレベルのサプリメンタル・ロギングを削除します。このコマンドを使用する前に、DBLOGIN コマンドを使用してデータベース接続を確立してください。このコマンドで指定するユーザーは、サプリメンタル・ログ・グループを削除する権限を持っている必要があります。

構文

```
DELETE SCHEMATRANDATA <schema>
```

引数	説明
<schema>	サプリメンタル・ロギングを削除するスキーマ。ワイルドカードは使用しないでください。

例

```
DELETE SCHEMATRANDATA SCOTT
```

DELETE TRANDATA

DELETE TRANDATA では、次のいずれかの操作を行います。

- DB2 LUW および z/OS 上の DB2: 表を DATA CAPTURE NONE に変更
- Oracle: サプリメンタル・ロギングの無効化
- Sybase: レプリケーションの無効化
- SQL Server: 拡張ロギングの停止

このコマンドを使用する前に、DBLOGIN コマンドを使用してデータベース接続を確立してください。このコマンドで指定するユーザーは、ADD TRANDATA の実行に必要な権限と同じ権限を持っている必要があります。

構文 `DELETE TRANDATA <owner.table>`

引数	説明
`<owner.table>`	表またはファイルの所有者および名前。ワイルドカードは、表名には使用できますが所有者名には使用できません。

例 1 `DELETE TRANDATA finance.acct`

例 2 `DELETE TRANDATA finance.ac*`

INFO SCHEMATRANDATA

INFO SCHEMATRANDATA では、Oracle のスキーマレベルのサプリメンタル・ロギングが、指定したスキーマに対して有効かどうかを確認します。このコマンドを使用する前に、DBLOGIN コマンドを使用してデータベース接続を確立してください。

構文 `INFO SCHEMATRANDATA <schema>`

引数	説明
`<schema>`	サプリメンタル・ロギングを確認するスキーマ。ワイルドカードは使用しないでください。

例 `INFO SCHEMATRANDATA scott`

INFO TRANDATA

INFO TRANDATA では、次の情報を取得します。

- DB2 LUW および z/OS 上の DB2: DATA CAPTURE が有効かどうかを確認します。
- Oracle: サプリメンタル・ロギングが有効かどうか、およびサプリメンタル・ロギングされる列名を表示するかどうかを確認します。すべての列がロギングされた場合、個別の列名のかわりに表記 "ALL" が表示されます。
- Sybase: レプリケーションが有効かどうか、およびすべての LOB 列が (ADD TRANDATA LOB オプションの指定と) 同じ設定を持つかどうかを確認します。

- SQL Server: 拡張ロギングが有効かどうかを確認します。

このコマンドを使用する前に、DBLOGIN コマンドを使用してデータベース接続を確立してください。

引数	説明
<owner.table>	表示するトランザクション情報が含まれる表またはファイルの所有者および名前。DBLOGIN コマンドで指定したユーザーと同じである場合、所有者の指定は不要です。ワイルドカードは、表名には使用できますが所有者名には使用できません。

例 1 `INFO TRANDATA finance.acct`

例 2 `INFO TRANDATA finance.ac*`

チェックポイント表コマンド

チェックポイント表コマンドでは、Oracle GoldenGate がトレイル内の Replicat の現在位置の追跡に使用するチェックポイント表を管理します。チェックポイント表の使用の詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。

コマンドの概要

ADD CHECKPOINTTABLE

CLEANUP CHECKPOINTTABLE

DELETE CHECKPOINTTABLE

INFO CHECKPOINTTABLE

ADD CHECKPOINTTABLE

ADD CHECKPOINTTABLE では、ターゲット・データベースにチェックポイント表を作成します。Replicat は、リカバリで活用するために、この表を使用してトレイル内の読取り位置のレコードを保持します。

チェックポイント表はオプションです。チェックポイントは、ディスク内のファイルにも保持されます。チェックポイント表を使用すると、チェックポイントは Replicat トランザクションの一部になります。これにより Replicat は、特定の状況において、チェックポイントのみを使用するときよりも正確にリカバリできます。

GLOBALS の CHECKPOINTTABLE パラメータで 1 つの表を指定する場合、この表を 1 つの Oracle GoldenGate インスタンス内のすべての Replicat グループのデフォルト・チェックポイント表として使用できます。複数の Oracle GoldenGate のインスタンス (複数のインストール) で同じチェックポイントを使用できます。異なるインスタンスに同じ Replicat グループ名が存在する場合でも、Oracle GoldenGate はチェックポイントを追跡します。詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。

このコマンドを使用する前に、DBLOGIN コマンドを使用してデータベース接続を確立してください。この表の列の名前または属性は変更しないでください。ただし、表記憶域の属性は変更できます。

構文　　`ADD CHECKPOINTTABLE [<owner.table>]`

引数	説明
`<owner.table>`	作成するチェックポイント表の所有者および名前。名前には、引用符、バックスラッシュ、ドル記号、パーセント記号などの特殊文字を含めることはできません。 この表をデフォルトのチェックポイント表として使用し、この表を GLOBALS ファイルの CHECKPOINTTABLE に指定している場合は、所有者と名前を省略できます。 必須ではありませんが、Oracle GoldenGate 専用のスキーマでこの表を作成することをお薦めします。所有者と名前を指定しない場合、デフォルト表は GLOBALS パラメータ・ファイルの CHECKPOINTTABLE パラメータに基づいて作成されます。 統計の表示やこの表の削除で必要になるため、この表の名前を記録しておいてください。

例 1　次の例では、GLOBALS ファイルに指定されているデフォルト名でチェックポイント表を追加します。

```
ADD CHECKPOINTTABLE
```

例 2　次の例では、ユーザー定義名でチェックポイント表を追加します。

```
ADD CHECKPOINTTABLE ggs.fin_check
```

CLEANUP CHECKPOINTTABLE

CLEANUP CHECKPOINTTABLE では、(GGSCI が起動された) ワーキング Oracle GoldenGate ディレクトリ内に、関連するチェックポイント・ファイルが存在しないチェックポイント・レコードをチェックポイント表から削除します。このコマンドの目的は、グループの変更やファイルの移動によって不要になったチェックポイント・レコードを削除することです。

このコマンドを使用する前に、DBLOGIN コマンドを使用してデータベース接続を確立してください。

構文　　`CLEANUP CHECKPOINTTABLE [<owner.table>]`

引数	説明
`<owner.table>`	クリーンアップするチェックポイント表の所有者および名前。所有者と名前を指定しない場合は、GLOBALS パラメータ・ファイルの CHECKPOINTTABLE パラメータで指定されている表が影響を受けます。

例　　`CLEANUP CHECKPOINTTABLE ggs.fin_check`

DELETE CHECKPOINTTABLE

DELETE CHECKPOINTTABLE では、データベースからチェックポイント表を削除します。このコマンドを使用する前に、DBLOGIN コマンドを使用してデータベース接続を確立してください。

関連付けられている Replicat グループがアクティブな状態のときにチェックポイント表の使用を停止するには、次の手順を実行します。

1. GGSCI を実行します。
2. Replicat を停止します。

   ```
   STOP REPLICAT <group>
   ```

3. Replicat グループを削除した後、次のコマンドを使用してこのグループをもう一度追加します。

   ```
   DELETE REPLICAT <group>
   ADD REPLICAT <group>, EXTTRAIL <trail>, NODBCHECKPOINT
   ```

4. GGSCI を終了し、もう一度起動します。
5. Replicat をもう一度起動します。

   ```
   START REPLICAT <group>
   ```

6. DBLOGIN コマンドとデータベースに対して適切な認証オプションを使用して、データベースにログインします。77 ページを参照してください。
7. DELETE CHECKPOINTTABLE を使用してチェックポイント表を削除します。

Replicat が実行中でトランザクションが発生している間にチェックポイント表を削除すると、Replicat は異常終了し、チェックポイント表が見つからないことを示すエラーを返します。ただし、チェックポイントはディスク内のチェックポイント・ファイルに保持されています。処理を再開するには、チェックポイント表を同じ名前でもう一度追加します。トレイル内データのレプリケーションが再開されます。それでもチェックポイント表を削除する必要がある場合は、上記の推奨手順に従ってください。

構文

```
DELETE CHECKPOINTTABLE [<owner.table>] [!]
```

引数	説明
<owner.table>	削除するチェックポイント表の所有者および名前。GLOBALS ファイルの CHECKPOINTTABLE パラメータの指定と同じ場合、所有者と名前は省略できます。
!	表を削除するかどうかを確認するプロンプトをバイパスします。

例

```
DELETE CHECKPOINTTABLE ggs.fin_check
```

INFO CHECKPOINTTABLE

INFO CHECKPOINTTABLE では、チェックポイント表の存在を確認し、作成日時を表示します。次のようなメッセージが返されます。

```
Checkpoint table HR.CHKPT_TBLE created 2011-01-06 11:51:53.
```

このコマンドを使用する前に、DBLOGIN コマンドを使用してデータベース接続を確立してください。

構文

```
INFO CHECKPOINTTABLE [<owner.table>]
```

引数	説明
<owner.table>	チェックポイント表の所有者および名前。GLOBALS ファイルの CHECKPOINTTABLE パラメータの指定と同じ場合、所有者と名前は省略できます。

例 `INFO CHECKPOINTTABLE ggs.fin_check`

Oracle トレース表コマンド

トレース表コマンドでは、Oracle データベースの双方向同期で使用する Oracle GoldenGate トレース表を管理します。Replicat は、各トランザクションの開始時にトレース表に操作を生成します。Extract は、トレース表上の操作で開始されるすべてのトランザクションを無視します。Replicat の操作を無視することで、ソース表とターゲット表間のデータのループバックを防止できます。

双方向同期の詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。

コマンドの概要

ADD TRACETABLE

DELETE TRACETABLE

INFO TRACETABLE

ADD TRACETABLE

ADD TRACETABLE では、Oracle データベースにトレース表を作成します。トレース表は、USERID パラメータで構成した Oracle GoldenGate Extract ユーザーのスキーマにある必要があります。トレース表により、双方向同期構成で Replicat トランザクションがもう一度抽出されることを防止できます。

このコマンドを使用する前に、DBLOGIN コマンドを使用してデータベース接続を確立してください。

次にトレース表の説明を示します。

表 3　トレース表の説明

名前	NULL かどうか	タイプ	説明
GROUP_ID	NOT NULL	VARCHAR2(8)	Replicat グループまたは特別実行プロセスの名前。
DB_USER		VARCHAR2(30)	Replicat グループまたは特別実行プロセスのユーザー ID。
LAST_UPDATE		DATE	トランザクションのタイムスタンプ。

構文　　ADD TRACETABLE [<owner>.<table>]

引数	説明
<owner>.<table>	(オプション) トレース表にデフォルトの GGS_TRACE と異なる名前を指定する場合にのみ使用します。所有者は、Extract パラメータ・ファイルの USERID パラメータで指定した所有者と同一である必要があります。 デフォルト名を使用するには、この引数を省略します。可能なときは常にデフォルトの表名を使用してください。 デフォルトの GGS_TRACE 以外のトレース表名を指定する場合は、Extract および Replicat パラメータ・ファイルの TRACETABLE パラメータを使用して指定します。パラメータ・ファイルで使用するときや、統計の表示およびこの表の削除で必要になるため、この表の名前を記録しておくいてください。詳細は、383 ページの「TRACETABLE	NOTRACETABLE」を参照してください。

例 1　　次の例では、デフォルト名 GGS_TRACE のトレース表を追加します。

```
ADD TRACETABLE
```

例 2　　次の例では、ユーザー定義名 ora_trace を持つトレース表を追加します。

```
ADD TRACETABLE ora_trace
```

DELETE TRACETABLE

DELETE TRACETABLE では、トレース表を削除します。このコマンドを使用する前に、DBLOGIN コマンドを使用してデータベース接続を確立してください。

構文　　DELETE TRACETABLE [<owner.table>]

引数	説明
<owner.table>	削除するトレース表の所有者および名前。所有者が USERID パラメータの指定と同じで、トレース表がデフォルト名の GGS_TRACE を持つ場合は、所有者と名前を省略できます。

例　　DELETE TRACETABLE ora_trace

INFO TRACETABLE

INFO TRACETABLE コマンドでは、指定したトレース表がデータベースのローカル・インスタンスに存在しているかどうかを検証します。表が存在する場合、Oracle GoldenGate は表の名前と作成日時を表示します。表が存在しない場合、Oracle GoldenGate はこの表が存在しないことを示すメッセージを表示します。このコマンドを使用する前に、DBLOGIN コマンドを使用してデータベース接続を確立してください。

構文　　INFO TRACETABLE [<owner.table>]

引数	説明
<owner.table>	検証するトレース表の所有者および名前。所有者が USERID パラメータの指定と同じで、トレース表がデフォルト名の GGS_TRACE を持つ場合は、所有者と名前を省略できます。

例 `INFO TRACETABLE ora_trace`

DDL コマンド

次のコマンドでは、DDL レプリケーションを制御します。

DUMPDDL

DUMPDDL コマンドでは、Oracle GoldenGate DDL 履歴表のデータを表示します。この情報は、Extract プロセスが使用する情報と同じです。これは固有のフォーマットで保管されていますが、判別可能なフォーマットで画面に、または標準 SQL を使用して問合せ可能な SQL 表にエクスポートできます。

DUMPDDL は常に DDL 履歴表のすべてのレコードをダンプします。特定のオブジェクトおよび操作に関する情報を表示するには、SQL 問合せまたは検索を使用して標準出力にリダイレクトします。履歴表には膨大な量のデータが含まれているため、効率的なパフォーマンスを維持するために、各 DDL 文の最初の約 4,000 バイトのみが表示されます。メタデータのフォーマットは文字列ベースです。完全にエスケープされており、表名および列名がネイティブな文字セットでサポートされています。

この情報は、DDL *変更前*トリガーによって提供される履歴データであるため、DDL 変更前のオブジェクトの状態を表します。したがって、CREATE 操作のデータはありません。

注意　このビフォア・トリガーのデフォルト名は、GGS_DDL_TRIGGER_BEFORE です。

DUMPDDL を使用する前に、DBLOGIN コマンドを使用して、履歴表の所有者としてデータベースにログインしてください。

基本 DUMPDDL

基本 DUMPDDL コマンドでは、次の表にメタデータを出力します。

表 4　DUMPDDL 表

表	説明
GGS_DDL_OBJECTS	同期されている DDL 操作のオブジェクトに関する情報。SEQNO は主キーです。ここで説明する他のすべての表には、GGS_DDL_OBJECTS に対する外部キーとなる SEQNO 列が含まれます。
GGS_DDL_COLUMNS	DDL 同期に関連するオブジェクトの列に関する情報。
GGS_DDL_LOG_GROUPS	DDL 同期に関連するサプリメンタル・ログ・グループに関する情報。
GGS_DDL_PARTITIONS	DDL 同期に関連するオブジェクトのパーティションに関する情報。
GGS_DDL_PRIMARY_KEYS	DDL 同期に関連するオブジェクトの主キーに関する情報。

SEQNO 列は、Extract および Replicat レポート・ファイルに表示される DDL 順序番号です。これは、DDL 履歴表 (デフォルト名 GGS_DDL_HIST) に問合せて取得することもできます。

これらの表はすべて、DDL オブジェクトのインストール中に Oracle GoldenGate DDL スキーマとして指定されたスキーマに所有されています (『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください)。これらの表の構造を表示するには、SQL*Plus の DESC コマンドを使用します。

DUMPDDL と SHOW

DUMPDDL と SHOW オプションでは、履歴表に含まれる情報を標準出力フォーマットで画面にダンプします。出力表は作成されません。DDL 履歴表のすべてのレコードが表示されます。

構文

```
DUMPDDL [SHOW]
```

引数	説明
SHOW	DDL 情報を標準出力フォーマットで画面にダンプします。

その他のコマンド

次のコマンドでは、Oracle GoldenGate のその他の様々な機能を制御します。

コマンドの概要

! コマンド

! コマンドでは、以前に実行した GGSCI コマンドを、変更せずに実行します。再実行する前にコマンドに変更を加えるには、FC コマンドを使用します (100 ページを参照してください)。以前に実行したコマンドのリストを表示するには、HISTORY コマンドを使用します (102 ページを参照してください)。

引数なしの ! では、最後に実行したコマンドを実行します。オプションを使用して行番号またはテキスト部分文字列を指定すると、以前に実行した任意のコマンドを実行できます。コマンド履歴はセッション間で保持されないため、以前に実行したコマンドは、現在の GGSI セッション中に発行されたもののみ再実行できます。

構文

```
! [<n> | -<n> | <string>]
```

引数	説明
`<n>`	指定した GGSCI 行からコマンドを実行します。各 GGSCI コマンド行には、セッションの最初を 1 として順序番号が付与されています。
`-<n>`	現在の行から <n> 行前に発行されたコマンドを実行します。
`<string>`	指定した文字列から始まる最新のコマンドを実行します。

例 1

```
! 9
```

例 2

```
! -3
```

例 3

```
! sta
```

ALLOWNESTED

ALLOWNESTED では、ネストされた OBEY ファイルの使用を有効にします。ネストされた OBEY ファイルとは、他の OBEY ファイルが含まれる OBEY ファイルです。ネスト・レベルの最大数は 16 です。デフォルト・モードの NOALLOWNESTED で、ネストされた OBEY ファイルの実行を試みると、次のようなエラーが発生します。

```
ERROR: Nested OBEY scripts not allowed. Use ALLOWNESTED to allow nested scripts.
```

GGSCI セッションを終了すると、次の GGSCI セッションでは設定が NOALLOWNESTED に戻ります。

詳細は、104 ページの「OBEY」を参照してください。

構文

```
ALLOWNESTED | NOALLOWNESTED
```

CREATE SUBDIRS

CREATE SUBDIRS は、Oracle GoldenGate のインストール時に使用します。このコマンドでは、Oracle GoldenGate ホーム・ディレクトリ内にデフォルト・ディレクトリを作成します。CREATE SUBDIRS は、他の構成タスクを実行する前に使用してください。

構文

```
CREATE SUBDIRS
```

FC

FC では、以前に発行した GGSCI コマンドを表示して編集した後、このコマンドを再実行します。以前に実行したコマンドはメモリー・バッファに保持されているので、HISTORY コマンドを発行することによって表示できます (102 ページを参照してください)。

以前に実行したコマンドの表示

引数なしで FC を発行すると、最後に実行したコマンドが表示されます。オプションを使用して行番号またはテキスト部分文字列を指定すると、以前に実行した任意のコマンドを実行できます。コマンド履歴はセッション間で保持されないため、以前に実行したコマンドは、現在の GGSI セッション中に発行されたもののみ編集できます。

コマンドの編集

FC を実行すると、指定したコマンドが表示された後、エディタが開き、2 つのドットで始まる空白行が表示されます。コマンドを編集するには、[Space] キーを使用して表示されているコマンド内の編集開始文字の下にカーソルを動かし、次のいずれかの引数を使用します。引数は大 / 小文字が区別されません。また引数は、組み合せて使用できます。

表 5　FC エディタ・コマンド

引数	説明

i <text>	テキストを挿入します。次に例を示します。
	`GGSCI (SysA) 24> fc 9`
	`GGSCI (SysA) 24> send mgr`
	`GGSCI (SysA) 24..        i childstatus`
	`GGSCI (SysA) 24> send mgr childstatus`

r <text>	テキストを置き換えます。次に例を示します。
	`GGSCI (SysA) 25> fc 9`
	`GGSCI (SysA) 25> info mgr`
	`GGSCI (SysA) 25..     rextract extjd`
	`GGSCI (SysA) 25> info extract extjd`

d	文字を削除します。複数の文字を削除するには、1 文字につき d を 1 つ入力します。次に例を示します。
	`GGSCI (SysA) 26> fc 10`
	`GGSCI (SysA) 26> info extract extjd, detail`
	`GGSCI (SysA) 26..                  ddddddd`
	`GGSCI (SysA) 26> info extract extjd`

<replacement text>	表示されているコマンドを、入力するテキストと 1 対 1 で置き換えます。次に例を示します。
	`GGSCI (SysA) 26> fc 10`
	`GGSCI (SysA) 26> info mgr`
	`GGSCI (SysA) 26..     extract extjd`
	`GGSCI (SysA) 26> info extract extjd`

このコマンドを実行するには、**[Enter]** を 2 回押します。1 回目でエディタを終了し、2 回目でコマンドを発行します。編集をキャンセルするには、フォワード・スラッシュ (/) を 2 回入力します。

構文　`FC [<n> | -<n> | <string>]`

引数	説明
<n>	指定した行からコマンドを表示します。各 GGSCI コマンド行には、セッションの最初を 1 として順序番号が付与されています。
-<n>	現在の行から <n> 行前に発行されたコマンドを表示します。
<string>	指定した文字列から始まる最新のコマンドを表示します。

例 1　`FC 9`

例 2　`FC -3`

例 3　`FC sta`

HELP

HELP では、Oracle GoldenGate コマンドに関する情報を取得します。基本コマンドは、コマンド・カテゴリおよび関連コマンドのリストを返します。<command> オプションでは、特定のコマンドに出力を制限します。

構文 `HELP [<command>]`

引数	説明
<command>	ヘルプを表示するコマンド。

例 `HELP add replicat`

HISTORY

HISTORY では、GGSCI セッションの開始以降で最近発行された GGSCI コマンドのリストを表示します。! コマンド (99 ページ) または FC コマンド (100 ページ) を使用して、リスト内のコマンドを再実行できます。

構文 `HISTORY [<n>]`

例 `HISTORY 7`

引数	説明
<n>	特定の最近のコマンドを返します。<n> には、任意の正の数を指定します。

このコマンドの結果は次のようになります。

```
1: start manager
2: status manager
3: info manager
4: send manager childstatus
5: start extract extjd
6: info extract extjd
7: history
```

INFO ALL

INFO ALL では、システム上のすべての Manager、Extract および Replicat プロセスのステータスおよび (関連する場合は) ラグを表示します。オプションなしの基本コマンドでは、オンライン (継続的な) プロセスのみを表示します。タスクを表示するには、INFO ALL TASKS または INFO ALL ALLPROCESSES を使用します。

Status フィールドおよび **Lag at Chkpt** (チェックポイント) フィールドは、INFO EXTRACT コマンドおよび INFO REPLICAT コマンドと同じプロセス・ステータスおよびラグを表示します。

図 10 INFO ALL 出力の例

```
Program     Status      Group       Lag at Chkpt  Time Since Chkpt
MANAGER     RUNNING
EXTRACT     ABENDED     EXTCUST     00:00:00      96:56:14
EXTRACT     STOPPED     INITDL
EXTRACT     STOPPED     INITDBL
```

構文 `INFO ALL [TASKS | ALLPROCESSES]`

引数	説明
TASKS	タスクの情報のみ表示します。
ALLPROCESSES	オンライン・プロセスおよびタスクの情報を表示します。

例 1 `INFO ALL TASKS`

例 2 `INFO ALL ALLPROCESSES`

INFO MARKER

INFO MARKER では、NonStop システムの最近処理されたマーカーを確認します。GGSCI、Logger、Extract または Replicat によるマーカー処理ごとに 1 つレコードが表示されます。

マーカーは、HP NonStop ソフトウェア用の Oracle GoldenGate for NonStop を使用してのみ NonStop システムで追加できます。

次に出力の例を示します。

```
Processed             Added                Diff     Prog    Group    Node
2012-02-16:14:41:15   2012-02-16:14:41:08 00:00:07 Extract PQACMD   \QAMD
                      GROUPCMD REPLICAT RQACMD CLOSEFILES
2012-02-16:14:41:13   2012-02-16:14:41:08 00:00:05 Extract PQACMD   \QAMD
                      TACLCMD REPLICAT RQACMD FUP PURGEDATA $QA16.QAETAR
```

条件 :

- **Processed** は、マーカーが処理されたローカル時間です。
- **Added** は、マーカーが NonStop オーディット・トレイルまたはログ・トレイルに挿入されたローカル時間です。
- **Diff** は、Processed 値と Added 値の時間差です。Diff は、ユーザー・アプリケーションと Extract および Replicat のアクティビティのラグを示すインジケータとして活用できます。
- **Prog** は、GGSCI、Logger、Extract または Replicat など、マーカーを処理したプロセスです。
- **Group** は、マーカーを処理した Extract または Replicat グループ、または Logger プロセスです。GGSCI がマーカーを処理した場合は、N/A が表示されます。
- **Node** は、オーディット・トレイル内のマーカーの挿入先ノードです。
- ADD MARKER 文にユーザー定義テキストが含まれていた場合は、追加の列が表示されることがあります。

構文　　INFO MARKER [COUNT <num items>]

引数	説明
COUNT <num items>	リストへの出力を特定の数の最新のマーカーに制限します。

OBEY

OBEY では、Oracle GoldenGate コマンドのリストを含むファイルを処理します。OBEY は、特定の順序で頻繁に使用されるコマンドを実行するときに便利です。

OBEY ファイルは、他の OBEY ファイルから呼出しできます。こうしたファイルをネストされた OBEY ファイルと呼びます。最大で 16 の OBEY ファイルをネストできます。ネストされた OBEY ファイルを使用するには、まず ALLOWNESTED コマンドを発行してこの機能を有効化する必要があります。100 ページを参照してください。

構文　　OBEY <file name>

引数	説明
<file name>	コマンドのリストを含むファイルの相対パス名または完全修飾パス名。

例 1　　obey ./mycommands.txt

上記のコマンドでは、次のようなファイルを実行します。

```
add extract fin, tranlog, begin now
add exttrail dirdat/aa, extract fin
add extract hr, tranlog, begin now
add exttrail dirdat/bb, extract hr
start extract *
info extract *, detail
```

例 2　　次に、ネストされた OBEY ファイルの例を示します。OBEY ファイル名を addcmds.txt とします。このファイル内には、startcmds.txt という名前の OBEY ファイルを呼び出して別のコマンド・セットを実行する、別の OBEY コマンドも含まれています。

```
obey ./addcmds.txt
```

(このコマンドは次のことを実行します)

```
add extract fin, tranlog, begin now
add exttrail ggs/dirdat/aa, extract fin
add extract hr, tranlog, begin now
add exttrail ggs/dirdat/bb, extract hr
add replicat fin2, exttrail ggs/dirdat/aa, begin now
add replicat hr2, exttrail ggs/dirdat/bb, begin now
obey ./startcmds.txt
```

(ネストされた startcmds.txt ファイルは次のことを実行します)

```
start extract *
info extract *, detail
start replicat *
info replicat *, detail
```

SHELL

SHELL では、GGSCI インタフェース内からシェル・コマンドを実行します。

構文

```
SHELL <command>
```

引数	説明
<command>	実行するシステム・コマンド。

例 1

```
SHELL dir dirprm\*
```

例 2

```
SHELL rm ./dat*
```

SHOW

SHOW では、Oracle GoldenGate 環境を表示します。

図 11 SHOW の表示例

```
Parameter settings:
SET SUBDIRS    ON
SET DEBUG      OFF
Current directory: C:\GG_81
Using subdirectories for all process files
Editor:  notepad
Reports (.rpt)                C:\GG_81\dirrpt
Parameters (.prm)             C:\GG_81\dirprm
Replicat Checkpoints (.cpr)   C:\GG_81\dirchk
Extract Checkpoints (.cpe)    C:\GG_81\dirchk
Process Status (.pcs)         C:\GG_81\dirpcs
SQL Scripts (.sql)            C:\GG_81\dirsql
Database Definitions (.def)   C:\GG_81\dirdef
```

構文

```
SHOW
```

VERSIONS

VERSIONS では、オペレーティング・システムおよびデータベースのバージョン情報を表示します。ODBC 接続の場合は、ドライバ・バージョン情報も表示されます。データベース情報を表示するために、まず DBLOGIN コマンドを発行してデータベース接続を確立してください。

構文

```
VERSIONS
```

VIEW GGSEVT

VIEW GGSEVT では、Oracle GoldenGate エラー・ログ (ggserr.log ファイル) を表示します。このファイルには、プロセス起動、シャットダウン、例外条件などの Oracle GoldenGate イベントに関する情報が含まれています。この情報はシステム・エラー・ログにも記録されますが、Oracle GoldenGate エラー・ログのほうが簡単に表示でき、またシステム・エラー・ログよりも過去に遡った情報が保持されていることもあります。

表示される情報は長くなることがあります。最後に到達する前に表示を終了するには、オペレーティング・システムの標準の画面出力終了方法を使用してください。

図 12 VIEW GGSEVT の出力例

```
2011-01-08 11:20:56  GGS INFO     301  GoldenGate Manager for Oracle,
mgr.prm:  Command received from GUI (START GGSCI ).
2011-01-08 11:20:56  GGS INFO     302  GoldenGate Manager for Oracle,
mgr.prm:  Manager started GGSCI process on port 7840.
2011-01-08 11:21:31  GGS INFO     301  GoldenGate Manager for Oracle,
mgr.prm:  Command received from GUI (START GGSCI ).
```

構文

```
VIEW GGSEVT
```

VIEW REPORT

VIEW REPORT では、Extract または Replicat が生成したプロセス・レポートを表示します。このレポートには、プロセス・パラメータ、実行統計、エラー・メッセージ、その他の診断情報がリストされます。

このコマンドでは、現在のレポートのみ表示します。プロセスが起動されるたびにレポートはエージングされます。過去のレポートには、順序番号が付けられます (例 : finance0.rpt、finance1.rpt)。過去のレポートを表示するには、[<n>] オプションを使用します。

図 13 出力例

```
***********************************************************************
** Running with the following parameters **
***********************************************************************
sourceisfile
USERID ogg, PASSWORD AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC, &
    AES128, ENCRYPTKEY securekey1
rmthost sys1, mgrport 8040
rmtfile /home/jdad/ggsora/dirdat/tcustord.dat, purge
table tcustord;
rmtfile /home/jdad/ggsora/dirdat/tcustmer.dat, purge
table tcustmer;
```

```
Processing table TCUSTORD
Processing table TCUSTMER

***********************************************************************
* ** Run Time Statistics ** *
***********************************************************************

Report at 2011-01-13 11:07:36 (activity since 2011-01-13 11:07:31)

Output to /home/jdad/ggsora/dirdat/tcustord.dat:

From Table TCUSTORD:
       #       inserts:         2
       #       updates:         0
       #       deletes:         0
       #      discards:         0

Output to /home/jdad/ggsora/dirdat/tcustmer.dat:

From Table TCUSTMER:
       #       inserts:         2
       #       updates:         0
       #       deletes:         0
       #      discards:         0
```

構文

```
VIEW REPORT {<group name>[<n>] | <file name>}
```

引数	説明
<group name>	グループ名。このコマンドは、Oracle GoldenGate dirrpt サブディレクトリの <group>.rpt という名前のファイルをレポート・ファイルとみなします。
<n>	過去のレポートの番号。レポート・ファイルには、0(最新) ～ 9(最も古い) までの番号が付けられています。
<file name>	ファイルの相対パス名または完全パス名 (例 : c:\ggs\dirrpt\orders.rpt)。

例 1 次の例では、orders グループの過去のレポート・ファイル (番号 3) を表示します。

```
VIEW REPORT orders3
```

例 2 次の例では、ファイル名で指定した特定のレポートを表示します。

```
VIEW REPORT dirrpt\orders.rpt
```

第 2 章
Oracle GoldenGate パラメータの概要

この章では、Oracle GoldenGate のすべての構成パラメータの概要について説明します。オンラインで表示すると、各パラメータ名はそのパラメータの詳細ドキュメントにリンクしています。

パラメータ・カテゴリ

Oracle GoldenGate パラメータは、次のカテゴリに分類されます。

GLOBALS パラメータ

Manager パラメータ

Extract と Replicat に共通のパラメータ

Extract パラメータ

Replicat パラメータ

DEFGEN パラメータ

GLOBALS パラメータ

GLOBALS ファイルは、特定のプロセス実行時パラメータではなく、Oracle GoldenGate インスタンス全体に関連するパラメータを保持します。

表 6　すべての GLOBALS パラメータ

パラメータ	説明
CHARSET	パラメータ・ファイルの読取り時にオペレーティング・システムのデフォルトのかわりに使用するプロセスのマルチバイト・キャラクタ・セットを指定します。
CHECKPOINTTABLE	デフォルトのチェックポイント表を指定します。
DDLTABLE	Oracle の DDL 同期をサポートする DDL 履歴表の非デフォルト名を指定します。
ENABLEMONITORING	Oracle GoldenGate Monitor での Oracle GoldenGate インスタンスの表示および監視を有効化します。
GGSCHEMA	Oracle の DDL 同期をサポートするデータベース・オブジェクトを含むスキーマ名を指定します。

表 6　すべての GLOBALS パラメータ (続き)

パラメータ	説明
MARKERTABLE	Oracle の DDL 同期をサポートする DDL マーカー表の非デフォルト名を指定します。
MGRSERVNAME	Windows サービスとしてインストールされている場合に Manager プロセス名を指定します。
OUTPUTFILEUMASK	Oracle GoldenGate プロセスがトレイル・ファイルおよび破棄ファイルの作成に使用できる umask を指定します。
USEANSISQLQUOTES	引用符で囲まれたオブジェクト名とリテラルに対して SQL-92 ルールを有効にします。
SYSLOG	システム・ログに書き込まれる Oracle GoldenGate メッセージのタイプをフィルタします。
TRAILCHARSET	トレイルのバージョンが古く、ソース文字セットが格納されない場合、またはトレイルに格納されている文字セットより優先させる場合、ソース・データの文字セットを指定します。
USEIPV6	Oracle GoldenGate で TCP/IP 接続に IPv6 を使用させます。

Manager パラメータ

Manager は Oracle GoldenGate の親プロセスで、自身のプロセス、リソース、ユーザー・インタフェース、しきい値およびエラーのレポートを管理します。ほとんどの場合、Manager はデフォルト設定で問題なく機能します。

表 7　Manager パラメータ : 一般

パラメータ	説明
CHARSET	パラメータ・ファイルの読取り時にオペレーティング・システムのデフォルトのかわりに使用するプロセスのマルチバイト・キャラクタ・セットを指定します。
COMMENT | --	パラメータ・ファイルにコメントを挿入します。
SOURCEDB	ログイン情報の一部としてデータ・ソース名を指定します。
USERID	データベースにアクセスするときに必要なログイン情報を Manager に提供します。
SYSLOG	システム・ログに書き込まれる Oracle GoldenGate メッセージのタイプをフィルタします。

表 8　Manager パラメータ : ポート管理

パラメータ	説明
DYNAMICPORTLIST	Collector が動的に割り当て可能なポートを指定します。
PORT	Manager がリクエストをリスニングする TCP/IP ポート番号を指定します。

表 9　Manager パラメータ : プロセス管理

パラメータ	説明
AUTORESTART	障害発生後に Manager によって再起動されるプロセスを指定します。
AUTOSTART	Manager の起動時に起動されるプロセスを指定します。
BOOTDELAYMINUTES	システムのブート後、Manager がメイン処理アクティビティを実行するまでの遅延時間を決定します。このパラメータは、Windows をサポートします。
UPREPORT	プロセス・ハートビート・メッセージをレポートする間隔を決定します。

表 10　Manager パラメータ : イベント管理

パラメータ	説明
DOWNCRITICAL	正常または異常終了したプロセスをレポートします。
DOWNREPORT	停止したプロセスをレポートする間隔を制御します。
LAGCRITICAL	クリティカルとみなすラグしきい値を指定し、このしきい値に到達したときにとエラー・ログに警告を生成します。
LAGINFO	情報メッセージをエラー・ログにレポートするラグしきい値を指定します。
LAGREPORT	ラグ・タイムをエラー・ログにレポートする間隔を設定します。

表 11　Manager パラメータ : メンテナンス

パラメータ	説明
CHECKMINUTES	Manager がメンテナンス・アクティビティを実行する間隔を決定します。
PURGEDDLHISTORY	Oracle DDL 履歴表から不要になった行をパージします。
PURGEDDLHISTORYALT	表 ID に関連付けられているパーティション ID を追跡する代替 Oracle DDL 履歴表から行をパージします。
PURGEMARKERHISTORY	不要になった Oracle マーカー表の行をパージします。
PURGEOLDEXTRACTS	不要になったトレイル・データをパージします。
PURGEOLDTASKS	指定した時間の経過後に Extract および Replicat タスクをパージします。
STARTUPVALIDATIONDELAY[CSECS]	Manager が起動後のプロセスが実行中かどうかを確認するまでの遅延時間を設定します。

Extract と Replicat に共通のパラメータ

これらのパラメータは、Extract および Replicat プロセスの両方で使用できます。

表 12　Extract と Replicat に共通のパラメータ : 一般

パラメータ	説明
CHARSET	パラメータ・ファイルの読取り時にオペレーティング・システムのデフォルトのかわりに使用するプロセスのマルチバイト・キャラクタ・セットを指定します。
CHECKPARAMS	パラメータ・ファイルの構文を検証します。
COMMENT | --	パラメータ・ファイルのコメントを示します。
GETENV	SETENV パラメータで設定された変数を取得します。
OBEY	異なるパラメータ・ファイルに含まれているパラメータ文を処理します。
SETENV	GGSCI インタフェース内から UNIX 環境変数の値を指定します。

表 12　Extract と Replicat に共通のパラメータ : 一般 (続き)

パラメータ	説明
TRACETABLE | NOTRACETABLE	Replicat がターゲット・データベースを更新するたびにレコードを追加するトレース表を指定します。Replicat によって生成されたデータベース変更を Extract に無視させます。Oracle をサポートします。Oracle の双方向レプリケーションをサポートします。
USERID	データベース接続情報を指定します。

表 13　Extract と Replicat に共通のパラメータ : データの選択、変換およびマッピング

パラメータ	説明
ASCIITOEBCDIC	UNIX システム・サービスを実行する z/OS システムの DB2 用に ASCII テキストを EBCDIC に変換します。
COLMATCH	グローバル列マッピング・ルールを作成します。
DDL	DDL 操作の取得を有効化およびフィルタします。
BINARYCHARS | NOBINARYCHARS	バイナリ文字が NULL 終了文字列として処理されるかどうかを制御します。
DDLSUBST	DDL 処理で文字列を置き換えます。
GETDELETES | IGNOREDELETES	削除操作の抽出を制御します。
GETINSERTS | IGNOREINSERTS	挿入操作の抽出を制御します。
GETTRUNCATES | IGNORETRUNCATES	切捨て文の抽出を制御します。
GETUPDATEAFTERS | IGNOREUPDATEAFTERS	アフター・イメージの抽出を制御します。
GETUPDATEBEFORES | IGNOREUPDATEBEFORES	ビフォア・イメージの抽出を制御します。
GETUPDATES | IGNOREUPDATES	更新操作の抽出を制御します。
REPLACEBADCHAR	無効な文字値を別の値に置き換えます。
SOURCEDEFS	DEFGEN ユーティリティによって作成されたソース・データ定義を含むファイルを指定します。
TRIMSPACES | NOTRIMSPACES	CHAR 列を VARCHAR 列にマッピングする際に、末尾の空白を切り捨てるかどうかを制御します。

表 13　Extract と Replicat に共通のパラメータ : データの選択、変換およびマッピング (続き)

パラメータ	説明
VARWIDTHNCHAR | NOVARWIDTHNCHAR	NCHAR 列の長さ情報をトレイルに書き込むかどうかを制御します。
WILDCARDRESOLVE	TABLE 文でワイルドカードで指定されている表を処理するルールを定義します。

表 14　Extract と Replicat に共通のパラメータ : カスタム処理

パラメータ	説明
CUSEREXIT	処理中にユーザー・イグジット・ルーチンを起動します。
INCLUDE	マクロ・ライブラリを起動します。
MACRO	Oracle GoldenGate マクロを定義します。
MACROCHAR	デフォルトの # 以外のマクロ文字を定義します。
SQLEXEC	Extract 処理中にストアド・プロシージャまたは問合せを実行します。

表 15　Extract と Replicat に共通のパラメータ : レポート

パラメータ	説明
CMDTRACE	レポート・ファイルにマクロ展開手順を表示します。
LIST | NOLIST	レポート・ファイルにマクロのリストを表示するかどうかを制御します。
REPORT	統計レポートをスケジュールします。
STATOPTIONS	統計表示に含める情報を指定します。
REPORTCOUNT	処理されたレコード数をレポートします。
TRACE | TRACE2	プロセスのボトルネックの把握に役立つ処理情報を表示します。

表 16　Extract と Replicat に共通のパラメータ : チューニング

パラメータ	説明
ALLOCFILES	NUMFILES の値に到達した後に割り当てる増分メモリー構造数を制御します。
CHECKPOINTSECS	プロセスのチェックポイント書込み頻度を制御します。
DBOPTIONS	データベース・オプションを指定します。
DDLOPTIONS	DDL 処理オプションを指定します。
DYNAMICRESOLUTION | NODYNAMICRESOLUTION	Extract が表のトランザクション・データを検出するまで、その表のメタデータ参照を抑止します。多数の表を同期するときに、Extract の処理開始を高速化します。
EOFDELAY | EOFDELAYCSECS	プロセスがデータ・ソース内で処理する新しいデータを探すまでの待機時間を指定します。
FUNCTIONSTACKSIZE	Oracle GoldenGate ファンクションの処理に使用するメモリー・スタックのサイズを制御します。
NUMFILES	Oracle GoldenGate が処理する表に関する情報を格納する専用メモリーの初期割当てを制御します。

表 17　Extract と Replicat に共通のパラメータ : エラー処理

パラメータ	説明
DDLERROR	DDL 抽出のエラー処理を制御します。
DISCARDFILE	処理できなかったレコードを記録します。

表 18　Extract と Replicat に共通のパラメータ : メンテナンス

パラメータ	説明
DISCARDROLLOVER	新しい破棄ファイルを作成する頻度を制御します。
PURGEOLDEXTRACTS	古いトレイル・ファイルをパージします。
REPORTROLLOVER	新しいレポート・ファイルをいつ作成するか指定します。
DECRYPTTRAIL	トレイルまたは抽出ファイルのデータを復号化します。

Extract パラメータ

Extract プロセスは、構成パラメータに応じて完全なデータ・レコードまたはトランザクション・データの変更を取得した後、ターゲット表に適用されるかロード・ユーティリティなどの別のプロセスによって処理されるデータをターゲット・システムに送信します。

表 19　Extract パラメータ : 一般

パラメータ	説明
ETOLDFORMAT	Oracle GoldenGate リリース 6.0 より前の Replicat バージョンと互換性のあるフォーマットでトレイルを生成します。
RECOVERYOPTIONS	Extract プロセスのリカバリ・モードを制御します。
SOURCEDB	ログイン情報の一部としてデータ・ソースを指定します。
TCPSOURCETIMER | NOTCPSOURCETIMER	送信元および送信先システムの時間が異なるときに、送信されるレコードを送信先システムのタイムスタンプに合わせて調整します。

表 20　Extract パラメータ : 処理方法

パラメータ	説明
DSOPTIONS	Teradata Access Module (TAM) を使用する場合の Extract 処理オプションを指定します。
EXTRACT	Extract グループをオンライン・プロセスとして定義します。
GETAPPLOPS | IGNOREAPPLOPS	Replicat 以外のすべてのプロセスの操作をトレイルまたはファイルに書き込むかどうかを制御します。
GETREPLICATES | IGNOREREPLICATES	レプリケートされた操作を、同じシステム上で動作している Extract に取得させるかどうかを制御します。
PASSTHRU | NOPASSTHRU	パススルー・モードのデータポンプ Extract が表をパススルーモードで処理するか、またはデータ定義を使用するかを制御します。
RMTTASK	リモート・システムに処理タスクを作成します。
SOURCEISTABLE	ソース表からレコード全体を抽出します。
VAM	Teradata Access Module (TAM) を使用して Extract プロセスにトランザクション・データを提供することを示します。

表 21　Extract パラメータ : データの選択、変換およびマッピング

パラメータ	説明
COMPRESSDELETES | NOCOMPRESSDELETES	Oracle GoldenGate が削除操作のときにトレイルにキーのみを書き込むかすべての列を書き込むかを制御します。
COMPRESSUPDATES | NOCOMPRESSUPDATES	更新操作のときに主キー列および変更された列のみを書き込みます。
FETCHOPTIONS	Oracle GoldenGate のデータ・フェッチ方法を制御します。
SEQUENCE	同期する順序を指定します。
Extract 用 TABLE	抽出する表を指定し、列マッピングおよび変換を制御します。
TABLEEXCLUDE	抽出プロセスから表を除外します。
TARGETDEFS	NonStop プラットフォーム上のターゲット・データベースのターゲット表定義を含むファイルを指定します。
TRAILCHARSETASCII	ASCII と EBCDIC の両方の表がある場合、z/OS の DB2 から取得されたデータに対して ASCII 文字セットを指定します。
TRAILCHARSETEBCDIC	ASCII と EBCDIC の両方の表がある場合、z/OS の DB2 から取得されたデータに対して EBCDIC 文字セットを指定します。

表 22　Extract パラメータ : データ・ルーティング

パラメータ	説明
EXTFILE	ローカル・システム上に、抽出されたデータを書き込む抽出ファイルを指定します。
EXTTRAIL	ローカル・システム上に、抽出されたデータを書き込むトレイルを指定します。
RMTFILE	リモート・システム上に、抽出されたデータを書き込む抽出ファイルを指定します。
RMTHOST	ターゲット・システムおよび Manager ポート番号を指定します。
RMTTRAIL	リモート・システム上に、抽出されたデータを書き込むトレイルを指定します。

表 23　Extract パラメータ : データ・フォーマット

パラメータ	説明
FORMATASCII	抽出データを外部 ASCII フォーマットでフォーマットします。
FORMATSQL	抽出データを SQL 文にフォーマットします。
FORMATXML	抽出データを XML 構文にフォーマットします。
NOHEADERS	レコード・ヘッダーのトレイルへの書き込みを防ぎます。

表 24　Extract パラメータ : チューニング

パラメータ	説明
BR	Extract の Bounded Recovery 機能を制御します。
CACHEMGR	仮想メモリー・キャッシュ・マネージャを制御します。
FLUSHSECS | FLUSHCSECS	レコード・データがトレイルに書き込まれるまでバッファに保持される時間を指定します。
LOBMEMORY	LOB を含むトランザクションのキャッシングに使用できるメモリーおよび一時ディスク領域の量を制御します。
MAXFETCHSTATEMENTS	Extract がデータベースからのデータのフェッチにできる、準備済問合せの最大数を制御します。
RMTHOSTOPTIONS	パッシブ Extract グループが使用する TCP/IP 接続のホスト情報以外の接続属性を指定します。
THREADOPTIONS	Extract の Oracle Real Application Cluster 環境での動作方法を制御します。Oracle をサポートします。
TRANLOGOPTIONS	取得処理オプションを指定します。
TRANSMEMORY	コミットされていないトランザクション・データのキャッシングに使用できるメモリーおよび一時ディスク領域の量を制御します。
WARNLONGTRANS	長時間におよぶトランザクションを定義し、これらのトランザクションを確認およびレポートする間隔を制御します。

表 25　Extract パラメータ : メンテナンス

パラメータ	説明
ROLLOVER	トレイル・ファイルをエージングする方法を指定します。

表 26　Extract パラメータ : セキュリティ

パラメータ	説明
ENCRYPTTRAIL | NOENCRYPTTRAIL	トレイルまたは抽出ファイルのデータの暗号化を制御化します。

Replicat パラメータ

Replicat プロセスは、Extract プロセスによって抽出されたデータを読み取り、このデータをターゲット表に適用するか、ロード・ユーティリティなどの他のアプリケーションがこのデータを使用できるように準備を整えます。

表 27　Replicat パラメータ : 一般

パラメータ	説明
TARGETDB	ログイン情報の一部としてデータ・ソースを指定します。

表 28　Replicat パラメータ : 処理方法

パラメータ	説明
BEGIN	Replicat 処理の開始位置を指定します。SPECIALRUN を指定するときに指定する必要があります。
BULKLOAD	Oracle SQL*Loader ユーティリティのインタフェースに直接データをロードします。
END	Replicat 処理の停止位置を指定します。SPECIALRUN を使用するときに指定する必要があります。
GENLOADFILES	データベース・ロード・ユーティリティと互換性のある実行ファイルと制御ファイルを生成します。
REPLICAT	オンライン変更同期を行う Replicat グループを指定します。
SPECIALRUN	実行間でチェックポイントを必要としないワンタイム処理タスクで使用します。

表 29　Replicat パラメータ : データの選択、変換およびマッピング

パラメータ	説明
ALLOWDUPTARGETMAP | NOALLOWDUPTARGETMAP	パラメータ・ファイル内でソースとターゲットが同一の MAP 文を複数回使用することを許可します。
ALLOWNOOPUPDATES | NOALLOWNOOPUPDATES	Replicat が "no-op" 操作に対応する方法を制御します。no-op 操作は、ターゲット表では無効な操作です。
APPLYNOOPUPDATES | NOAPPLYNOOPUPDATES	SET 句および WHERE 句の両方ですべての列を使用して、"no-op" 更新を適用させます。
ASSUMETARGETDEFS	ソース表とターゲット表が同一の列構造を持つとみなします。
CHARSETCONVERSION | NOCHARSETCONVERSION	データの適用時に、Replicat がソース・データの文字セットをターゲット・データベースの文字セットに変換するかどうかを制御します。
INSERTALLRECORDS	レコードに対する各変更操作を、新しいレコードとしてターゲット表に挿入します。
INSERTDELETES | NOINSERTDELETES	削除を挿入に変換します。
INSERTMISSINGUPDATES | NOINSERTMISSINGUPDATES	ターゲット行が存在しない場合に、更新を挿入に変換します。
INSERTUPDATES | NOINSERTUPDATES	更新を挿入に変換します。
Replicat 用 MAP	1 つ以上のソース表およびターゲット表の関係を指定し、列マッピングおよび変換を制御します。
MAPEXCLUDE	MAP 文でワイルドカードで指定されている表を処理から除外します。
SPACESTONULL | NOSPACESTONULL	空白のみを含むターゲット列を NULL に変換するかどうかを制御します。
Replicat 用 TABLE	行がフィルタ基準を満たしたときにイベント・アクションの対象にする 1 つまたは複数の表を指定します。
UPDATEDELETES | NOUPDATEDELETES	削除を更新に変換します。

表 29　Replicat パラメータ : データの選択、変換およびマッピング (続き)

パラメータ	説明
UPDATEINSERTS | NOUPDATEINSERTS	挿入を更新に変換します。

表 30　Replicat パラメータ : データ・ルーティング

パラメータ	説明
EXTFILE	レプリケートするデータを含むローカル・システム上の抽出ファイル名を定義します。ワンタイム処理で使用します。
EXTTRAIL	レプリケートするデータを含むトレイルを定義します。ワンタイム処理で使用します。

表 31　Replicat パラメータ : エラー処理およびレポート

パラメータ	説明
HANDLECOLLISIONS | NOHANDLECOLLISIONS	重複レコードおよび不明レコードのエラーを処理します。
HANDLETPKUPDATE	一時主キー更新のレプリケートに関連する制約エラーを防止します。
OVERRIDEDUPS | NOOVERRIDEDUPS	重複レコード・エラーが発生するたびに、既存のターゲット・レコードをレプリケートされた挿入レコードで上書きします。
RESTARTCOLLISIONS | NORESTARTCOLLISIONS	競合のために Oracle GoldenGate が異常終了した後に、Replicat が HANDLECOLLISIONS ロジックを適用するかどうかを制御します。
REPERROR	Replicat がデータベース・エラーに対応する方法を決定します。
REPFETCHEDCOLOPTIONS	データベースからのフェッチを必要とする操作に対する Replicat のレスポンス方法を決定します。
SHOWSYNTAX	Replicat の SQL 文をレポート・ファイルに出力します。
SQLDUPERR	重複レコードを示すデータベース・エラー番号を指定します。OVERRIDEDUPS とともに使用します。
WARNRATE	データベース・エラーをレポートする頻度を指定します。

表 32　Replicat パラメータ : チューニング

パラメータ	説明
BATCHSQL	同様の SQL 文を配列にまとめ、より高速に適用することによって、Replicat 処理のスループットを向上させます。
DEFERAPPLYINTERVAL	レプリケートされた操作をターゲット・データベースに適用するまで Replicat が待機する時間を指定します。
DYNSQL | NODYNSQL	1 回のコンパイルで何回も同じ文を使用する方法のかわりに、Replicat にリテラル SQL 文を使用させます。
GROUPTRANSOPS	1 つの Replicat トランザクションにグループ化されるレコード数を制御します。
INSERTAPPEND | NOINSERTAPPEND	INSERT 操作を Oracle ターゲット表に適用するときに、Replicat が APPEND ヒントを使用するかどうかを制御します。
MAXDISCARDRECS	破棄ファイルにレポートする破棄レコード数を制限します。
MAXSQLSTATEMENTS	Replicat が使用できる準備済 SQL 文の数を制御します。
MAXTRANSOPS	大きなソース・トランザクションをターゲット・システム上で小さなトランザクションに分割します。
NUMFILES	Oracle GoldenGate が処理する表に関する情報を格納する専用メモリーの初期割当てを制御します。
RETRYDELAY	失敗した SQL 操作を再試行するまでの遅延を指定します。
TRANSACTIONTIMEOUT	Replicat が、オープンしているターゲット・トランザクションのコミット、およびソース・トランザクション全体の適用準備ができるまで保持しているターゲット・トランザクション内の未完了のソース・トランザクションのロールバックを実行するまで待機する時間を指定します。

DEFGEN パラメータ

DEFGEN では、ソース表またはターゲット表のデータ定義を含むファイルを作成します。データ定義は、ソース表とターゲット表が異なる定義を持つ場合、またはデータベースのタイプが異なる場合に必要です。

表 33　すべての DEFGEN パラメータ

パラメータ	説明
CHARSET	パラメータ・ファイルの読取り時にオペレーティング・システムのデフォルトのかわりに使用するプロセスのマルチバイト・キャラクタ・セットを指定します。
DEFSFILE	DEFGEN が定義を書き込むファイルの名前を指定します。
SOURCEDB	ログイン情報の一部としてデータ・ソースを指定します。
DEFGEN 用 TABLE	定義を取得する表を指定します。
USERID	データベース接続情報を指定します。

DDL パラメータ

これらのパラメータは、Oracle GoldenGate の DDL サポートを制御します。DDL サポートには他のパラメータが必要な場合がありますが、ここに示すものは特に DDL 機能を処理します。

表 34　すべての DDL パラメータ

パラメータ	説明
DDL	DDL サポートを有効化し、DDL をフィルタします。
DDLERROR	DDL レプリケーション中に発生するエラーを処理します。
DDLOPTIONS	フィルタリングおよび文字列置換以外の DDL レプリケーションを構成します。
DDLSUBST	DDL 操作で文字列の置換えを有効化します。
DDLTABLE	DDL 履歴表の代替名を指定します。
GGSCHEMA	DDL レプリケーションをサポートするオブジェクトを含むスキーマ名を指定します。
PURGEDDLHISTORY	DDL 履歴表のサイズを制御します。
PURGEMARKERHISTORY	DDL マーカー表のサイズを制御します。

第 3 章
Oracle GoldenGate パラメータ

この章では、Oracle GoldenGate プロセスの構成、実行および管理にユーザーが使用可能なすべての Oracle GoldenGate パラメータについて説明します。

Oracle GoldenGate パラメータ・ファイルの作成、変更および格納方法は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。

ALLOCFILES

適用対象 Extract および Replicat

ALLOCFILES では、NUMFILES パラメータで指定されている初期メモリー割当てに到達した後に割り当てる追加のメモリー構造数を制御します (286 ページを参照してください)。この 2 つのパラメータによって、処理中のソース表およびターゲット表に関する情報を保持するプロセス・メモリーの割当て方法を制御します。

システム・リソースが許可するメモリーが、プロセスによって必要に応じて割り当てられるので、NUMFILES と ALLOCFILES は、ともにデフォルト値で問題なく機能するはずです。

ALLOCFILES は、すべての TABLE または MAP エントリが有効になる前に指定する必要があります。

デフォルト 500

構文

```
ALLOCFILES <number of structures>
```

引数	説明
`<number of structures>`	割り当てる追加のメモリー構造数。メモリーの無駄な消費を防ぐため、ALLOCFILES を不必要に高い数に設定しないでください。Oracle GoldenGate のメモリー構造は、200 万表までサポートします。

例

```
ALLOCFILES 1000
```

ALLOWDUPTARGETMAP | NOALLOWDUPTARGETMAP

適用対象 Replicat

ALLOWDUPTARGETMAP および NOALLOWDUPTARGETMAP パラメータでは、1 つのパラメータ・ファイル内で、同一のソースおよびターゲット・オブジェクトに対する重複する MAP 文を受け入れるかどうかを制御します。たとえば、ALLOWDUPTARGETMAP が有効な場合、次のパラメータ・ファイルは許可されます。

```
REPLICAT repcust
USERID ogg, PASSWORD AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC &
    AES128, ENCRYPTKEY securekey1
SOURCEDEFS /ggs/dirdef/source.def
ALLOWDUPTARGETMAP
GETINSERTS
GETUPDATES
IGNOREDELETES
MAP ggs.tcustmer, TARGET ggs.tcustmer, COLMAP (USEDEFAULTS, deleted_row = "N");
IGNOREINSERTS
IGNOREUPDATES
GETDELETES
UPDATEDELETES
MAP ggs.tcustmer, TARGET ggs.tcustmer, COLMAP (USEDEFAULTS,
deleted_row = "Y");
```

ALLOWDUPTARGETMAP が指定されず、同じソース表およびターゲット表が複数回マップされる場合、最初の MAP 文のみが使用され、その他は無視されます。

デフォルト NOALLOWDUPTARGETMAP

構文 ALLOWDUPTARGETMAP | NOALLOWDUPTARGETMAP

ALLOWNONVALIDATEDKEYS

適用対象 GLOBALS

ALLOWNONVALIDATEDKEYS では、Extract、Replicat および GGSCI コマンドが、検証されていない主キーまたは Oracle GoldenGate による使用が無効なキーを、一意の識別子として許可できるようにします。このパラメータは、Oracle GoldenGate によって使用されるキー選択基準より優先されます。これが有効化されると、Oracle GoldenGate は、NON VALIDATED および NOT VALID 主キーを一意の識別子として使用します。

オブジェクトの再編成や他の多くのアクションの結果、キーが無効になることがありますが、キーが有効であることが判明している場合、ALLOWNONVALIDATEDKEYS は、特にテスト環境で、キーを再検証する停止時間を保存します。ただし、ALLOWNONVALIDATEDKEYS を使用する場合、テスト環境か本番環境であるかにかかわらず、ターゲット・データがレプリケーションを通じて正確に保持されない可能性があるというリスクを受け入れます。つまり、キーが無効であると判明し、定義される表に同じキー値を持つ複数のレコードが含まれる場合、Oracle GoldenGate は、更新するターゲット行を誤って選択する可能性があります。

DML サポートのみがアクティブである場合に ALLOWNONVALIDATEDKEYS 機能を有効化する方法

DML レプリケーションのみが使用中である場合に ALLOWNONVALIDATEDKEYS を有効化するには、すべてのプロセスを停止してから、ALLOWNONVALIDATEDKEYS を GLOBALS パラメータ・ファイルに追加して、プロセスを再起動します。無効化するには、GLOBALS ファイルから削除して、プロセスを再起動します。

DDL サポートがアクティブである場合に ALLOWNONVALIDATEDKEYS 機能を有効化する方法

1. Oracle GoldenGate インストール・ディレクトリから ddl_setup.sql スクリプトを実行することによって、DDL トリガーを再構築します。このスクリプトを使用するには、Oracle GoldenGate DDL オブジェクトがインストールされているスキーマを把握している必要があります。

2. ALLOWNONVALIDATEDKEYS パラメータを GLOBALS パラメータ・ファイルに追加します。
3. 次の SQL を使用して、DDL スキーマの GGS_SETUP 表を更新します。

```
UPDATE <owner>.GGS_SETUP SET value='1' WHERE
property='ALLOWNONVALIDATEDKEYS';
COMMIT;
```

4. Manager を含むすべての GoldenGate プロセスを再起動します。この時点から、Oracle GoldenGate は、非検証または無効の主キーを一意の識別子として選択します。

DDL サポートがアクティブである場合に ALLOWNONVALIDATEDKEYS 機能を無効化する方法

1. ALLOWNONVALIDATEDKEYS を GLOBALS パラメータ・ファイルから削除します。
2. GGS_SETUP 表に追加したレコードを 0 に更新します。
3. すべての Oracle GoldenGate プロセスを再起動します。

デフォルト なし (無効)

構文 ALLOWNONVALIDATEDKEYS

ALLOWNOOPUPDATES | NOALLOWNOOPUPDATES

適用対象 Replicat

ALLOWNOOPUPDATES および NOALLOWNOOPUPDATES では、Replicat が "no-op" 操作に対応する方法を制御します。no-op 操作は、ターゲット表では無効な操作です。次に、これを指定する場合の例を示します。

- ソース表に、ターゲット表に存在しない列がある、または (COLSEXCEPT 句を使用して) レプリケーションから除外された列がある。どちらの場合も、そのソース列を更新する場合、Replicat SQL 文内の SET 句で使用するターゲット列名がありません。
- 列を現在の値と同じ値に設定する更新が行われる。データベースは実際には変更されなかったため、新しい値は記録されません。しかし、主キーが記録されたため、Oracle GoldenGate はこの操作を変更レコードとして抽出しますが、Replicat SQL 文の SET 句に列値がありません。

デフォルト (NOALLOWNOOPUPDATES) では、これらのタイプの操作はデータベースを更新しないため、Replicat はエラーとともに異常終了します。ALLOWNOOPUPDATES を指定すると、Replicat は異常終了せずに操作を無視します。Replicat によってレポートされる統計には、更新は行われたが、データベースは更新されないと示されます。

内部パラメータ APPLYNOOPUPDATES を使用して、強制的に更新を適用させることができます。APPLYNOOPUPDATES は、ALLOWNOOPUPDATES より優先されます。両方が指定された場合、キー列のみを持つ更新が適用されます。デフォルトでは、ソース・キー列のみがあり、ターゲット表に定義されるキーがない場合、Oracle GoldenGate は次のメッセージとともに異常終了します。

```
2011-01-25 02:28:42 GGS ERROR   160 Encountered an update for target table TELLER,
which has no unique key defined. KEYCOLS can be used to define a key. Use
ALLOWNOOPUPDATES to process the update without applying it to the target database.
Use APPLYNOOPUPDATES to force the update to be applied using all columns in both
the SET and WHERE clause.
```

エラー処理が設定されている場合の例外

HANDLECOLLISIONS または INSERTMISSINGUPDATES パラメータが使用されているときに ALLOWNOOPUPDATES が指定された場合、および Oracle GoldenGate にすべてのターゲット・キー値がある場合、Oracle GoldenGate は更新を無視しませんが、かわりに、SET 句および WHERE 句のすべてのキー値を使用して (APPLYNOOPUPDATES 動作を起動して)、更新を適用します。更新されている行が見つからない場合に Oracle GoldenGate が検出できるように、これが必要になります。その場合、Oracle GoldenGate は更新を挿入に変更します。

デフォルト NOALLOWNOOPUPDATES (表にキーが含まれていない場合にのみ適用)

構文 `ALLOWNOOPUPDATES | NOALLOWNOOPUPDATES`

APPLYNOOPUPDATES | NOAPPLYNOOPUPDATES

適用対象 Replicat

APPLYNOOPUPDATES では、SET 句および WHERE 句の両方ですべての列を使用して、"no-op" 更新を適用させます。"no-op" の説明は、「ALLOWNOOPUPDATES | NOALLOWNOOPUPDATES」を参照してください。

トレイル内のどのデータでも、APPLYNOOPUPDATES によって使用されます。主キー更新レコードがある場合、ソースからビフォア列が使用されます。標準 (非キー) 更新がある場合、アフター値がビフォア値と同じであるとみなされます (それ以外の場合、主キー更新になります)。上記の場合、ソース・キーとターゲット・キーは同一であるとみなされます。同一ではない場合、ソースでの TABLE 文の KEYCOLS 句を使用する必要があります。

デフォルト `NOAPPLYNOOPUPDATES`

構文 `APPLYNOPUPDATES | NOAPPLYNOPUPDATES`

ASCIITOEBCDIC

適用対象 Extract データ・ポンプおよび Replicat

ASCIITOEBCDIC パラメータでは、ASCII から EBCDIC フォーマットへの入力トレイル・ファイル内のデータの変換を制御します。このパラメータは、入力トレイル・ファイルがリリース v10.0 より前の Extract によって作成された場合に、下位互換性をサポートするためにのみ使用する必要があります。ASCII から EBCDIC への変換が現在はデフォルトになっているため、それ以外の場合は、これは無視されます。

バージョン 11.2.1 より、データ・ポンプによる変換はできません。

デフォルト なし

構文 `ASCIITOEBCDIC`

ASSUMETARGETDEFS

適用対象 Replicat

ASSUMETARGETDEFS パラメータは、ホット・サイトを同期するときなど、MAP 文で指定したソースおよびターゲット表が同一の列構造を持つ場合に使用します。Oracle GoldenGate にソース定義ファイルからソース構造を参照しないよう指示します。

ソース表およびターゲット表が同一の構造を持たない場合は、ASSUMETARGETDEFS のかわりに SOURCEDEFS パラメータを使用します。331 ページの「SOURCEDEFS」を参照してください。

Oracle GoldenGate が、ソース表とターゲット表を同一または異なると判断する基準については、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』の「レプリケートされたデータとメタデータとの関連付け」を参照してください。

デフォルト なし

構文

```
ASSUMETARGETDEFS
```

AUTORESTART

適用対象 Manager

AUTORESTART パラメータでは、処理に失敗した 1 つ以上の Extract および Replicat プロセスを自動的に起動します。AUTORESTART は、一時的なネットワーク停止や、トランザクション・ログへのアクセスを妨害するプログラムなどによってプロセスが一時的に妨害されるときに、フォルト・トレランスを提供します。

AUTORESTART 文は、同一のパラメータ・ファイル内で複数回使用できます。

パッシブ・モードで作成された Extract グループにこのパラメータを適用するには、関連付けられている別名 Extract グループが存在するターゲット・システム上の Manager に対してこのパラメータを使用します。Oracle GoldenGate は、ソース・システムに起動コマンドを送信します。パッシブ Extract グループに対してローカルで使用すると、AUTORESTART は無視されます。

Manager は、再起動時に順序が正しくないトランザクションを検出すると、Extract を再起動しません。かわりに警告を出し、SEND EXTRACT の ETROLLOVER オプションを使用してトレイルを次のファイルに切り替え、エラーの原因のトランザクションをスキップするよう通知します。

デフォルト 自動的に再起動しない

構文

```
AUTORESTART <process type> <group name>
[, RETRIES <max retries>]
[, WAITMINUTES <wait minutes>]
[, RESETMINUTES <reset minutes>]
```

引数	説明

`<process type>`	次のいずれかを指定します。
	`EXTRACT`
	`REPLICAT`
	`ER`(Extract および Replicat)

引数	説明
<group name>	1 つのグループ名、または複数のグループを指定するワイルドカード。ワイルドカードを使用する場合、Oracle GoldenGate は、**パッシブ**・モードのものを除き、ワイルドカードの条件を満たす、指定された <process type> のローカル・システム上のすべてのグループを起動します。
RETRIES <max retries>	再試行を中止するまで Manager がプロセスの再起動を試行する最大回数。デフォルトの試行回数は 2 です。
WAITMINUTES <wait minutes>	異常終了したプロセスの検出からプロセスの再起動までの待機時間 (分)。このオプションは、必要なリソースが利用可能になるまで、または他のイベントが発生するまで再起動を遅延させるために使用します。デフォルトの遅延は 2 分です。
RESETMINUTES <reset minutes>	再試行回数をカウントする時間枠。デフォルトは 120 分 (2 時間) です。この時間が過ぎると、再試行回数は 0 に戻されます。

例 次の例では、Manager は、1 時間の時間枠内に処理に失敗したすべての Extract プロセスの再起動を 3 回試行します。各再試行の前に 5 分待機します。

```
AUTORESTART EXTRACT *, RETRIES 3, WAITMINUTES 5, &
RESETMINUTES 60
```

AUTOSTART

適用対象 Manager

AUTOSTART パラメータでは、Manager が起動するときに、1 つ以上の Extract および Replicat プロセスを自動的に起動します。AUTOSTART によって、すべてのプロセス・グループを見逃すことなく、即座に同期アクティビティを開始できます。

AUTOSTART 文は、同一のパラメータ・ファイル内で複数回使用できます。

パッシブ・モードで作成された Extract グループにこのパラメータを適用するには、関連付けられている別名 Extract グループが存在するターゲット・システム上の Manager に対してこのパラメータを使用します。Oracle GoldenGate は、ソース・システムに起動コマンドを送信します。パッシブ Extract グループに対してローカルで使用すると、AUTOSTART は無視されます。

Manager は、再起動時に順序が正しくないトランザクションを検出すると、Extract を再起動しません。かわりに警告を出し、SEND EXTRACT の ETROLLOVER オプションを使用してトレイルを次のファイルに切り替え、エラーの原因のトランザクションをスキップするよう通知します。

デフォルト 自動的に起動しない

構文

```
AUTOSTART <process type> <group name>
```

引数	説明
<process type>	次のいずれかを指定します。 EXTRACT REPLICAT ER(Extract および Replicat)
<group name>	1 つのグループ名、または複数のグループを指定するワイルドカード。ワイルドカードを使用する場合、Oracle GoldenGate は、**パッシブ**・モードのものを除き、ワイルドカードの条件を満たす、指定された <process type> のローカル・システム上のすべてのグループを起動します。

例 AUTOSTART ER *

BATCHSQL

適用対象 Replicat

BATCHSQL パラメータでは、Replicat のパフォーマンスを向上させます。BATCHSQL を使用すると、Replicat は同様の SQL 文を配列にまとめてより高速に適用します。標準モードでは、Replicat は 1 回に 1 つの SQL 文を適用します。

BATCHSQL は次のデータベースに有効です。

- DB2 LUW
- z/OS 上の DB2
- Oracle
- NonStop SQL/MX
- PostgreSQL
- SQL Server
- Teradata

BATCHSQL の動作

BATCHSQL モードでは、Replicat は同様の SQL 文をメモリー・キュー内でバッチにまとめ、1 回のデータベース操作で各バッチを適用します。1 つのバッチに含まれる SQL 文は、同一の表、操作タイプ (挿入、更新または削除) および列リストを処理します。たとえば、次の各グループは 1 つのバッチです。

- 表 A への挿入
- 表 B への挿入
- 表 A への更新
- 表 B への更新
- 表 A からの削除
- 表 B からの削除

注意 Oracle GoldenGate は、バッチの実行の前に、バッチの外部キー参照依存関係を分析します。異なるバッチ内の文の間に依存関係がある場合、各バッチの複数の SQL 文で参照整合性の維持が必要になることがあります。

キャッシュされる文の数の制御

MAXSQLSTATEMENTS パラメータは、キャッシュされる文の数を制御します (283 ページを参照してください)。古い SQL 文は、最低使用頻度アルゴリズムを使用して再利用されます。バッチは、指定されているしきい値 (「メモリーの管理」を参照してください) に基づいて実行されます。

使用制限

例外として処理される SQL 文には、次のものが含まれます。

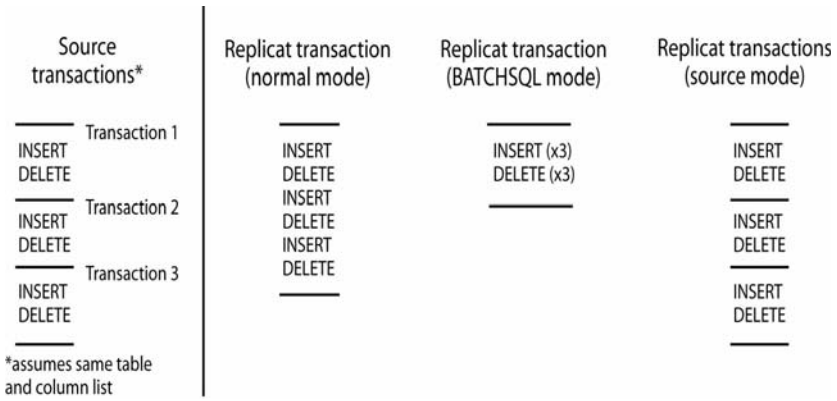
- LOB または LONG データを含む文。
- 25K よりも長い行を含む文。
- ターゲット表に、主キー以外に 1 つ以上の一意キーが含まれている文。BATCHSQL では、主キー以外のキーの値が変更されたときにそれらのキーの正しい順序付けが保証されないため、バッチではこのような文を処理できません。
- (SQL Server) ターゲット表にトリガーが含まれている文。
- エラーを発生させる文。

Replicat は、バッチ・モードで例外が発生すると、バッチ操作をロールバックし、次の方法で例外の処理を試行し、常にトランザクションの整合性を維持します。

- まず、Replicat は標準モードの使用を試み、GROUPTRANSOPS パラメータ (217 ページを参照してください) で設定されているトランザクション境界内で 1 回に 1 つの SQL 文を適用します。
- 標準モードで失敗すると、Replicat はソース・モードの使用を試み、ソースと同じトランザクション境界内で SQL を適用します。

例外処理が終了すると、Replicat は BATCHSQL モードを再開します。

図 14 Replicat のモード



BATCHSQL を使用するケースの決定

Replicat が BATCHSQL モードの場合、小さな行の変更は大きな行の変更よりも高いパフォーマンスの向上を示します。行変更当たりのデータ量が 100 バイトの場合、BATCHSQL を使用することによって Replicat のパフォーマンスが最大で 300% 向上することがわかっていますが、実際のパフォーマンス向上は、混在する操作に応じて異なります。行変更当たりのデータ量が約 5,000 バイトの場合、

BATCHSQL を使用するメリットは低下します。

メモリーの管理

SQL 文をバッチにまとめると、効率性は向上しますが、メモリーも消費します。最適なパフォーマンスを維持するために、次の BATCHSQL オプションを使用してください。

BATCHESPERQUEUE

BYTESPERQUEUE

OPSPERBATCH

OPSPERQUEUE

値を設定するベンチマークとして、それぞれ 500 バイトの 1,000 の SQL 文をまとめた 1 つのバッチが必要とするメモリーは 10MB 未満であると想定してください。

デフォルト 無効 (標準 Replicat モードで処理)

構文

```
BATCHSQL
[BATCHERRORMODE | NOBATCHERRORMODE]
[BATCHESPERQUEUE <n>]
[BATCHTRANSOPS <n>]
[BYTESPERQUEUE <n>]
[OPSPERBATCH <n>]
[OPSPERQUEUE <n>]
[TRACE]
```

オプション	説明
`BATCHERRORMODE | NOBATCHERRORMODE`	BATCHSQL 処理モード中に発生するエラーに対する Replicat のレスポンスを設定します。 ◆ BATCHERRORMODE では、BATCHSQL モードのまま Replicat にエラーの解決を試行させます。重複レコード・エラーが原因で失敗した挿入を更新に変換し、削除に対する行方不明レコード・エラーを無視します。BATCHERRORMODE を使用する場合は、HANDLECOLLISIONS パラメータを使用して Replicat の異常終了を防止します。 ◆ NOBATCHERRORMODE (デフォルト) では、エラーが発生すると、Replicat は一時的に BATCHSQL 処理を無効にし、まず標準モードでトランザクションを再試行し、標準モードで失敗すると、ソース・モード (ソースと同じトランザクション境界) で実行します。
`BATCHESPERQUEUE <n>`	1 つのメモリー・キューに維持できる最大バッチ数を制御します。BATCHESPERQUEUE の値に到達すると、ターゲット・トランザクションが実行されます。 ◆ 最小値は 1 です。 ◆ 最大値は 1000 です。 ◆ デフォルトは 50 です。

オプション	説明
BATCHTRANSOPS <n>	コミットが必要になる前に 1 つのトランザクションにグループ化可能な最大バッチ操作数を制御します。BATCHTRANSOPS の値に到達すると、操作がターゲットに適用されます。BATCHTRANSOPS は、デフォルトの 1000 以上に設定します。 ◆ 最小値は 1 です。 ◆ 最大値は 100000 です。 ◆ デフォルトは 1000 です。
BYTESPERQUEUE <n>	1 つのキューに維持できる最大バイト数を設定します。BYTESPERQUEUE の値に到達すると、ターゲット・トランザクションが実行されます。 ◆ 最小値は 1000000 バイト (1MB) です。 ◆ 最大値は 1000000000 バイト (1GB) です。 ◆ デフォルトは 2000000 バイト (20MB) です。
OPSPERBATCH <n>	1 つのバッチに含められる最大行操作数を設定します。OPSPERBATCH の値に到達すると、ターゲット・トランザクションが実行されます。 ◆ 最小値は 1 です。 ◆ 最大値は 100000 です。 ◆ デフォルトは 1200 です。
OPSPERQUEUE <n>	1 つのキューに維持できるすべてのバッチの最大行操作数を設定します。OPSPERQUEUE の値に到達すると、ターゲット・トランザクションが実行されます。 ◆ 最小値は 1 です。 ◆ 最大値は 100000 です。 ◆ デフォルトは 1200 です。
TRACE	コンソールおよびレポート・ファイルへの BATCHSQL アクティビティの詳細なトレースを有効化します。トレースを設定するときは、必ず Oracle サポート・アナリストの指示を受けるようにしてください。

例 `BATCHSQL BATCHESPERQUEUE 100, OPSPERBATCH 2000`

BEGIN

適用対象 Replicat

BEGIN パラメータでは、BEGIN で指定した時刻と同じまたはそれより大きいタイムスタンプを持つ、データベース・トランザクション・ログ内または Oracle GoldenGate トレイル内の最初のレコードから処理を開始します。それ以降のレコードは、タイムスタンプが指定した時刻よりも小さいものを含めすべて処理されます。BEGIN は、SPECIALRUN が同じ Replicat グループに指定されているときに使用します。

デフォルト なし

構文 `BEGIN <date> [<time>]`

引数	説明
<date> [<time>]	いつ処理を開始するかを指定します。有効な値 : ◆ yyyy-mm-dd:hh:mi[:ss[.cccccc]] (24時間表記) フォーマットの日付とオプションの時刻。秒およびセンチ秒はオプションです。

例 BEGIN 2011-01-01:04:30:00

BINARYCHARS | NOBINARYCHARS

適用対象 Extract および Replicat

BINARYCHARS および NOBINARYCHARS では、文字データをバイナリ・データまたは NULL 終了文字列として処理するかどうかを制御します。

デフォルトの BINARYCHARS は、ソース表に入力された方法でデータを保持します。これにより、ソース・データベースまたはターゲット・データベース内の列が文字用の列として定義され、バイナリ文字をその列に入力できる場合に、適切な処理が保証されます。BINARYCHARS は BULKLOAD パラメータ (ダイレクト・バルク・ロード) と互換性がありません。NOBINARYCHARS を使用します。

NOBINARYCHARS によって、Oracle GoldenGate では、バイナリ文字がその列のデータの末尾であると解釈されることがあります。バイナリ・データの後にさらにデータがある場合、Oracle GoldenGate によって処理されず、データ整合性が損なわれます。FORMATASCII の DELIMITER オプションで定義された文字に加え、NULL 文字によってもこれが発生します。NOBINARYCHARS を使用する正当な理由がない場合、データがソース表に入力された方法で保持されるように、デフォルト設定を BINARYCHARS のままにすることをお薦めします。NOBINARYCHARS を使用する前に、Oracle サポートに連絡してください。

BINARYCHARS および NOBINARYCHARS は表固有のものです。一方のパラメータは、もう一方が見つかるまで、それ以降のすべての TABLE または MAP 文に有効です。

デフォルト BINARYCHARS

構文 BINARYCHARS | NOBINARYCHARS

BLOBMEMORY

このパラメータは、LOBMEMORY の別名です (227 ページを参照してください)。

BOOTDELAYMINUTES

適用対象 Manager

Windows システム向けの BOOTDELAYMINUTES パラメータでは、Manager が起動時に実行するアクティビティ (パラメータの実行など) を遅延させます。たとえば、BOOTDELAYMINUTES を使用して、データベース・サービスが開始されるまで AUTOSTART パラメータの実行を遅延させることができます。

BOOTDELAYMINUTES は、他のパラメータ・エントリより先に指定します。このパラメータは、Windows のみサポートしています。

デフォルト なし (遅延なし)

構文 BOOTDELAYMINUTES <minutes>

引数	説明
<minutes>	システムの起動後、Oracle GoldenGate の処理開始までの遅延時間 (分)。

例 BOOTDELAYMINUTES 5

BR

適用対象 Extract(Oracle のみ)

BR パラメータでは、Bounded Recovery (BR) 機能を制御します。この機能は、現在 Oracle データベースをサポートしています。

Bounded Recovery は、一般的な Extract チェックポイント機能のコンポーネントです。計画的または計画外の何らかの理由で Extract が停止した場合、その時点でオープンしている (コミットされていない) トランザクションの数、およびトランザクションの古さにかかわらず、Extract の停止後の効率的なリカバリを保証します。Bounded Recovery により、Extract が停止した位置までリカバリし、正常な処理を再開するまでにかかる最大時間の上限を設定できます。

警告 **このパラメータをデフォルト設定から変更する前に、Oracle サポートに連絡してください。ほとんどの本番環境では、このパラメータの変更は必要ありません。ただし、Bounded Recovery チェックポイント・ファイル用のディレクトリは、Oracle サポートの支援なしで指定できます。**

Extract によるオープンしていたトランザクションのリカバリ方法

Extract は、REDO ログ内でトランザクションの開始位置 (Oracle の場合は最初の実行可能 SQL 文) を検出すると、このトランザクション内の取得が指定されているすべてのデータをメモリーにキャッシングし始めます。取得が必要なデータを含まないトランザクションにも、今後の操作によって取得が必要なデータが含まれる可能性があるため、Extract はこのようなトランザクションもキャッシュする必要があります。

Extract は、トランザクションのコミット・データを検出すると、キャッシュしているこのトランザクション全体をトレイルに書込み、メモリーからクリアします。Extract は、トランザクションのロールバック・レコードを検出すると、このトランザクション全体をメモリーから破棄します。Extract がコミットまたはロールバックを処理するまで、トランザクションは*オープンしている*とみなされ、その情報収集が継続されます。

トランザクションのコミットまたはロールバック・レコードを検出する前に Extract が停止した場合には、Extract の再起動の際に、キャッシュされていた情報をすべてリカバリする必要があります。Extract が停止したときにオープンしていたすべてのトランザクションがリカバリの対象になります。

Extract は、次のようにこのリカバリを実行します。

- Extract が停止したときにオープンしているトランザクションがなかった場合は、現在の Extract 読取りチェックポイントからリカバリを開始します。これは標準リカバリです。

- ログ内での開始位置の時刻が Extract の停止時刻と非常に近いオープンしていたトランザクションがある場合、Extract は*その中で最も古いトランザクションの開始位置*からログを再度読み取ってリカバリを開始します。この場合 Extract は、停止前にトレイルに書き込み済または破棄したトランザクションについては重複作業を行う必要がありますが、この作業は処理するデータ量が比較的少ないため許容できます。これも標準リカバリとみなされます。
- Extract が*長時間オープンしているトランザクション*とみなすトランザクションが 1 つ以上あった場合は、Extract は *Bounded Recovery* を利用してリカバリを開始します。

Bounded Recovery の動作

トランザクションは、BR パラメータの BRINTERVAL オプションで指定されている 1 つの *Bounded Recovery 間隔*より長くオープンしている場合に長時間におよぶトランザクションとみなされます。たとえば、Bounded Recovery 間隔が 4 時間の場合は、4 時間よりも前に開始されたすべてのトランザクションが長時間オープンしているトランザクションになります。

Extract は、Bounded Recovery 間隔ごとに *Bounded Recovery チェックポイント*を作成し、(存在する場合) 長時間におよぶトランザクションの状態とデータを含め、Extract の現在の状態とデータをディスクに永続化します。Extract は、Bounded Recovery チェックポイント作成後に停止した場合、オープンしている長時間におよぶ最も古いトランザクションが最初に登場するログ内の位置からではなく、直前の Bounded Recovery 間隔内の位置、または最新の Bounded Recovery チェックポイントからリカバリを開始します。

最大 Bounded Recovery 時間 (Extract が停止位置までのリカバリにかかる最大時間) は、現在の Bounded Recovery チェックポイント間隔の 2 倍を上回ることはありません。実際のリカバリ時間は、次の要因によって決まります。

- 最新の有効な Bounded Recovery 間隔から Extract 停止までの時間。
- その期間の Extract 使用状況。
- 以前にトレイルに書き込まれたトランザクションの使用率。Bounded Recovery は、これらのトランザクションをディスクに書き込まなければならなかった最初の Extract 処理よりも、(破棄することによって) はるかに早くこれらのトランザクションを処理します。トランザクション・データに対して発生する再処理のほとんどは破棄です。

Extract は、リカバリの際、最新の Bounded Recovery チェックポイントで保存された永続データおよび状態 (長時間におよぶトランザクションのものを含む) をリストアします。

たとえば、あるトランザクションが 24 時間オープンしていて、Bounded Recovery 間隔が 4 時間であるとします。このときの最大リカバリ時間は、Extract 処理時間の 8 時間を超えず、たいていこの時間を下回ります。最新の有効な Bounded Recovery チェックポイントに対する停止位置、およびその時点における Extract のアクティビティによって、実際の時間は異なります。

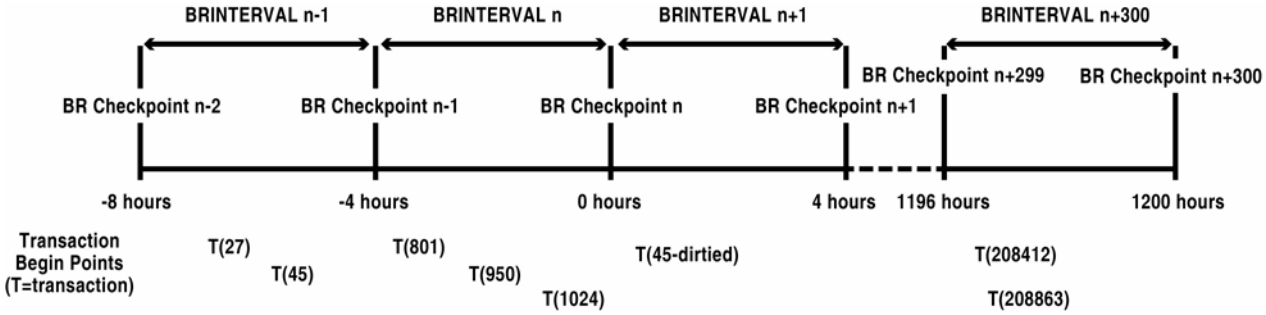
Bounded Recovery によって解決される問題

ディスクの永続性を使用して長時間におよぶトランザクションを保管し、リカバリすることによって、Extract は発生頻度は少ないながら、発生した場合にパフォーマンスに重大な (悪い) 影響を及ぼす状況に対応できます。多くの場合、長時間におよぶトランザクションの開始点は、Extract が停止したときのログ内の処理位置よりもはるか前に遡ります。長時間におよぶトランザクションは、多数の古いログにまたがっている可能性があり、古いログの一部はアクセス可能な記憶域に存在しないか、削除されていることもあります。長時間におよぶトランザクションの開始からもう一度ログを読み取るためには、許容できないほど長時間要するだけではなく、長時間におよぶトランザクションはまれにしか発生しないため、大半はすでにトレイルに書き込み済、または破棄された他のトランザクションをもう一度取得する作業が必要になります。永続化された長時間におよぶトランザクションの状態とデータをリストアできれば、この作業は不要になります。

Bounded Recovery の例

次の図は、一連のトランザクションが開始されたタイムラインを表しています。これは、長時間オープンしているトランザクションが特定の間隔の後にディスクにどのように永続化され、処理失敗の後にリカバリされるかを示しています。これは、この例で使用されている用語の理解に役立ちます。

- *永続オブジェクトとは*、Bounded Recovery チェックポイントで永続化されたキャッシュ内の任意のオブジェクトです。通常はトランザクションの状態またはデータですが、キャッシュは Extract の内部のオブジェクトにも使用されます。これらすべてを集合的にオブジェクトと呼びます。
- *最も古い非永続オブジェクト*とは、現在の Bounded Recovery チェックポイント直前の間隔内のキャッシュにある、オープンしている最も古いオブジェクトです。通常は、この間隔内にあるオープンしている最も古いトランザクションです。Bounded Recovery を再起動すると、実行時プロセスは最も古い非永続オブジェクトの位置から再開し、通常のトランザクションの場合は、これが REDO ログ内のこのトランザクションの位置になります。



この例では、Bounded Recovery 間隔は 4 時間です。オープンしているトランザクションは、現在の Bounded Recovery チェックポイントから 1 つの Bounded Recovery 間隔より長い間オープンしている場合、現在のチェックポイントで永続化されます。

BR チェックポイント n:

- オープンしているトランザクションは、T(27)、T(45)、T(801)、T(950) および T(1024) の 5 つです。これ以外のトランザクションはすべて、コミットされてトレイルに送信されたか、ロールバックされました。トランザクションは、タイムラインに沿って開始位置が示されています。
- 1つのBounded Recovery間隔より長くオープンしているトランザクションは、T(27)およびT(45)です。**BR チェックポイント n** でこれらはディスクに永続化されます。
- 最も古い非永続オブジェクトは T(801) です。これは、オープンしている時間が 1 つの Bounded Recovery 間隔に満たないので、ディスクに永続化される資格がありません。このトランザクションは最も古い非永続化オブジェクトなので、ログ内のこのトランザクションの開始位置が **BR チェックポイント n** チェックポイント・ファイルに保管されます。Extract は **BR チェックポイント n** 以降に予期せず停止すると、このログ位置までリカバリし、そこからログの再読取りを開始します。直前の Bounded Recovery 間隔に最も古い非永続オブジェクトが存在しない場合、Extract は現在の Bounded Recovery チェックポイントのログ位置からログの再読取りを開始します。

BR チェックポイント n+1:

- T(45) は直前の Bounded Recovery 間隔で使用済 (更新済) なので、新しい永続オブジェクト・ファイルに書き込まれます。古いファイルは、**BR チェックポイント n+1** 完了後に削除されます。
- Extract は、**BR チェックポイント n+1** の書込み中、または **BR チェックポイント n** と **BR チェックポイント n+1** 間の Bounded Recovery チェックポイント間隔内のある時点で処理に失敗すると、最後の有効なチェックポイントである **BR チェックポイント n** からリカバリを開始します。**BR チェックポイント n** の再開始位置は、最も古い非永続トランザクションである T(801) の開始点です。したがって、最も長いリカバリ時間は常に、Extract が停止したポイントから 2 つの Bounded Recovery 間隔未満、このケースでは 8 時間未満になります。

BR チェックポイント n+3000

- このシステムは長い時間実行されています。T(27) および T(45) のみが永続トランザクションです。T(801) および T(950) は、**BR チェックポイント n+2999** より前のある時点でコミットされ、トレイルに書き込まれました。現在オープンしているトランザクションは、T(208412) および T(208863) のみです。
- **BR チェックポイント n+3000** が書き込まれます。
- **BR チェックポイント n+3000** の後の間隔で電源障害が発生します。
- 新しい Extract が **BR チェックポイント n+3000** までリカバリします。T(27) および T(45) は、**BR チェックポイント n** 時点の状態を含む、それぞれの永続ファイルからリストアされます。ログ読取りは、T(208412) の開始から再開します。

長時間におよぶトランザクションの管理

Oracle GoldenGate は、長時間におよぶトランザクションを管理するために、次のパラメータとコマンドを提供します。

- WARNLONGTRANS パラメータでは、Extract によってトランザクションが長時間におよんでいることの警告メッセージが生成されるまでの、トランザクションのオープン時間を指定します。また WARNLONGTRANS も使用して、Oracle GoldenGate が長時間におよぶトランザクションをチェックする頻度を制御します。この設定は Bounded Recovery 間隔とは無関係で、Bounded Recovery 間隔に影響しません。
- SEND EXTRACT コマンドと SKIPTRANS オプションでは、指定したトランザクションを Extract にスキップさせます。
- SEND EXTRACT コマンドと FORCETRANS オプションでは、Extract に、指定したトランザクションをコミット済トランザクションとしてトレイルに書き込ませます。
- TRANLOGOPTIONS パラメータと PURGEORPHANEDTRANSACTIONS オプションでは、ノードに障害が発生し、Extract がロールバックを取得できないときに発生する孤立トランザクションのパージを有効化します。

ディスクに書き込まれるファイルについて

Bounded Recovery 間隔が終了すると、Extract は常に Bounded Recovery チェックポイント・ファイルを作成します。永続化が必要な長時間におよぶトランザクションが存在する場合、これらのトランザクションは各自の永続オブジェクト・ファイルに書き込まれます。永続オブジェクト・ファイルには、ディスクに永続化される単一のトランザクションの状態およびデータが格納されます。

これまでの経験では、長時間におよぶトランザクションを永続化する必要はほとんどなく、このようなケースのほとんどでトランザクションは空です。

以前に永続化されたオブジェクトがまだオープンしていて、完了したばかりの Bounded Recovery 間隔中にその状態やデータが変更された場合、このオブジェクトは新しい永続オブジェクト・ファイルに再度永続化されます。このような変更が発生しない場合、オープンしているトランザクションの以前に作成された永続化オブジェクト・ファイルは変更されません。

理論的には、1 つの長時間におよぶトランザクションを永続化するために、複数の永続ファイルが必要になること可能性もあります。

注意 **Bounded Recovery ファイルは、Extract が別のシステムに移動された場合、同一のデータベースが使用されているとしても、新しいシステムのすべての関連要素が元のシステムと同一でなければ、Extract の状態のリカバリに使用できません。たとえば、Oracle 11g Solaris プラットフォームに書き込まれたチェックポイント・ファイルは、Oracle 11g Linux プラットフォーム上の Extract のリカバリに使用できません。**

Bounded Recovery を標準リカバリに変更する状況

Extract は、永続オブジェクトが存在するまれな状況を除き、ほとんどのケースで Bounded Recovery ではなく標準リカバリを使用します。特定の異常な状況では、Extract は Bounded Recovery から標準リカバリ・モードに切り替えられない場合があります。たとえば、(長時間におよぶトランザクションの永続データが保管されている) ディスクの物理的な破損、Bounded Recovery チェックポイント・ファイルの不注意な削除、環境の継続性に影響を及ぼすその他のアクションやイベントなどが発生する状況があげられます。もっと容易に修正可能な障害が原因の場合もあります。

一部のケースを除き、Extract は、リカバリ中に Bounded Recovery 処理に失敗すると、標準モードに切り替えます。標準リカバリの完了後、Bounded Recovery はもう一度有効化されます。

Bounded Recovery は、次の状況では起動されません。

- Extract の開始位置が CSN または時間によって変更された。
- Extract I/O チェックポイントが変更された。
- Extract パラメータ・ファイルがリカバリ中に変更された (TABLE 指定の変更など)。

リカバリの完了後、Bounded Recovery は次の実行に向けてもう一度有効化されます。

Bounded Recovery 中に Extract が異常終了した場合の措置

Extract が Bounded Recover 中に異常終了した場合は、エラー・ログを確認して理由を特定してください。無効なパラメータ・ファイル、Bounded Recovery ファイルが含まれるディレクトリの不適切な権限など、迅速に解決できる問題が原因の場合もあります。こうしたケースでは、問題の修正後、Extract を再起動して Bounded Recovery を有効化できます。

問題が修正できない場合は、Extract を GGSCI の BRRESET コマンドで再起動してください。Extract は標準モードでリカバリを実行し、その後 Bounded Recovery を再び有効化します。

BR パラメータの変更

Bounded Recover は、デフォルトで有効化されており、デフォルトの Bounded Recovery 間隔は 4 時間です (BRINTERVAL オプションで制御します)。この間隔は、ほとんどの環境に適切なはずです。BR パラメータを変更するときは、先に必ず Oracle サポート・アナリストの指示を受けるようにしてください。Oracle GoldenGate アナリストは、使用可能な Bounded Recovery 実行時統計に基づいて Bounded Recovery 使用状況を分析し、デフォルトでは不十分な不測のイベント発生時に適切な BRINTERVAL の設定を特定できます。

BR の変更を要求された場合は、Bounded Recovery 間隔が標準 Extract チェックポイント間隔の倍数であることに注意してください。Extract チェックポイントは、CHECKPOINTSECS パラメータで制御されます。したがって、BR パラメータは、標準 Extract チェックポイントに対する Bounded Recovery チェックポイントの割合を制御します。Oracle 担当者から指示された場合は、両方のパラメータを変更する必要があります。

サポートされているデータベース

このパラメータは Oracle データベースに適用されます。他のデータベースでは、Extract は障害時の最も古いトランザクションの開始位置から古いログを読取ってリカバリを行い、長時間におよぶトランザクションは永続化されません。

デフォルト BR BRINTERVAL 4, BRDIR BR

構文

```
BR
[, BRDIR <directory>]
[, BRINTERVAL <interval><unit>]
[, BRKEEPSTALEFILES]
[, BROFF]
[, BROFFONFAILURE]
[, BRRESET]
```

引数	説明
BRDIR <directory>	BR ディレクトリを含める親ディレクトリの相対パス名または完全パス名を指定します。BR ディレクトリには、Bounded Recovery チェックポイント・ファイルが保管され、このディレクトリ名は変更できません。BR ディレクトリのデフォルトの親ディレクトリは、Oracle GoldenGate インストール・ファイルを含むルート・ディレクトリにある BR という名前のディレクトリです。 特定の Oracle GoldenGate インストール内の各 Extract グループは、BRDIR で指定されたディレクトリの下に各自のサブディレクトリを持ちます。各サブディレクトリには、関連する Extract グループの名前が付けられます。 <directory> には、文字列 "temp" または "tmp"(大 / 小文字を区別しない) を含む名前を使用しないでください。一時ディレクトリは、内部または外部のクリーンアップ・プロシージャで削除されることがあります。

引数	説明
BRINTERVAL <interval><unit>	Bounded Recovery チェックポイント間の時間を指定します。この時間を *Bounded Recovery 間隔*と呼びます。この間隔は、CHECKPOINTSECS パラメータで制御される標準 Extract チェックポイント間隔の整数の倍数です。ただし、正確な値を設定する必要はありません。Bounded Recovery は、必要に応じて内部で有効な BRINTERVAL パラメータを調整します。 <interval> の最小値は 20 分です。最大値は 96 時間です。 <unit> は次のように指定できます。 ◆ M (分の場合) ◆ H (時間の場合) デフォルト間隔は 4 時間です。
BRKEEPSTALEFILES	古い Bounded Recovery チェックポイント・ファイルを保持します。デフォルトでは、現在のチェックポイント・ファイルのみが保持されます。Extract は、古い Bounded Recovery チェックポイント・ファイルからはリカバリできません。古いファイルは、Oracle サポート・アナリストに指示された場合にのみ保持するようにしてください。
BROFF	Bounded Recovery の実行およびリカバリを無効にします。このオプションを使用する前に Oracle サポートに連絡してください。通常の場合、Bounded Recovery に問題があると、Bounded Recovery は自身を無効化します。
BROFFONFAILURE	エラー発生後、Bounded Recovery を無効化します。デフォルトでは、Extract は Bounded Recovery 処理中にエラーを検出すると、標準リカバリに切り替えますが、リカバリの完了後 Bounded Recovery をもう一度有効化します。BROFFONFAILURE では、Bounded Recovery の実行時処理を無効化します。
BRRESET	**注意 :** このオプションを使用するには、Extract をコマンドラインから起動する必要があります。Extract をコマンドラインから起動するには、次を実行します。 `replicat paramfile <name>.prm reportfile <name>.rpt` 条件 : paramfile <name>.prm は、Extract パラメータ・ファイルの相対名または完全修飾名です。このコマンド名は pf に短縮できます。 reportfile <name>.rpt は、Extract レポート・ファイルをデフォルト以外の場所に配置する場合の、Extract レポート・ファイルの相対名または完全修飾名です。このコマンド名は rf に短縮できます。 BRRESET では、Extract に、現在の実行の間は標準リカバリを使用させ、リカバリの完了後に Bounded Recovery を再び有効化させます。このパラメータは、エラーを検出しても Bounded Recovery が標準モードに戻らない、まれなケースで使用します。Bounded Recovery は、実行時に有効化されます。このオプションを使用する前に Oracle に連絡してください。

例 BR BRDIR /user/checkpt/br では、Bounded Recovery チェックポイント・ファイルの作成先を /user/checkpt/br ディレクトリに指定します。

BULKLOAD

適用対象 Replicat

初期ロード Replicat 用の BULKLOAD パラメータは、SQL*Loader へのダイレクト・バルク・ロード方法を使用するときに使用します。この方法では、ダイレクト・ロードを実行するために、初期ロード・データを Oracle の SQL*Loader ユーティリティのインタフェースに直接渡します。Collector プロセスとトレイルは使用されません。

BULKLOAD パラメータを使用するときは、Extract パラメータ・ファイルの NOBINARYCHARS パラメータを使用します。NOBINARYCHARS を使用する前に、Oracle サポートに連絡してください。詳細は、http://support.oracle.com を参照してください。

Oracle GoldenGate でのデータ・ロード方法の詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。

デフォルト なし

構文 BULKLOAD

CACHEMGR

適用対象 Extract(z/OS 上の DB2 および NonStop SQL/MX を除くすべてのデータベース)

CACHEMGR パラメータでは、コミットされていないトランザクション・データのキャッシュに使用可能な仮想メモリーおよび一時ディスク領域の量を制御します。

警告 **このパラメータをデフォルト・キャッシュ設定から変更する前に、Oracle サポートに連絡してください。Oracle GoldenGate のキャッシュ・マネージャは、内部で自動構成および自動調整を行うため、ほとんどの本番環境ではこのパラメータを変更する必要はありません ただし、ページ・ファイルのディレクトリは、Oracle サポートの支援なしで指定できます。**

メモリー管理について

注意 **ここにはできるかぎり正確に説明していますが、メモリー管理コンポーネントの基本設計は、製品の継続的な改善のために変更されることがあります。**

Oracle GoldenGate はコミットされたトランザクションのみをレプリケートするため、トランザクションのコミットまたはロールバックを受信するまで、キャッシュと呼ばれる管理仮想メモリー・プールに各トランザクションの操作を保持します。1 つのグローバル・キャッシュが 1 つの Extract プロセスの共有リソースとして機能します。このグローバル・キャッシュから、次の仮想メモリー・サブプールが割り当てられます。

- 大半のトランザクション行データ用としてログ・リーダー・スレッド当たり 1 つのサブプール
- BLOB データおよびその他の大きなアイテム用として 1 つのサブプール

各サブプール内に、グローバル・キャッシュから個別のバッファが割り当てられます。各バッファには、Oracle GoldenGate が処理中の 1 つのトランザクションに関連する情報が保持されます。

Oracle GoldenGate キャッシュ・マネージャは、Oracle GoldenGate が継続的かつ効率的に作業を処理できるように、オペレーティング・システムのメモリー管理機能を活用します。キャッシュ内では、最新の仮想メモリー技法を活用して次のことを行います。

- アクティブなバッファの割当ておよび管理を効率的に行う。
- 可能な場合はディスクにページングせずに古いバッファを再利用する。
- 必要な場合は使用頻度が少ない情報をディスクにページングする。

すべての Oracle GoldenGate プロセスが使用する実際の物理メモリー量は、Oracle GoldenGate プログラムではなく、オペレーティング・システムによって制御されます。

キャッシュ・マネージャは、グローバル・キャッシュ・サイズの弱い制限の範囲で Oracle GoldenGate プロセスを機能させ、オンデマンドで仮想メモリーのみを割り当てます。(キャッシュ・マネージャは、制御されない物理メモリーを割り当てません。) キャッシュ・サイズ増加のためのシステム・コールは、最後の手段としてのみ使用され、使用された場合は、その後必ず仮想メモリーが解放されてシステムに戻されます。

システムは、今後実行する各 Oracle GoldenGate Extract および Replicat プロセスのために、十分なスワップ領域を確保している必要があります。必要なスワップ領域を特定するには、次の手順を実行します。

1. 1 つの Extract プロセスおよび 1 つの Replicat プロセスを起動します。
2. GGSCI を実行します。
3. 実行中の各プロセスのレポート・ファイルを表示し、PROCESS VM AVAIL FROM OS (min) 行を探します。
4. 必要な場合、各値を次の整数 (GB) に切り上げます。たとえば、1.76GB の場合は 2GB に切り上げます。
5. 繰り上げられた Extract の値に、Extract プロセスの数を掛けます。
6. 繰り上げられた Replicat の値に、Replicat プロセスの数を掛けます。
7. 2 つの結果に、システム上の Oracle GoldenGate プロセスおよびその他のプロセスで必要とする追加スワップ領域を足します。

   ```
   ((<PROCESS VM> x <n_Extracts>) + (<PROCESS VM> x <n_Replicats>) +
   ((<swap for other processes>) = Max swap space on system
   ```

   この合計が、これらのプロセスに必要な最大スワップ領域になります。必要なプロセス数を特定する方法は、『*Oracle GoldenGate Windows and UNIX 管理者ガイド*』の構成の章を参照してください。

すべての Oracle GoldenGate プロセスが使用する実際の物理メモリー量は、Oracle GoldenGate プロセスではなく、オペレーティング・システムによって制御されます。グローバル・キャッシュ・サイズは、CACHEMGR の CACHESIZE オプションで制御します。

注意 **キャッシュ・マネージャは、Oracle GoldenGate によって、Extract の BLOB 用サブプールやその他のトランザクション・データ用のサブプールの制御以外の目的でも内部で使用されます。統計を表示するときに、このような追加メモリー・プールの情報が表示されることがあります。**

CACHEMGR を調整する時期

メモリー・マネージャが生成する統計は、SEND EXTRACT または SEND REPLICAT コマンドを CACHEMANAGER オプションとともに使用すると表示できます。この統計は、メモリー・プールのサイズ、ページングの頻度、トランザクションのサイズ、システム・プロファイルを作成するその他の情報を表示します。

このプロファイルを確認し、パフォーマンスの問題がファイル・キャッシングに関連していると思われるときは、メモリー・キャッシュの調整が必要になることがあります。その場合は、まず CACHESIZE および CACHEPAGEOUTSIZE パラメータを変更します。生成されているトランザクションのサイズとタイプに応じて、より大きなまたは小さなキャッシュ・サイズ、より大きなまたは小さなページ・サイズ、またはこれら 2 つを組み合せて使用する必要があります。

ただし、オペレーティング・システムの制約によって、CACHEMGR パラメータでのコンポーネント変更の効果が制限されることがあります。特に、オペレーティング・システムでプロセス当たりの仮想メモリーの制限が低い値に設定されている場合は、CACHEMGR の構成にかかわらず、多くのファイル・キャッシングを強制されます。

キャッシュ・マネージャ統計の使用の詳細は、36 ページの「SEND EXTRACT」を参照してください。

レポート・ファイルの基本統計の表示

キャッシュ・マネージャは、自身の初期化を完了すると、Extract レポート・ファイルに次のような統計を書き込みます。

```
CACHEMGR virtual memory values (may have been adjusted)
CACHESIZE: 1G
CACHEPAGEOUTSIZE (normal): 4M
PROCESS VM AVAIL FROM OS (min): 1.79G
CACHESIZEMAX (strict force to disk): 1.58G
```

条件 :

- CACHESIZE は、Extract がトランザクション・データのキャッシュに使用できる仮想メモリーの弱い制限を示します。これは、PROCESS VM AVAIL FROM OS (min) の値に基づいて動的に決定されます。CACHEMGR の CACHESIZE オプションを使用して制御できます。
- CACHEPAGEOUTSIZE (normal) はしきい値で、この値を超えると必要に応じてトランザクション・データをディスクにページングできます。CACHEMGR の CACHEPAGEOUTSIZE オプションを使用して制御できます。
- PROCESS VM AVAIL FROM OS (min) は、このプロセスが使用可能と判断したおおよその仮想メモリー量を示します。内部的な理由で、この量はオペレーティング・システムによって使用可能と表示される量より少ない場合があります。
- CACHESIZEMAX (strict force to disk) は、PROCESS VM AVAIL FROM OS および CACHESIZE から導出されます。これは、キャッシュ・マネージャがディスクへのページング候補のトランザクションを決定する手段と解釈できます。通常は、現在仮想メモリー・バッファが CACHEPAGEOUTSIZE を超えているトランザクションのみがページングの候補です。メモリー・リクエストの合計が CACHESIZE の値を超えると、キャッシュ・マネージャはディスクに書き込むトランザクションを探し、ページング候補のリストからトランザクションを選択します。ページング候補のトランザクションがすでにディスクにページングされていて、使用中の仮想メモリーが CACHESIZEMAX (strict force to disk) を超えている場合は、追加のバッファを必要とするすべてのトランザクションをページング候補にできます。このような方法で、常に仮想メモリーの可用性が確保されています。使用中のメモリーが CACHESIZEMAX を下回ると、CACHEPAGEOUTSIZE ルールが再度適用されます。

ページング・ディレクトリの特定

デフォルトでは、Oracle GoldenGate は、ディスクにスワップするデータを Oracle GoldenGate インストール・ディレクトリの dirtmp サブディレクトリに保管します。キャッシュ・マネージャは、ファイル・システムのすべての空き容量を使用可能とみなします。ディレクトリを割り当てるには、CACHEMGR パラメータの CACHEDIRECTORY オプションを使用します。

CACHEMGR 使用のガイドライン

- このパラメータは、z/OS 上の DB2 および NonStop SQL/MX を除くすべてのデータベースに有効です。
- 少なくとも 1 つの引数を指定する必要があります。CACHEMGR 単独では無効です。
- パラメータ・オプションは、任意の順番で指定できます。
- 1 つのパラメータ・ファイルでは、1 つの CACHEMGR パラメータのみの使用が許可されています。
- (ページ・ファイルのディレクトリを指定する以外に) このパラメータを正しく使用するためには、システムのプロファイル、およびアプリケーションから伝播されるトランザクションの種類を理解する必要があります。キャッシュ・マネージャは自己調整を行うため、標準の環境ではこのパラメータを変更する必要はありません。調整が必要と思われる場合は、Oracle サービス・リクエストをオープンしてください。詳細は、http://support.oracle.com を参照してください。

デフォルト なし

構文

```
CACHEMGR {
    [, CACHESIZE <size>]
    [, CACHEDIRECTORY <path> [<size>] [, ...]]
    [, CACHEPAGEOUTSIZE <size>]
    }
```

引数	説明
`CACHESIZE <size>`	トランザクション・データのキャッシングにできる仮想メモリー量 (キャッシュ・サイズ) の弱い制限を指定します。64 ビットシステムでは、デフォルトは 64GB です。32 ビットシステムでは、キャッシュ・サイズはキャッシュ・マネージャによって動的に決定されます。 メモリーはオンデマンドで割り当てられます。デフォルトでは、キャッシュ・マネージャは、オペレーティング・システムから提供される使用可能な仮想メモリー量を動的に判断し、適切なキャッシュ・サイズを決定します。使用可能な仮想メモリーは、レポート・ファイルの PROCESS VM AVAIL FROM OS 値にレポートされます。CACHESIZE 値は、プロセスが使用可能な仮想メモリー量よりも大きい、またはこの量に非常に近い場合、低減されます。ただし、大規模なアドレス領域を持つシステムでは、内部制限に到達した後にキャッシュ・マネージャが使用可能なメモリーを判断することはありません。

引数	説明
	CACHESIZE 値は、PROCESS VM AVAIL FROM OS 自体が 2 の 2 乗値の場合を除き (そのケースではその値の半分に値になります)、その値から切り捨てられて常に 2 の累乗値になります。指定されたサイズがデータによって消費されると、メモリー・マネージャはシステムに新しいメモリーをリクエストする前に、ディスクにデータをページングするか、古いバッファを再利用してメモリーの解放を試みます。 あるトランザクションが必要とするキャッシュ仮想メモリーが初期バッファ割当てを上回ると、このトランザクションの現在のキャッシュ・データのサイズ、新しいデータに必要なサイズ、すべてのトランザクションで使用されている仮想メモリー量などの要因に基づいて、キャッシュ・マネージャによって割り当てられる追加のキャッシュ・メモリー量が動的に決定されます。
`CACHEPAGEOUTSIZE <size>`	注意 : このパラメータを変更する前に、Oracle サポートに連絡してください。 このパラメータは、必要な場合、この値を超えたときにトランザクション・データをディスクにページングするしきい値を設定します。システム・オーバーヘッドの不必要な使用を回避するため、キャッシュ・マネージャは次の状況に当てはまる場合にのみトランザクションをファイル・システムにページングします。 ◆ トランザクションがページングの候補である。 ◆ CACHESIZE に基づくと、新しいメモリー・リクエストを満たすための十分な空き仮想メモリーがない。 例外的に、他のトランザクションがファイル・キャッシングの候補になることがあります。 デフォルト・サイズはおよそ 8MB です。1MB より小さい値を使用しないでください。 この値は、バイト、または次のいずれかの形式の GB、MB または KB で指定できます。 GB | MB | KB | G | M | K | gb | mb | kb | g | m | k システムにパフォーマンスの問題があると考えられる場合、このパラメータを変更することでパフォーマンスを改善できることがあります。それ以外の場合は、この値はデフォルトのままにしてください。多数の大規模な同時トランザクションや、1 つまたは 2 つの非常に大規模な長時間におよぶトランザクションが発生している環境では、ページングのしきい値がパフォーマンスの問題の原因になることがあります。たとえば、メモリーに制約があるシステム上で、多数の LOB データを含む大規模なトランザクションが行われている場合です。こうした状況では、キャッシュ・マネージャの統計を活用して、次の内容を確認します。 ◆ BLBO メモリー・プールにキャッシュされているトランザクションのサイズの分布。 ◆ ログ・リーダーのメモリー・プール。 ◆ ファイル・キャッシングに関する統計。

引数	説明
CACHEDIRECTORY <path> [<size>]	Oracle GoldenGate が必要なときに一時的にトランザクション・データをディスクに書き込むディレクトリの名前を指定します。このパラメータを指定しない場合のデフォルトは、Oracle GoldenGate インストール・ディレクトリの dirtmp サブディレクトリです。一時ファイル用のディレクトリは、Oracle データベース・ファイル・システムには配置できますが、AIX などの mmap() システム・コールをサポートしていないダイレクト I/O またはコンカレント I/O がマウントされたファイル・システムには配置できません。 ◆ <path> は、完全修飾ディレクトリ名です。 ◆ <size> には、指定したディレクトリに割当て可能な最大ディスク領域量を指定します。上限は、最大ファイル・サイズやファイル数など、ファイル・システムによって決められている値です。最小サイズは、強制的に 2GB になります。デフォルト値はありません。リソースの制限によって、Oracle GoldenGate が使用するスワップ領域を制限する必要がないかぎり、このオプションは使用しないでください。 ディレクトリごとに CACHEDIRECTORY 句を使用すると、複数のディレクトリを指定できます。ディレクトリの最大数は 100 です。 この値は、バイト、または次のいずれかの形式の GB、MB または KB で指定できます。 GB | MB | KB | G | M | K | gb | mb | kb | g | m | k

例 `CACHEMGR CACHESIZE 500MB, CACHEDIRECTORY /ggs/temp 2GB, CACHEDIRECTORY /ggs2/temp 2GB`

CHARSET

適用対象 Extract、Replicat、DEFGEN、Manager、GLOBALS

CHARSET パラメータでは、ローカルの Oracle GoldenGate インスタンスでのパラメータ・ファイルの文字セットを指定します。デフォルトでは、パラメータ・ファイルは、ローカル・オペレーティング・システムのデフォルトの文字セットで作成されます。CHARSET では、ローカル・プラットフォームが必要な文字をサポートしていない場合に、かわりに使用する文字セットを指定します。

CHARSET では、ローカル・プラットフォームがオペレーティング・システムのデフォルトの文字セットとしてマルチバイト文字をサポートしていない場合に、エスケープ・シーケンス (\uXXXX) を必要とせずに、パラメータ・ファイルでマルチバイト文字などのオペレーティング・システムの互換性のない文字を使用できます。

CHARSET は、あるシステムでパラメータ・ファイルを作成中に使用することもできますが、別の文字セットを使用した別のシステムで使用されます。異なる文字セット間の非互換性の可能性を回避するには、Oracle GoldenGate がパラメータ・ファイルを使用するシステムと同じシステム上にパラメータ・ファイルを作成する必要があります。

パラメータ・ファイルでの配置

CHARSET は、パラメータ・ファイルの最初の行に配置する必要があります。

GLOBALS ファイルでの使用方法

GLOBALS ファイルで使用する CHARSET では、すべてのローカル・プロセスのパラメータ・ファイルに対してデフォルトの文字セットを設定します。個別のパラメータ・ファイルで使用される CHARSET は、GLOBALS で設定されるデフォルト値より優先されます。

ネストされたパラメータ・ファイルでの使用方法

OBEY または INCLUDE パラメータを含むパラメータ・ファイルで CHARSET を使用できますが、参照パラメータ・ファイルは CHARSET 文字セットを継承しません。参照ファイル内のワイルドカード・オブジェクト名を読み取るために CHARSET 文字セットが使用されますが、エスエープ・シーケンス (\uXXXX) を使用して、参照ファイル内の他のすべての互換性のない文字を指定する必要があります。

デフォルト なし

構文

```
CHARSET <character_set>
```

引数	説明
<character_set>	サポートされている文字セット。

例

```
CHARSET UTF-8
```

CHARSETCONVERSION | NOCHARSETCONVERSION

適用対象 Replicat

CHARSETCONVERSION および NOCHARSETCONVERSION パラメータを使用して、異なる文字セットのデータベース間で文字型のデータをレプリケートする場合に Replicat が文字セットの変換を行うかどうかを制御します。ほとんど (すべてではない) の場合、Replicat はデフォルトで変換を行います。詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。

NOCHARSETCONVERSION は、Replicat が文字セットの変換を行わないようにします。データ整合性エラーのリスクが生じるため、通常は使用しません。NOCHARSETCONVERSION が適切と考えられる 1 つの例は、すべてのデータが ASCII で、ASCII と互換性のない文字が文字列にないことが確認されている場合です。ご使用の環境で NOCHARSETCONVERSION が適切かどうかが不明な場合、使用する前に Oracle サポートにご連絡ください。

> **注意** Replicat ではなく、Oracle ターゲットで変換を行うには、NLS_LANG 環境変数を *ソース・データベース*の文字セットに設定する SETENV パラメータを Replicat パラメータ・ファイルに含める必要があります。

デフォルト Oracle 以外のターゲットの場合、デフォルトは CHARSETCONVERSION です。Oracle ターゲットの場合、取得したソース・データが DB2 z/OS の EBCDIC でないかぎり、デフォルトは NOCHARSETCONVERSION です。EBCDIC の場合、デフォルトは CHARSETCONVERSION です。

構文

```
CHARSETCONVERSION | NOCHARSETCONVERSION
```

CHECKMINUTES

適用対象 Manager

CHECKMINUTES パラメータでは、Manager がメンテナンス・アクティビティを行う間隔を制御します。このパラメータの値を低くすると、トレイル・ファイルのロールオーバーを頻繁に行う場合にはパフォーマンスに大きな影響が及ぶことがあります。プロセスの異常終了などの他のイベントが発生した場合にも、メンテナンス・サイクルがトリガーされます。

デフォルト 10 分間隔

構文 `CHECKMINUTES <minutes>`

引数	説明
`<minutes>`	メンテナンス・アクティビティを実行する間隔 (分)。

例 `CHECKMINUTES 15`

CHECKPARAMS

適用対象 Extract および Replicat

CHECKPARAMS パラメータでは、パラメータ・ファイルの構文をテストします。テストを開始するには、次の手順を実行します。

1. パラメータ・ファイルを編集して CHECKPARAMS を追加します。
2. (オプション) 表を検証する場合には、NODYNAMICRESOLUTION パラメータを追加します。
3. プロセスを起動します。Oracle GoldenGate は、データ処理は行わず、構文を検証します。NODYNAMICRESOLUTION が存在する場合、Oracle GoldenGate はデータベースに接続して、TABLE または MAP で指定されている表が存在するかどうか検証します。構文エラーがある場合、プロセスはエラー 190 で異常終了します。構文にエラーがない場合、プロセスは停止し、パラメータを正常に処理したことを示すメッセージをレポート・ファイルに書き込みます。
4. 次のいずれかを実行します。
   - テストが成功した場合は、パラメータ・ファイルを編集して CHECKPARAMS パラメータ (および使用している場合は NODYNAMICRESOLUTION パラメータ) を削除した後、プロセスを起動して処理を再開します。
   - テストが失敗した場合は、パラメータ・ファイルを編集し、レポートの結果に基づいて構文を修正した後、NODYNAMICRESOLUTION を削除してプロセスをもう一度起動します。

CHECKPARAMS は、パラメータ・ファイル内の任意の位置に配置できます。

デフォルト なし

構文 CHECKPARAMS

CHECKPOINTSECS

適用対象 Extract および Replicat

CHECKPOINTSECS パラメータでは、Extract および Replicat がルーチンのチェックポイントを作成する頻度を制御します。

- この値を低くすると、チェックポイントの作成頻度が増えます。この場合、プロセスが失敗したときに再処理が必要なデータ量が減りますが、データのディスクへの書込み頻度が増えるため、パフォーマンスが低下することがあります。
- この値を高くすると、チェックポイントの作成頻度が減ります。この場合、パフォーマンスは向上しますが、プロセスが失敗したときに再処理が必要なデータ量は増加します。Extract チェックポイントの作成頻度を少なくするときは、データの再処理が必要になるときに備えて、常にトランザクション・ログを利用できる状態にしてください。

> **注意** Replicat は、ルーチン・チェックポイントの他に、トランザクションをコミットするときにもチェックポイントを作成します。

CHECKPOINTSECS を変更するときは、最初に必ず Oracle サービス・リクエストをオープンしてください。詳細は、http://support.oracle.com を参照してください。

デフォルト 10 秒

構文

```
CHECKPOINTSECS <seconds>
```

引数	説明
<seconds>	チェックポイントを発行するまでの待機する時間 (秒)。

例

```
CHECKPOINTSECS 20
```

CHECKPOINTTABLE

適用対象 GLOBALS

GLOBALS パラメータ・ファイルの CHECKPOINTTABLE パラメータでは、1 つ以上の Oracle GoldenGate インスタンス内のすべての Replicat グループが使用できる、デフォルトのチェックポイント表の名前を指定します。ADD REPLICAT コマンドで作成されたすべての Replicat グループは、このコマンドの CHECKPOINTTABLE オプションを使用して上書きされないかぎり、デフォルトでこの表を使用します。

チェックポイント表を作成するには、GGSCI で ADD CHECKPOINTTABLE コマンドを使用します。

デフォルト なし

構文

```
CHECKPOINTTABLE <owner.table>
```

引数	説明
owner.table	チェックポイント表の所有者および名前。

例

```
CHECKPOINTTABLE ggs.chkpt
```

CMDTRACE

適用対象 Extract および Replicat

CMDTRACE パラメータでは、レポート・ファイルにマクロ展開手順を表示します。このパラメータは、異なるマクロに異なるオプションを設定するために、パラメータ・ファイルで複数回使用できます。

デフォルト OFF

構文 CMDTRACE [ON | OFF | DETAIL]

引数	説明
ON	マクロ展開の表示を有効化します。
OFF	マクロ展開の表示を無効化します。
DETAIL	マクロ展開の詳細な表示を生成します。

例 次の例では、#testmac を起動する前にトレースを有効化し、マクロの実行後に無効化します。

```
MACRO #testmac
BEGIN
col1 = col2,
col3 = col4
END;
...
CMDTRACE ON
MAP test.table2 , TARGET test.table2,
COLMAP (#testmac);
CMDTRACE OFF
```

COLMATCH

適用対象 Extract および Replicat

COLMATCH パラメータでは、列マッピングのグローバル・ルールを作成します。COLMATCH ルールは、COLMATCH 文に続くすべての TABLE または MAP 文に適用されます。グローバル・ルールを以降の TABLE または MAP エントリに対して無効にするには、RESET オプションを使用します。

COLMATCH を使用すると、同様の構造を持っていて、同じデータ・セットを異なる列名に保持している表同士をマップできます。COLMATCH では、個別の TABLE または MAP 文で COLMAP 句を使用するよりも、このタイプの列を効率的にマップできます。

COLMATCH では、次のことを実行できます。

- 列名に基づいて明示的にマップする
- 名前の接頭辞またはサフィックスを無視する

名前が異なるソース列とターゲット列をマッピングする場合は、COLMATCH か、TABLE または MAP 文の COLMAP 句を使用する必要があります。

列マッピングの詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。

デフォルト なし

構文

```
COLMATCH
{NAMES <target column> = <source column> |
PREFIX <prefix> |
SUFFIX <suffix> |
RESET }
```

引数	説明
`NAMES <target column> = <source column>`	ターゲットおよびソースの列名を指定します (例 : CUSTOMER_CODE および CUST_CODE)。
`PREFIX <prefix>`	無視する列名接頭辞を指定します。たとえば、ORDER_ID という名前のターゲット列を P_ORDER_ID という名前のソース列にマップする場合は、次のように指定します。 `COLMATCH PREFIX P_`
`SUFFIX <suffix>`	無視する列名サフィックスを指定します。たとえば、CUST_CODE_K という名前のターゲット列を CUST_CODE という名前のソース列にマップする場合は、次のように指定します。 `COLMATCH SUFFIX _K`
`RESET`	前に定義済の COLMATCH ルールを以降の TABLE または MAP 文で無効化します。

例 1 `COLMATCH NAMES CUSTOMER_CODE = CUST_CODE`

例 2 `COLMATCH PREFIX P_`

例 3 `COLMATCH SUFFIX _K`

例 4 `COLMATCH RESET`

COMMENT | --

適用対象 Manager、Extract、Replicat

COMMENT パラメータまたは 2 連ハイフン (--) では、パラメータ・ファイル内のコメントを示します。COMMENT または 2 連ハイフン以降の同一行内の記述は、処理中に無視されます。

COMMENT または 2 連ハイフンを使用して、パラメータ・ファイル内のどの位置にもコメントを記述できます。次の行にコメントが続く場合は、次の行頭に COMMENT キーワードまたは 2 連ハイフンをもう一度記述する必要があります。

注意 **同期されている表の列に "comment" という単語が含まれている場合は、COMMENT パラメータと競合する可能性があります。その場合には 2 連ハイフンを使用してください。**

デフォルト なし

構文

```
{COMMENT <comment text>} | {-- <comment text>}
```

例 1 `COMMENT GoldenGate param file for fin Extract group.`

例 2 `-- GoldenGate param file for fin Extract group.`

COMPRESSDELETES | NOCOMPRESSDELETES

適用対象 Extract

COMPRESSDELETES および NOCOMPRESSDELETES パラメータでは、削除操作の際に列をトレイル・レコードに書き込む方法を制御します。

デフォルトの COMPRESSDELETES では、削除操作の場合、Extract は主キーのみをトレイルに書き込みます。主キーのみを書き込むと、処理が必要なデータ量を制限しながら、適切なターゲット・レコードの削除に十分な情報を提供できます。

NOCOMPRESSDELETES では、すべての列をトレイルに送信します。表定義に主キーまたは一意索引が含まれていない場合は、これがデフォルトになります。TABLE の KEYCOLS オプションで代替キーが定義されている場合は、実際のキーが定義されているかどうかにかかわらず、これらの列がトレイルに書き込まれます。

Oracle GoldenGate がサポートするすべてのプラットフォームで DB2 データベースに競合の検出および解決 (CDR) 機能を使用する場合、NOCOMPRESSDELETES も必要です。CDR の詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。

COMPRESSDELETES および NOCOMPRESSDELETES は、パラメータ・ファイル内のすべての TABLE 文にグローバルに適用することも、各 TABLE 文に個別スイッチとして使用することもできます。

これらのパラメータは、データ・ポンプには影響しません。

デフォルト `COMPRESSDELETES`

構文 `COMPRESSDELETES | NOCOMPRESSDELETES`

COMPRESSUPDATES | NOCOMPRESSUPDATES

適用対象 Extract

COMPRESSUPDATES および NOCOMPRESSUPDATES パラメータでは、Extract が更新操作の際に列をトレイル・レコードに書き込む方法を制御します。

デフォルトの COMPRESSUPDATES では、更新操作の場合、Extract は主キーおよび行の変更された列のみをトレイルに書き込みます。これらを書き込むと、処理が必要なデータ量を制限しながら、適切なターゲット・レコードの更新に十分な情報を提供できます。

NOCOMPRESSUPDATES では、すべての列をトレイルに送信します。表定義に主キーまたは一意索引が含まれていない場合は、これがデフォルトになります。TABLE パラメータの KEYCOLS オプションで表に代替キーが定義されている場合は、実際のキーが定義されているかどうかにかかわらず、これらの列がトレイルに書き込まれます。

Oracle GoldenGate がサポートするすべてのプラットフォームで DB2 データベースに競合の検出および解決 (CDR) 機能を使用する場合、NOCOMPRESSUPDATES も必要です。CDR の詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。

COMPRESSUPDATES および NOCOMPRESSUPDATES は、パラメータ・ファイル内のすべての TABLE 文にグローバルに適用されます。

このパラメータは、次のデータベースのみをサポートしています。

- DB2 LUW
- DB2 z/OS
- Teradata バージョン 12 以上
- SQL Server
- Sybase

COMPRESSUPDATES および NOCOMPRESSUPDATES は、データ・ポンプには影響しません。

デフォルト COMPRESSUPDATES

構文 COMPRESSUPDATES | NOCOMPRESSUPDATES

CUSEREXIT

適用対象 Extract および Replicat

CUSEREXIT パラメータを使用して、Oracle GoldenGate 処理内の定義済イグジット・ポイントで、Windows DLL または UNIX 共有オブジェクトから、C プログラミング・コードで記述されたカスタム・イグジット・ルーチンをコールします。ユーザー・イグジット・ルーチンは、Extract および Replicat プロセスから様々なイベントおよび情報を受け入れ、リクエストどおりに情報を処理し、コール元 (このルーチンをコールした Oracle GoldenGate プロセス) にレスポンスと情報を返す必要があります。

ユーザー・イグジットは、Oracle GoldenGate ソリューション内で使用可能なデータ変換ファンクションのかわりとしても、こうしたファンクションと一緒にも使用できます。

ユーザー・イグジットの作成および実装の詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。

デフォルト なし

構文

```
CUSEREXIT <DLL or shared object name> <routine name>
[, PASSTHRU]
[, INCLUDEUPDATEBEFORES]
[, PARAMS "<startup string>"]
```

引数	説明
<DLL or shared object name>	ユーザー・イグジット・ファンクションを含む Windows DLL または UNIX 共有オブジェクトの名前。
<routine name>	実行するイグジット・ルーチン名。

引数	説明
PASSTHRU	Extract データ・ポンプにのみ有効です。データベースが不要で、出力トレイルが許可されていないとみなされます。ユーザー・イグジットがすべての処理を実行し、Extract がレコードをスキップするとみなされます。Extract は、ユーザー・イグジットにレコードを渡す前にすべての必要なデータ・マッピングを行います。 レスポンスのステータスは、EXIT_OK_VAL ではなく EXIT_PROCESSED_REC_VAL になります。すべての処理統計は、レコードが Oracle GoldenGate によって処理されたように更新されます。
INCLUDEUPDATEBEFORES	列値のビフォア・イメージをユーザー・イグジットに渡します。このパラメータを使用するときは、requesting_before_after_ind フラグをサポートするコールバック・ファンクション内でこのフラグを BEFORE_IMAGE_VAL に設定し、ビフォア・イメージを明示的にリクエストする必要があります。この設定を行わないと、アフター・イメージのみがユーザー・イグジットに渡されます。デフォルトでは、Oracle GoldenGate はアフター・イメージのみを使用します。 データ・ポンプまたは Replicat からコールされるユーザー・イグジットに対して INCLUDEUPDATEBEFORES を使用する場合は、常にプライマリ Extract プロセスで GETUPDATEBEFORES パラメータを使用することにより、ビフォア・イメージを取得してトレイルに書き込み、ユーザー・イグジットで process_record イベントを発生させます。プライマリ Extract にもユーザー・イグジットが含まれている場合は、GETUPDATEBEFORES によってビフォアおよびアフター・イメージ両方が別々の EXIT_CALL_PROCESS_RECORD イベントとしてユーザー・イグジットに送信されます。 ユーザー・イグジットが (トランザクション・ログを読み込む) プライマリ Extract からコールされる場合、この Extract に必要なのは INCLUDEUPDATEBEFORES のみです。このケースでは、他のダウンストリームの Oracle GoldenGate プロセスがトレイルへのビフォア・イメージ書込みを必要としないかぎり、GETUPDATEBEFORES は必要ありません。INCLUDEUPDATEBEFORES を指定しても、ビフォア・イメージはトレイルに書き込まれません。
PARAMS "<startup string>"	指定した文字列を起動時に渡します。プロパティ・ファイル、起動パラメータ、その他の文字列を渡すために使用できます。文字列は、二重引用符で囲む必要があります。 この文字列のデータは、EXIT_CALL_START exit_params_def.function_param のユーザー・イグジットに渡されます。PARAMS で、引用符で囲まれた文字列が指定されない場合、exit_params_def.function_param は NULL です。

例 1 `CUSEREXIT userexit.dll MyUserExit`

例 2 `CUSEREXIT userexit.dll MyUserExit, PARAMS "init.properties"`

例 3 `CUSEREXIT userexit.dll MyUserExit, INCLUDEUPDATEBEFORES, & PARAMS "init.properties"`

例 4
```
CUSEREXIT userexit.dll MyUserExit, INCLUDEUPDATEBEFORES, PASSTHRU, & PARAMS
"init.properties"
```

例 5
```
CUSEREXIT cuserexit.dll MyUserExit, & PASSTHRU, & INCLUDEUPDATEBEFORES, & PARAMS "Some
text to start with during startup"
```

DBOPTIONS

適用対象 Extract および Replicat

DBOPTIONS パラメータでは、データベース・オプションを指定します。これはグローバル・パラメータで、パラメータ・ファイル内のすべての TABLE または MAP 文に適用されます。

DBOPTIONS は、TARGETDB か SOURCEDB パラメータ文、または USERID 文 (あるいはその両方) の前に配置する必要があります。一部の DBOPTIONS オプションは、Extract または Replicat のみに適用されます。

デフォルト なし

構文

```
DBOPTIONS
[ALLOWLOBDATATRUNCATE | NOALLOWLOBDATATRUNCATE]
[ALLOWUNUSEDCOLUMN]
[CATALOGCONNECT | NOCATALOGCONNECT]
[CONNECTIONPORT <port>]
[DECRYPTPASSWORD <shared secret> ENCRYPTKEY {DEFAULT | <key name>}]
[DEFERREFCONST]
[DISABLECOMMITNOWAIT]
[DISABLELOBCACHING]
[EMPTYLOBSTRING '<string>']
[FETCHBATCHSIZE <num_recs>]
[FETCHLOBS | NOFETCHLOBS]
[HOST <host ID>]
[LIMITROWS | NOLIMITROWS]
[LOBBUFSIZE]
[LOBWRITESIZE <size>]
[SHOWINFOMESSAGES]
[SHOWWARNINGS]
[SPTHREAD | NOSPTHREAD]
[SUPPRESSTRIGGERS | NOSUPPRESSTRIGGERS]
[TDSPACKETSIZE <bytes>]
[TRUSTEDCONNECTION]
[USEODBC | USEREPLICATIONUSER]
[XMLBUFSIZE <buffer size>]
```

引数	説明
ALLOWUNUSEDCOLUMN	Oracle の Extract に有効です。Replicat が未使用の列を持つ表を検出したときに異常終了することを防ぎます。Replicat は、異常終了せずに処理を継続し、警告を生成します。 このパラメータを使用する場合は、ターゲットにも同じ未使用の列が存在している必要があります。この列が存在しない場合には、正しいメタデータ・マッピングを実行できるように、Replicat にこの表のソース定義ファイルを指定する必要があります。デフォルトでは、Extract は未使用の列を検出すると異常終了します。

引数	説明
ALLOWLOBDATATRUNCATE | NOALLOWLOBDATATRUNCATE	Sybase および MySQL の Replicat に有効です。ALLOWLOBDATATRUNCATE では、レプリケートする LOB データがターゲットの CHAR、VARCHAR、BINARY または VARBINARY 列に対して大きすぎる場合に Replicat が異常終了することを防ぎます。この LOB データは、ターゲット列の最大サイズに切り捨てられ、エラー・メッセージや*警告は生成されません*。 デフォルトは NOALLOWLOBDATATRUNCATE で、レプリケートする LOB が大きすぎる場合に、Replicat を異常終了させ、エラー・メッセージを生成します。
CATALOGCONNECT | NOCATALOGCONNECT	ODBC データベースの Extract および Replicat に有効です。デフォルトでは、Oracle GoldenGate はカタログ問合せのために新しい接続を作成しますが、NOCATALOGCONNECT を使用すると新しい接続の作成を防止できます。z/OS 上の DB2 の場合、NOCATALOGCONNECT では、MVS DB2 初期化パラメータ mvsattachtype が CAF に設定されているときに、Oracle GoldenGate による複数の接続の試みを防ぎます。CAF モードでは複数の接続がサポートされていないため、Oracle GoldenGate はオープンしている接続のコミットを受信するまで、システム・カタログ表領域にコミット・ロックを発行する可能性があります。コミット・ロックを防ぐために、Oracle GoldenGate では、複数の接続をサポートする RRSAF(mvsattachtype=RRSAF) を使用することをお薦めします。
CONNECTIONPORT <port>	マルチデーモン MySQL の Replicat に有効です。Replicat が接続する必要があるインスタンスの TCP/IP ポートを指定します。
DECRYPTPASSWORD <shared secret> <algorithm> ENCRYPTKEY {<keyname> | DEFAULT}	Extract に有効です (Oracle)。 Oracle Transparent Data Encryption (TDE) を使用して暗号化された REDO ログ・データを復号化する、TDE 鍵を復号化する*共通鍵* (パスワード) を指定します。TDE 鍵は、まず、共通鍵を鍵として使用して Oracle サーバー内で暗号化された後、Extract に配信され、そこで同じ共通鍵を使用して復号化されます。共通鍵は、Oracle サーバー・セキュリティ担当者によって、Oracle Wallet または Hardware Security Module で作成する必要があります。Oracle GoldenGate 管理者のみが、共通鍵を知っている必要があります。 復号化オプションを使用するには、まず GGSCI で ENCRYPT PASSWORD コマンドを使用して、暗号化された共通鍵を生成し、ENCKEYS ファイルを作成する必要があります。

引数	説明
	パラメータ・オプション : ◆ <shared secret> は、ENCRYPT PASSWORD コマンドの結果からコピーされる、暗号化された共通鍵 (パスワード) です。 ◆ <algorithm> は、パスワードの暗号化に使用した暗号化アルゴリズムを、AES128、AES192、AES256 または BLOWFISH の中から指定します。 ◆ ENCRYPTKEY <keyname> は、ENCKEYS 参照ファイル内のユーザー作成の暗号化鍵の論理名を指定します。ENCRYPT PASSWORD が KEYNAME <keyname> オプションとともに使用された場合に使用します。ENCKEYS ファイルをローカル・システムに作成する必要があります。 ◆ ENCRYPTKEY DEFAULT は、Oracle GoldenGate に、ランダム鍵を使用するように指示します。ENCRYPT PASSWORD が KEYNAME DEFAULT オプションとともに使用された場合に使用します。 TDE をサポートする Extract の構成の詳細は、『Oracle GoldenGate *Oracle インストレーションおよびセットアップ・ガイド*』を参照してください。 ENCKEYS などの Oracle GoldenGate 暗号化オプションの詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』のセキュリティの章を参照してください。
DEFERREFCONST	Oracle の Replicat に有効です。Replicat トランザクションがコミットされるまで、Oracle ターゲット・データベースによるカスケード削除およびカスケード更新の参照整合性制約のチェックおよび強制を延期するように、DEFERRABLE に制約を設定します。この時点で制約違反がある場合、エラーが生成されます。 ターゲット表での制約を無効化するか DEFERRED に設定するかわりに、DEFERREFCONST を使用できます。使用すると、DEFERREFCONST は、DEFERABLE および NOT DEFERABLE の両方の制約を延期します。DEFERREFCONST は、Replicat が処理するすべてのトランザクションに適用します。 この機能をサポートしていない Oracle リリースとともに使用する場合、DEFERREFCONST は無視され、GoldenGate ログに通知は返されません。コミット操作でのエラーを処理するために、パラメータ・ファイルのルート・レベルで REPERROR を使用し、TRANSDISCARD または TRANSEXCEPTION オプションを指定できます。
DISABLECOMMITNOWAIT	Oracle の Replicat に有効です。Replicat による非同期 COMMIT の使用を無効化します。非同期 COMMIT 文には、NOWAIT オプションが含まれます。 DISABLECOMMITNOWAIT を使用する場合、Replicat は、標準の同期 COMMIT (WAIT オプションを使用する COMMIT) を発行します。

引数	説明
DISABLELOBCACHING	Oracle の Replicat に有効です。Oracle の LOB キャッシング・メカニズムを無効にします。デフォルトでは、Replicat は Oracle の LOB キャッシング・メカニズムを有効にします。
[EMPTYLOBSTRING '<string>']	Replicat に有効です。ターゲットにレプリケートされる、Sybase IMAGE または TEXT 値などの空の (ゼロ長)LOB 列を、文字列値に置換します。デフォルトでは、Oracle GoldenGate は、空の列をターゲットで NULL に設定し、ターゲット・データベースが LOB 列に NULL を許可していない場合は異常終了します。このオプションは、Replicat の異常終了を防ぐために使用します。 '<string>' には、列が受け入れる任意の文字列を指定し、一重引用符で囲みます。デフォルトは NULL です。 例 : DBOPTIONS EMPTYLOBSTRING 'empty'
FETCHBATCHSIZE <num_recs>	Oracle、DB2、SQL/MX、Sybase、SQL Server、Sybase および Teradata の Extract に有効です。初期ロードのパフォーマンスを高めるために、1 行ごとのフェッチのかわりに配列フェッチを有効にします。有効な値は、フェッチ当たり 0 ～ 1000000 レコードです。デフォルトは 1000 です。バッチ・サイズが非常に小さくなるか非常に大きくなると、パフォーマンスは低下します。表に LOB データが含まれている場合、Extract は 1 行フェッチ・モードに戻り、その後バッチ・フェッチモードを再開します。
HOST <host id>	マルチデーモン MySQL の Replicat に有効です。Replicat が接続する必要があるインスタンスのホストの DNS 名または IP アドレスを指定します。
FETCHLOBS | NOFETCHLOBS	z/OS 上の DB2 および DB2 LUW に有効です。表の LOB オプションが NOT LOGGED に設定されている場合に、データベース表からの LOB の直接のフェッチを抑止します。NOT LOGGED が設定されている場合、この列の値はトランザクション・ログからは入手できず、表自体からのみ取得できます。デフォルトでは、Oracle GoldenGate は LOB への変更をトランザクション・ログから取得します。デフォルトは、FETCHLOBS です。

引数	説明

LIMITROWS | NOLIMITROWS	MySQL、Oracle、SQL Server および Sybase の Replicat に有効です。LIMITROWS では、ターゲット表に主キーまたは一意キーが含まれていない場合に、同一の Replicat SQL 文によって複数の行を更新または削除することを防ぎます。
	MySQL では、UPDATE または DELETE 文の LIMIT 1 句を使用します。
	Oracle では、WHERE 句がすでに存在するかどうかに応じて、次のいずれかの句を追加することにより、Replicat が使用する WHERE 句を変更します。
	`WHERE ROWNUM = 1`
	または
	`AND ROWNUM = 1`
	SQL Server および Sybase では、UPDATE または DELETE 文の前に SET ROWCOUNT 1 句を使用します。
	
	NOLIMITROWS では、同一の Replicat SQL 文によって複数の行を更新または削除することを許可します。このオプションは、Oracle の OCI を使用しているときは機能しません。
	デフォルトは LIMITROWS です。LIMITROWS および NOLIMITROWS は、パラメータ・ファイル内のすべての MAP 文にグローバルに適用されます。

LOBBUFSIZE <bytes>	Oracle の Extract に有効です。Oracle オブジェクト・タイプの各埋込み LOB 属性に割り当てるメモリー・バッファ・サイズを決定します。有効な値は、1024 ～ 10485760 バイトです。デフォルトは 1048576 バイトです。 埋込み LOB の長さが指定の LOBBUFSIZE サイズを超えると、次のようなエラー・メッセージが生成されます。 `GGS ERROR  ZZ-0L3  Buffer overflow, needed: 2048, allocated: 1024.`

引数	説明
LOBWRITESIZE <size>	Oracle の Replicat に有効です。Replicat がターゲット・データベースに書き込む各 LOB のフラグメント・サイズを指定します。LOB データは、このサイズに達するまでバッファに保持されます。LOB はフラグメントでデータベースに書き込む必要があるため、より大きなブロックで書き込むことで過度の I/O を防止できます。この値が大きいほど、LOB 全体をデータベースに書き込むための Replicat からデータベース・サーバーへの I/O コールは少なくなります。 Oracle LOB フラグメント・サイズの倍数を指定します。指定した値は、必要に応じて Oracle LOB フラグメント・サイズの倍数に切り上げられます。デフォルトの LOB 書込みサイズは 32K です。有効な値は、2,048 ～ 1,048,576 バイト (1MB) です。 デフォルトでは、Replicat は Oracle の LOB キャッシング・メカニズムを有効にします。Oracle の LOB キャッシングを無効にするには、DBOPTIONS の DISABLELOBCACHING オプションを使用します。

SHOWINFOMESSAGES	Sybase の Extract および Replicat に有効です。エラー・ログに次の Sybase サーバー・メッセージを出力します。
	`0: /* General informational message */`
	`5701: /* Changed Database Context */`
	`5703: /* Changed language setting */`
	`5704: /* Changed client character set */`
	`7326: /* Non ANSI Escaping */`
	Oracle GoldenGate 処理に影響がないため、通常はこれらのメッセージは出力されません。

SHOWWARNINGS	Sybase の Extract および Replicat に有効です。重大度レベル 10 を上回る Sybase サーバー・メッセージのロギングを有効化します。これらのメッセージは、Sybase がデータ変更を伴う修正処理を実行するときに、デバッグに役立ちます。
SPTHREAD | NOSPTHREAD	Extract および Replicat に有効です。ストアド・プロシージャに別のデータベース接続スレッドを作成します。デフォルトは NOSPTHREAD です。

引数	説明
SUPPRESSTRIGGERS | NOSUPPRESSTRIGGERS	Oracle の Replicat に有効です。Replicat セッション中にトリガーが起動するかどうかを制御します。 SUPPRESSTRIGGERS は、Oracle GoldenGate でレプリケーションするように構成されているトリガーがターゲット・オブジェクト上で起動することを防ぎます。 トリガーを手動で無効にするかわりに、次の Oracle リリースでは SUPPRESSTRIGGERS を使用できます。 ◆ Oracle 10.2.0.5 および以降の 10.2.0.5 へのパッチ ◆ Oracle 11.2.0.2 および以降の 11*g*R2 リリース SUPPRESSTRIGGERS は、11*g*R1 では使用できません。 SUPPRESSTRIGGERS を使用するには、Replicat ユーザーが Oracle Streams 管理者であることが必要ですが、この権限は次を起動することによって付与できます。 Oracle 10.2.0.5 およびパッチの場合、 dbms_streams_auth.grant_admin_privilege を使用します。 Oracle 11*g*R2 リリースの場合、 dbms_goldengate_auth.grant_admin_privilege を使用します。 これらの手順は、Oracle データベース・インストールの一部です。詳細は、データベースのマニュアルを参照してください。 デフォルトは NOSUPPRESSTRIGGERS で、ターゲット・トリガーの起動を許可します。 SOURCEDB および USERID パラメータは、SUPPRESSTRIGGERS が含まれる DBOPTIONS 文の前に指定する必要があります。

引数	説明
TDSPACKETSIZE <bytes>	Sybase の Extract および Replicat に有効です。Sybase ターゲットへのレプリケーションで使用する TDS パケット・サイズを設定します。 有効な値 : ◆ Sybase バージョン 12.5.4(注意 : このバージョンは、Oracle GoldenGate 11.2.1 よりサポート対象外です): 512 から 65024 まで デフォルトは、Extract の場合は 0、Replicat の場合は 512 です。 ◆ Sybase15 以上 : 2048 から 65024 まで デフォルトは、Extract の場合は 0、Replicat の場合は 2048 です。 値は 512 の倍数である必要があります。Sybase Adaptive Server の max network packet size および additional network memory パラメータに設定する値の範囲は、TDSPACKETSIZE で設定する値をサポートしている必要があります。 注意 : max network packet size の値が大きいほど、データベース・サーバーがネットワーク・データへの割当てに必要とするメモリー (additional network memory で設定される) が大きくなります。 最適なパフォーマンスのために、ネットワーク上の基礎となるパケット・サイズで効率的に動作するサーバー・パケット・サイズを選択してください。この手順の目的は、次のとおりです。 ◆ ネットワークに対するサーバーの読取りおよび書込み数を減らす。 ◆ ネットワーク・パケットでの未使用の領域を減らし、ネットワークのスループットを向上させる。 たとえば、お使いのネットワークのパケット・サイズで 1500 バイトのデータが処理される場合、サーバー上のパケット・サイズは、1536 (512*3) に設定するよりも、1024 (512*2) に設定する方が、転送のパフォーマンスを向上させることができます。 最適なパフォーマンスのために、次の構成で起動します。 DBOPTIONS TDSPACKETSIZE 8192 TDSPACKETSIZE が含まれる DBOPTIONS パラメータは、パラメータ・ファイル内の SOURCEDB または TARGETDB パラメータより前に配置する必要があります。
TRUSTEDCONNECTION	SQL Server の Extract および Replicat に有効です。Oracle GoldenGate に、trusted connection = yes を使用して接続させます。このオプションを使用する前に Oracle サポートに連絡してください。詳細は、http://support.oracle.com を参照してください。

引数	説明
USEODBC	SQL Server の Replicat に有効です。Replicat が ODBC を使用して DML 操作を実行するように構成します。デフォルトでは、OLE DB を使用します。USEREPLICATIONUSER を有効化しているときはこの設定は無効になり、Replicat は異常終了します。 **注意** : Replicat は、データベース・カタログに接続してメタデータを取得する場合は、常に ODBC を使用します。
USEREPLICATIONUSER	SQL Server の Replicat に有効です。SQL Server レプリケーション・ユーザーとしてターゲット DML 操作を実行するように Replicat を構成します。レプリケーション・ユーザーは、SQL Server ユーザーまたはアカウントではなく、データベース接続のプロパティです。USEREPLICATIONUSER は、SQL Server の NOT FOR REPLICATION フラグを有効化します。 レプリケーション・ユーザーを使用する場合は、データの整合性に対する次の懸念を解消する必要があります。 ◆ ターゲットの IDENTITY シードが更新されません。ターゲットが読取り専用の場合を除き、主キー違反の回避のためにパーティショニング・スキームが必要です。 ◆ 外部キー制約が強制されません。 ◆ ON UPDATE CASCADE、ON DELETE CASCADE およびトリガーが無効化されます。重複する操作が回避されるのでこの無効化は Replicat には有益ですが、ターゲット・アプリケーションにとっては適切でなく、データ整合性の確保のために制約またはトリガーのコードの変更が必要になることがあります。 ◆ CHECK 制約が強制されないため、ターゲット上でデータ整合性を確保できません。 これらの考慮事項の詳細は、『Oracle GoldenGate *SQL Server インストレーションおよびセットアップ・ガイド*』を参照してください。 デフォルトでは、USEREPLICATIONUSER は無効化されており、OLE DB を使用します。USEREPLICATIONUSER は、送信パフォーマンスを高める必要がある場合にのみ使用することをお薦めします。USEODBC を有効化しているときはこの設定は無効になり、Replicat は異常終了します。
XMLBUFSIZE <bytes>	Oracle の Extract に有効です。SDO_GEORASTER オブジェクト・タイプの sys.xmltype 属性から抽出された XML データを保持するメモリー・バッファのサイズを設定します。デフォルトは 1048576 バイト (1MB) です。データがデフォルトのバッファ・サイズを超えると、Extract は異常終了します。その場合は、バッファ・サイズを増やしてから、Extract を再起動してください。有効な値の範囲は、1024 ～ 10485760 バイトです。

例 1 `DBOPTIONS HOST 127.0.0.1, CONNECTIONPORT 3307`

例 2 `DBOPTIONS DECRYPTPASSWORD AACAAAAAAAAAAAIALCKDZIRHOJBHOJUH ENCRYPTKEY DEFAULT`

例 3 `DBOPTIONS TDSPACKETSIZE 2048`

例 4 DBOPTIONS FETCHBATCHSIZE 2000

例 5 DBOOPTION XMLBUFSIZE 2097152

DDL

適用対象 Extract および Replicat

DDL パラメータでは、次のことを行います。

- DDL サポートの有効化
- DDL 操作のフィルタ
- DDL レコードに基づく処理アクションの構成

オプションを指定せずに使用する場合、DDL パラメータは、フィルタリングを実行せず、すべての DDL 操作を次のように伝播します。

- Extract パラメータとして、サポートされているすべてのデータベース・オブジェクト上で生成されたすべての DDL 操作を取得し、トレイルに送信します。
- Replicat パラメータとして、Oracle GoldenGate トレイルから受信したすべての DDL 操作をレプリケートし、ターゲットに適用します。このパラメータを使用しない場合のデフォルトの動作と同じです。

オプションとともに使用すると、DDL パラメータは、次に基づいて DDL 操作を包含または除外するフィルタリング・エージェントとして機能します。

- 範囲
- オブジェクト・タイプ
- 操作タイプ
- オブジェクト名
- DDL コマンド構文またはコメントの文字列 (またはその両方)

パラメータ・ファイルでは 1 つの DDL パラメータのみを使用できますが、DDL の複数の包含および除外オプションを組み合せて、必要なレベルまで DDL をフィルタリングすることができます。

- DDL フィルタリング・オプションは、トランザクション・ソースから取得するプライマリ Extract には有効ですが、データ・ポンプ Extract には無効です。
- 組み合せて使用する場合、複数のフィルタ・オプション指定は "AND" 文として論理的に結合されます。
- 複数のオプションで指定されたすべてのフィルタ基準を満たさなければ、DDL 文はレプリケートされません。
- 複雑な DDL フィルタリング基準を使用する場合は、本番環境で使用する前にテスト環境で構成をテストすることをお薦めします。

適用対象 Extract および Replicat

次のプロセスでは、DDL を使用しないでください。

- Extract データ・ポンプ
- VAM ソート Extract(Teradata ソース・データベース)

これらのプロセス・タイプでは、DDL のマッピングまたは変換が許可されず、PASSTHRU モードで自動

的に DDL レコードが伝播されます (288 ページを参照してください)。特定の名前のソース表で実行される DDL (ALTER TABLE TableA... など) は、Replicat によって同一の名前の表に適用されます (ALTER TABLE TableA)。ALTER TABLE TableB としてマッピングすることはできません。

Oracle GoldenGate DDL サポートの使用の詳細は、必要に応じて Oracle or Teradata 用の Oracle GoldenGate のドキュメントを参照してください。

構文

```
DDL [
{INCLUDE | EXCLUDE}
    [, MAPPED | UNMAPPED | OTHER | ALL]
    [, OPTYPE <type>]
    [, OBJTYPE '<type>']
    [, OBJNAME <name>]
    [, INSTR '<string>']
    [, INSTRCOMMENTS '<comment_string>']
    [, STAYMETADATA]
    [, EVENTACTIONS (<action specification>)
]
[...]
```

表 35　DDL 包含および除外オプション

オプション	説明

`INCLUDE | EXCLUDE`	INCLUDE および EXCLUDE では、包含または除外句の開始位置を指定します。
	◆ 包含句には、このパラメータが影響する DDL を特定するフィルタリング基準を含めます。
	◆ 除外句には、このパラメータから特定の DDL を除外するフィルタリング基準を含めます。
	包含または除外句は、INCLUDE または EXCLUDE キーワードに続き、適用されるパラメータの他のオプションの有効な組合せで構成される必要があります。
	
	EXCLUDE を使用する場合は、対応する INCLUDE 句を作成する必要があります。たとえば、次は無効です。
	`DDL EXCLUDE OBJNAME "hr.*"`
	一方、次はいずれも使用できます。
	`DDL INCLUDE ALL, EXCLUDE OBJNAME "hr.*"`
	`DDL INCLUDE OBJNAME "fin.*" EXCLUDE OBJNAME "fin.ss"`
	EXCLUDE は、同一の基準を含むすべての INCLUDE よりも優先されます。包含および除外句は、複数使用できます。

表 35　DDL 包含および除外オプション (続き)

オプション	説明
`MAPPED | UNMAPPED | OTHER | ALL`	MAPPED、UNMAPPED、OTHER および ALL では、DDL 操作の範囲に基づいて INCLUDE または EXCLUDE を適用します。 ◆ MAPPED は、MAPPED 範囲の DDL 操作に INCLUDE または EXCLUDE を適用します。MAPPED フィルタリングは、他の DDL パラメータ・オプションで指定されているフィルタリングの前に実行されます。 ◆ UNMAPPED は、UNMAPPED 範囲の DDL 操作に INCLUDE または EXCLUDE を適用します。 ◆ OTHER は、OTHER 範囲の DDL 操作に INCLUDE または EXCLUDE を適用します。 ◆ ALL は、すべての範囲の DDL 操作に INCLUDE または EXCLUDE を適用します。
`OPTYPE <type>`	OPTYPE では、INCLUDE または EXCLUDE を、CREATE、ALTER および RENAME など、特定のタイプの DDL 操作に適用します。<type> には、データベースに有効な任意の DDL コマンドを指定します。たとえば、ALTER 操作を包含するための適切な構文は次のようになります。 `DDL INCLUDE OPTYPE ALTER`

`OBJTYPE '<type>'`	OBJTYPE では、特定のタイプのデータベース・オブジェクトに INCLUDE または EXCLUDE を適用します。<type> には、TABLE、INDEX および TRIGGER など、データベースに有効な任意のオブジェクト・タイプを指定します。Oracle マテリアライズド・ビューおよびマテリアライズド・ビュー・ログの適切なタイプは、それぞれ snapshot、snapshot log です。オブジェクト・タイプの名前は、一重引用符で囲みます。次に例を示します。
	`DDL INCLUDE OBJTYPE 'INDEX'`
	`DDL INCLUDE OBJTYPE 'SNAPSHOT'`
	Oracle オブジェクト・タイプ USER の場合、USER はスキーマのみを持つのに対し、OBJNAME で "owner.object" が予期されるため、OBJNAME オプションは使用しないでください。

`OBJNAME <name>`	OBJNAME では、オブジェクトの完全修飾名 (例 : owner.table_name) に INCLUDE または EXCLUDE を適用します。ワイルドカードは、オブジェクト名にのみ使用できます。
	次に例を示します。
	`DDL INCLUDE OBJNAME accounts.*`
	USER のみがスキーマを持つのに対し、OBJNAME で owner.object が予期されるため、Oracle USER オブジェクトに OBJNAME を使用しないでください。
	Replicat パラメータ・ファイルで MAPPED とともに OBJNAME を使用する場合は、OBJNAME の値は MAP 文の TARGET 句で指定された名前を参照する必要があります。たとえば、次の MAP 文では、正しい値は OBJNAME fin2.* になります。
	`MAP fin.exp_*, TARGET fin2.*;`

表 35　DDL 包含および除外オプション (続き)

オプション	説明

	この例では、CREATE TABLE 文はソース上でこのように実行されます。
	`CREATE TABLE fin.exp_phone;`
	ターゲット上ではこのように実行されます。
	`CREATE TABLE fin2.exp_phone;`
	ターゲットの所有者が MAP 文で指定されていない場合、Replicat は USERID パラメータで指定されているデータベース・ユーザーにターゲットの所有者をマップします。
	
	トリガーなどの導出オブジェクトを作成する DDL の場合、OBJNAME の値は、導出オブジェクトの名前ではなく、ベース・オブジェクトの名前である必要があります。
	たとえば、次の DDL 文を包含するための正しい値は、"hr.insert_trig" でなく、"hr.accounts" です。
	`CREATE TRIGGER hr.insert_trig ON hr.accounts;`
	RENAME 操作の場合は、OBJNAME の値は新しい表名である必要があります。たとえば、次の DDL 文を包含するための正しい値は、"hr.acct" です。
	`ALTER TABLE hr.accounts RENAME TO acct;`

`INSTR '<string>'`	INSTR では、コメント内ではなくコマンド構文内に特定の文字列を含む DDL 文に、INCLUDE または EXCLUDE を適用します。たとえば、次の文では索引を作成する DDL を除外します。 `DDL INCLUDE ALL EXCLUDE INSTR 'CREATE INDEX'` 文字列は一重引用符で囲みます。文字列検索では、大 / 小文字は区別されません。 INSTR では、文字列内の一重引用符 (`' '`) はサポートされません。また、NULL 値もサポートされません。

`INSTRCOMMENTS '<comment_string>'`	(Oraclet に有効)INSTRCOMMENTS では、DDL コマンド内ではなくコメント内に特定の文字列を含む DDL 文に、INCLUDE または EXCLUDE を適用します。INSTRCOMMENTS を使用することで、コメントをフィルタリング・エージェントとして使用できます。
	たとえば、次の文ではコメント内に "source" を含む DDL 文を除外します。
	`DDL INCLUDE ALL EXCLUDE INSTRCOMMENTS 'SOURCE ONLY'`
	この例では、次のような DDL 文はレプリケートされません。
	`CREATE USER john IDENTIFIED BY john /*source only*/;`
	文字列は一重引用符で囲みます。文字列検索では、大 / 小文字は区別されません。INSTR と INSTRCOMMENTS を組み合せると、同一の DDL 文のコマンド構文およびコメントの文字列でフィルタリングできます。
	INSTRCOMMENTS では、文字列内の一重引用符 (`' '`) はサポートされません。また、NULL 値もサポートされません。

表 35　DDL 包含および除外オプション (続き)

オプション	説明

`INSTRWORDS '<word list>'`	INSTRWORDS では、特定の単語を含む DDL 文に INCLUDE または EXCLUDE を適用します。
	<word list> には、一重引用符内に任意の順番で単語を指定します。空白を含めるには、空白 (および適切な場合は単語) を二重引用符で囲みます。二重引用符は、文を囲むためにも使用できます。
	INSTRWORDS を有効にするには、指定したすべての単語が DDL 内に存在している必要があります。
	次に例を示します。
	`ALTER TABLE INCLUDE INSTRWORDS 'ALTER CONSTRAINT " xyz"`
	この例は、次と一致します。
	`ALTER TABLE ADD CONSTRAINT xyz CHECK`
	および
	`ALTER TABLE DROP CONSTRAINT xyz`
	INSTRWORDS では、文字列内の一重引用符 (`' '`) はサポートされません。また、NULL 値もサポートされません。

`INSTRCOMMENTSWORDS '<word list>'`	(Oracle に有効)INSTRWORDS と同様に機能しますが、DDL 構文自体ではなく、DDL 文内のコメントのみに適用します。INSTRCOMMENTS を使用することで、コメントをフィルタリング・エージェントとして使用できます。 INSTRCOMMENTSWORDS では、文字列内の一重引用符 (`' '`) はサポートされません。また、NULL 値もサポートされません。 INSTRWORDS と INSTRCOMMENTSWORDS を組み合せると、同一の DDL 文のコマンド構文およびコメントの文字列でフィルタリングできます。
`STAYMETADATA`	(Oracle に有効です)。メタデータが Extract によって取得されたり、Replicat によって適用されたりしないようにします。 Extract が表で最初に DML を検出すると、その表のメタデータが取得されます。表で DDL が検出されると、古いメタデータは無効化されます。その表の次の DML は新しいメタデータと一致するため、ターゲット表の構造は、ソースの構造とともに常に最新です。 ただし、特定の DDL 操作が表のメタデータに影響しないことがわかっている場合、現在のメタデータが取得または複製されないように、STAYMETADATA を使用できます。これにより、インポートやエクスポートのような操作の際に役立つパフォーマンスの向上が可能になり、切捨てなどの DDL および制約の無効化がよく実行されます。表構造は後続のデータ・レプリケーションの整合性に関連するため、これらの操作は表構造に影響せず、このような場合に無視することができます。たとえば、ALTER TABLE ADD FOREIGN KEY は、表のメタデータに影響しません。

表 35　DDL 包含および除外オプション (続き)

オプション	説明
	これを選択的に適用する方法の例は、次のとおりです。 `DDL INCLUDE ALL INCLUDE STAYMETADATA OBJNAME xyz` この例では、レプリケーション用にすべての DDL が含まれますが、オブジェクト "xyz" を操作する DDL のみ STAYMETADATA の影響を受けるということが記述されています。 STAYMETADATA も、EXCLUDE 句の場合と同じ方法で使用できます。 STAYMETADATA は、ソースとターゲットでメタデータの整合性を保証する場合と同じ方法で使用する必要があります。 STAYMETADATA が使用中の場合、レポート・ファイルにメッセージが追加されます。DDL のレポートは、REPORT オプションとともに使用した DDLOPTIONS パラメータによって制御されます。 次のように @ddl_staymetadata スクリプトを使用して、ソースで検出されるすべての DDL に同じ機能をグローバルに適用できます。 ◆ @ddl_staymetadata_on は、メタデータ・バージョニングをグローバルにオフにします。 ◆ @ddl_staymetadata_off は、メタデータ・バージョニングを再度グローバルに有効にします。 このオプションは、どの DDL がオブジェクト・メタデータに影響するかが明確ではない場合があるため、Oracle GoldenGate テクニカル・サポート・スタッフの支援とともに使用する必要があります。適切に使用しないと、レプリケーション環境の整合性が失われる可能性があります。
`EVENTACTIONS (<action specification>)`	Extract または Replicat プロセスが、トランザクション・ログまたはトレイル内の DDL レコード (*イベント・レコード*と呼ばれる) に基づいて、定義済のアクションを実行するようにします。DDL パラメータのその他のフィルタリング・オプションに応じて、DDL レコードが、Extract またはデータ・ポンプによるトレイルへの書込み、または Replicat による実行が可能である場合、DDL イベントがトリガーされます。このシステムを使用すると、データベース・イベントに基づいて処理をカスタマイズできます。 <action specification> については、MAP および TABLE パラメータの下の EVENTACTIONS を参照してください。 EVENTACTIONS を DDL レコードで使用する場合のガイドラインは、次のとおりです。 ◆ CHECKPOINTBEFORE: 各 DDL レコードは自律的であるため、DDL レコードはトランザクションの開始であることが保証されます。そのため、CHECKPOINT BEFORE イベント・アクションは DDL レコードに対して暗黙的です。 ◆ IGNORE: このオプションは、DDL レコードには無効です。DDL 操作は自律的であるため、レコードを無視することはトランザクション全体を無視することと同等です。 次の DDL オブジェクトは導出オブジェクトであるため、EVENTACTIONS ではサポートされません。 ◆ 索引 ◆ トリガー ◆ シノニム ◆ 表での RENAME および ALTER TABLE RENAME

例 1 次に、DDL パラメータ・オプションを組み合せて使用する例を示します。

```
DDL  &
INCLUDE UNMAPPED &
    OPTYPE alter &
    OBJTYPE 'table' &
    OBJNAME users.tab* &
INCLUDE MAPPED OBJNAME * &
EXCLUDE MAPPED OBJNAME temporary.tab"
```

この文のフィルタリング基準の組合せでは、次のように指定しています。

- TABLEまたはMAP文でマップされていない(UNMAPPED範囲)表に対するすべてのALTER TABLE文を**包含**する
  - 表が "users" によって所有され、名前が "tab" で始まる場合のみ
- およびTABLEまたはMAP文でマップされている(MAPPED範囲)すべての表に対するすべてのDDL操作タイプを**包含**する
- および MAPPED 範囲の、すべての表に対するすべての DDL 操作タイプを**除外**する
  - "temporary" によって所有される場合のみ
  - かつ、名前が "tab" で始まる場合のみ

例 2 次の例では、すべての DDL レコードに対する REPORT のイベント・アクションを指定します。

```
DDL INCLUDE ALL EVENTACTIONS (REPORT)
```

例 3 次に、EVENTACTIONS を DDL のサブセットで使用する方法の例を示します。すべての DDL がレプリケートされますが、明示的に名前が付けられたオブジェクトで実行される DDL のみが、REPORT および LOG のイベント・アクションのトリガー対象となります。

```
DDL INCLUDE ALL &
    INCLUDE OBJNAME sales.t* EVENTACTIONS (REPORT)
    INCLUDE OBJNAME fin.my_tab EVENTACTIONS (LOG)
```

例 4 次に、GLOBALS パラメータの USEANSISQLQUOTES を使用する場合の結果の例を示します。Oracle GoldenGate ファンクションに渡される文字列パラメータは、(デフォルトの二重引用符のかわりに) 一重引用符で囲みます。この例では、DDL ファンクションと文字列ファンクションの両方を使用する複合ファンクションも示しています。

```
DDL INCLUDE OBJNAME src.t* &
EVENTACTIONS (SHELL ("echo shell("echo extract $0 DDL &
optype-objtype-owner.name: $1", &
var $0=@GETENV('GGENVIRONMENT','GROUPNAME'), &
var $1 = @strcat(@ddl(optype), '-', @ddl(objname), '-', &
@ddl(objowner), '-', @ddl(objname))))
```

"alter table t2 modify (col2 date)" に対するこのシェル・コマンドの結果は、次のようになります。

```
"extract E_CUST DDL optype-objtype-owner.name: ALTER-T2-SRC-T2"
```

DDLERROR

適用対象 Extract および Replicat

DDLERROR パラメータでは、ソースおよびターゲット・システムの DDL エラーを処理します。Extract および Replicat 用のオプションが用意されています。

Extract 用 DDLERROR オプション

Extract 用の DDLERROR パラメータ・オプションでは、Extract によって認識される、メタデータが検出不可能なオブジェクトに関するエラーを処理します。

デフォルト 異常終了

構文

```
DDLERROR [RESTARTSKIP <num skips>] [SKIPTRIGGERERROR <num errors>]
```

引数	説明
`RESTARTSKIP <num skips>`	Extract の起動時に、特定の数の DDL 操作をスキップおよび無視させ、エラーによる Extract の異常終了を防ぎます。デフォルトでは、DDL エラーが発生すると Extract は異常終了するため、操作はスキップされません。このパラメータで有効な値は、1 ～ 100000 です。 スキップした操作を Extract レポート・ファイルに書き込むには、DDLOPTIONS パラメータと REPORT オプションを使用します。
`SKIPTRIGGERERROR <num errors>`	(Oracle)Extract の起動時に、DDL トリガーが原因で発生する特定の数の DDL エラーをスキップおよび無視させます。<num errors> の有効な値は、1 ～ 100000 です。 SKIPTRIGGERERROR は、RESTARTSKIP オプションの前にチェックされます。Extract がトリガー・エラーのために DDL 操作をスキップする場合、その操作は RESTARTSKIP 指定の対象としてカウントされません。

Replicat 用 DDLERROR オプション

Replicat 用の DDLERROR パラメータ・オプションでは、DDL がターゲット・データベースに適用されるときに発生するエラーを処理します。DDLERROR オプションを使用することで、大半のエラーをデフォルトの方法 (処理を停止するなど) で処理しながら、他のエラーも特定の方法で処理できます。同一のパラメータ・ファイルで複数の DDLERROR インスタンスを使用して、発生しうるすべてのエラーに対処できます。

デフォルト 異常終了

構文

```
DDLERROR
{<error> | DEFAULT} {<response>}
{INCLUDE <inclusion clause> | EXCLUDE <exclusion clause>}
[IGNOREMISSINGOBJECTS | ABENDONMISSINGOBJECTS]
```

引数	説明
{<error> | DEFAULT} {<response>}	◆ <error> は、この文で処理される特定の DDL エラーです。 ◆ DEFAULT は、明示的な DDLERROR 文が指定されているエラーを除くすべての DDL エラーに対するグローバルなレスポンスを設定します。 ◆ <response> には、次のいずれかを指定できます。 ABEND 操作をロールバックし、処理を異常終了します。ABEND はデフォルトです。 DISCARD 問題のある操作を破棄ファイルに記録しますが、後続の DDL の処理を継続します。DISCARDFILE パラメータを使用して破棄ファイルを指定します。 IGNORE エラーを無視します。 RETRYOP MAXRETRIES <n> [RETRYDELAY <delay>] 問題のある操作を再試行します。MAXRETRIES オプションでは、再試行回数を制御します。指定した MAXRETRIES 回数の後、Replicat は異常終了します。整数を指定します。 RETRYDELAY を使用して、試行間隔 (秒) を設定します。
{INCLUDE <inclusion clause> | EXCLUDE <exclusion clause>}	特定の DDL が DDLERROR によって処理されるかどうかを制御する包含または除外句の始まりを識別します。構文と使用法については、165 ページの「DDL 包含および除外オプション」を参照してください。
[IGNOREMISSINGOBJECTS | ABENDONMISSINGOBJECTS]	ターゲット上に見つからなかったオブジェクトで DML が発行された場合、Extract が異常終了するかどうかを制御します。この状況は、通常、レプリケーション外のターゲットで直接発行された DDL、またはソース定義とターゲット定義間の不一致により発生します。 IGNOREMISSINGOBJECTS は、見つからない表での DML 操作を Replicat がスキップするようにします。 ABENDONMISSINGOBJECTS は、見つからない表での DML 操作で Replicat が異常終了するようにします。

例 1 DDLERROR の基本的な例

次の例では、DDLERROR 文に基づき、Replicat は指定されたエラーを無視しますが、その前に、該当の操作を 10 秒間隔で 3 回試行します。Replicat は、ワイルドカード指定 "tab1*" を満たすオブジェクト上で実行される操作を除き、"tab*" (任意のユーザー、任意の操作) を満たす名前のオブジェクトで実行される DDL 操作にエラー処理を適用します。

```
DDLERROR <error> IGNORE RETRYOP MAXRETRIES 3 RETRYDELAY 10 &
INCLUDE ALL OBJTYPE TABLE OBJNAME "tab*" EXCLUDE OBJNAME tab1*
```

このエラーを除くすべてのエラーを処理するために、次の DDLERROR 文を追加できます。

```
DDLERROR DEFAULT ABENDS
```

この場合、Replicat は DDL エラー発生時に異常終了します。

例 2 複数の DDLERROR 文の使用

パラメータ・ファイルに DDLERROR 文をリストする順番は、各文の有効性に影響しませんが、複数の DDLERROR 文に、追加の修飾子なしで同一のエラーが指定されている場合は例外です。このようなケースでは、Replicat は最初にリストされている文のみを使用します。たとえば、次のような文では、エラーが発生すると Replicat は異常終了します。

```
DDLERROR <error1> ABEND
DDLERROR <error1> IGNORE
```

ただし、適切な修飾子を指定すれば、上記の構成を有効に使用できます。次に例を示します。

```
DDLERROR <error1> ABEND INCLUDE OBJNAME tab*
DDLERROR <error1> IGNORE
```

このケースでは、INCLUDE 文が追加されているため、Replicat は、エラーの DDL 文のオブジェクト名がワイルドカード "tab*" に一致する場合にのみ異常終了します。Replicat は、他のオブジェクト名を含むエラー操作を無視します。

DDLOPTIONS

適用対象 Extract および Replicat

DDLOPTIONS パラメータでは、フィルタリングおよび文字列置換以外の DDL 処理を構成します。DDLOPTIONS 文は複数使用可能ですが、1 つの文のみを使用することをお薦めします。複数の DDLOPTIONS 文を使用する場合は、相互に上書きしないように、それぞれを一意の文にする必要があります。複数の DDLOPTIONS 文を使用する場合は、リストした順に実行されます。

Oracle GoldenGate の DDL サポートの範囲で DDLOPTIONS を使用する方法の詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。

デフォルト 引数の説明を参照してください。

構文

```
DDLOPTIONS
[, ADDTRANDATA [ABEND | RETRYOP <RETRYDELAY <seconds> MAXRETRIES <retries>]
[, DEFAULTUSERPASSWORDPASSWORD <password>
    [ENCRYPTKEY DEFAULT | ENCRYPTKEY <keyname>]]
[, GETAPPLOPS | IGNOREAPPLOPS]
[, GETREPLICATES | IGNOREREPLICATES]
[, IGNOREMAPPING]
[, MAPDERIVED | NOMAPDERIVED]
[, MAPSCHEMAS]
[, MAPSESSIONSCHEMA] <source_schema> TARGET <target_schema>
[, PASSWORD ENCRYPTKEY [DEFAULT | ENCRYPTKEY <keyname>]
[, REMOVECOMMENTS {BEFORE | AFTER}]
[, REPLICATEPASSWORD | NOREPLICATEPASSWORD]
[, REPORT | NOREPORT]
[, UPDATEMETADATA]
[, USEOWNERFORSESSION]
```

引数	説明
`ADDTRANDATA [ABEND | RETRYOP <RETRYDELAY <seconds> MAXRETRIES <retries>]`	Extract(Oracle および Teradata) に有効です。 ADDTRANDATA は次の目的で使用します。 ◆ CREATE TABLE で作成された新しい表に Oracle サプリメンタル・ロギングを自動的に有効化する。 ◆ 列を追加または削除する ALTER TABLE の影響を受けた表のサプリメンタル・ロギングを更新する。 ◆ 名前が変更された表のサプリメンタル・ロギングを更新する。 ◆ 一意キーまたは主キーが追加または削除された表のサプリメンタル・ロギングを更新する。 ABEND | RETRYOP <RETRYDELAY <seconds> MAXRETRIES <retries> 表がロックされているために ADD TRANDATA コマンドが失敗した場合に、Extract が異常終了するか再試行するかを制御します。 ◆ ABEND を指定すると、Extract は異常終了します。 ◆ RETRYOP を指定すると、Extract は RETRYDELAY および MAXRETRIES に基づいてコマンドを再試行します。RETRYOP を使用するときは、次を両方とも指定する必要があります。 ◆ RETRYDELAY では、Extract の再試行までの待機時間を設定します。最大待機時間は 10,000 秒です。 ◆ MAXRETRIES では、Extract が異常終了するまでに実行する再試行回数を設定します。最大再試行回数は 10,000 です。 デフォルトは次のとおりです。 `DDLOPTIONS ADDTRANDATA RETRYOP RETRYDELAY 10 MAXRETRIES 10`

引数	説明
	この機能を使用するには、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』の指示に基づいて、DDL 取得のために Oracle GoldenGate、データベース、および適切な表を構成する必要があります。Oracle の場合は、『Oracle GoldenGate *Oracle インストレーションおよびセットアップ・ガイド*』の指示に基づいて、Oracle GoldenGate DDL オブジェクトがインストールおよび構成されている必要があります。 CREATE TABLE で作成された新しい表には、ADDTRANDATA は Oracle ALTER TABLE コマンドと ADD SUPPLEMENTAL LOG GROUP オプションを発行するため、GGSCI のデフォルトの ADD TRANDATA コマンドと同じ結果になります。Oracle GoldenGate は、ソースで CREATE TABLE または ALTER TABLE が取得されると、このコマンドを実行します。サプリメンタル・ロギングに関して特別の要件がある場合は、DDLOPTIONS ADDTRANDATA ではなく、ADD TRANDATA コマンドを使用してください。 デフォルトでは、サプリメンタル・ロギングを追加する ALTER TABLE は、GETREPLICATES オプションを使用していなければ、ターゲットにレプリケートされません。 名前が変更された表には、ADDTRANDATA は古い名前の表のサプリメンタル・ログ・グループを削除し、新しい表のサプリメンタル・ログ・グループを作成します。ADDTRANDATA を使用していない場合に表の名前を変更するときは、次の手順を実行して、ログ・グループを作成してから名前を変更してください。 ***1.*** データベース・インタフェースまたは GGSCI の DELETE TRANDATA <table> を使用して、サプリメンタル・ログ・グループを削除します。 ***2.*** 表の名前を変更します。 ***3.*** データベース・インタフェースまたは GGSCI の ADD TRANDATA <table> を使用して、新しいサプリメンタル・ログ・グループを作成します。

引数	説明

	新しい DDL 操作の実行から ADD TRANDATA の発効までの間に、ラグがあることがあります。影響を受ける表のデータをレプリケートする場合は、この間にこの表での DML 操作 (INSERT、UPDATE、DELETE) を許可しないでください。許可すると、データは取得されません。ADDTRANDATA の発効後にいつ DML を再開できるかを特定するには、次の操作を実行します。
	1. GGSCI で Extract パラメータ・ファイルを編集します。
	警告: ローカル・オペレーティング・システムの文字セット以外の文字セットで作成されたパラメータ・ファイルを表示または編集するために、VIEW PARAMS または EDIT PARAMS コマンドを使用しないでください。GGSCI 外部からファイルを表示してください。そうしない場合、内容が破損することがあります。
	2. DDLOPTIONS に REPORT オプションを追加し、ファイルを保存して閉じます。
	`DDLOPTIONS [, other DDLOPTIONS options], REPORT`
	3. Extract を停止および起動してパラメータへの変更を有効化します。
	`STOP EXTRACT <group>`
	`START EXTRACT <group>`
	4. Extract プロセス・レポートを表示します。
	`VIEW REPORT <group name>`
	5. 表にログ・グループを追加した ALTER TABLE を探し、このコマンドが発効になった時刻をメモします。次のようなエントリです。
	`Successfully added TRAN DATA for table with the key, table [QATEST1.MYTABLE], operation [ALTER TABLE "QATEST1"."MYTABLE" ADD SUPPLEMENTAL LOG GROUP "GGS_MYTABLE_53475" (MYID) ALWAYS /* GOLDENGATE_DDL_REPLICATION */ ].`
	6. 新しい表での DML 操作を許可します。
	84 ページの「ADD SCHEMATRANDATA」も参照してください。

引数	説明
`DEFAULTUSERPASSWORD <password> [<algorithm> ENCRYPTKEY {<keyname> | DEFAULT}]`	Replicat に有効です。(Oracle のみ)。 レプリケートされる {CREATE | ALTER} USER <name> IDENTIFIED BY <password> 文に、ソース文で使用されているパスワードとは異なるパスワードを指定します。Replicat は、Extract がトレイルに書き込むプレースホルダを、指定されたパスワードに置き換えます。DEFAULTUSERPASSWORD を使用する場合、Extract には DDLOPTIONS の NOREPLICATEPASSWORD オプションを使用します。 オプションなしの DEFAULTUSERPASSWORD <password> では、クリアテキストのパスワードを指定します。パスワードに大文字と小文字の区別がある場合は、そのとおりに入力します。 GGSCI の ENCRYPT PASSWORD コマンドでパスワードが暗号化されている場合、次のオプションを使用します。 ◆ <algorithm> は、ENCRYPT PASSWORD でパスワードの暗号化に使用した暗号化アルゴリズムを、AES128、AES192、AES256 または BLOWFISH の中から指定します。 ◆ ENCRYPTKEY <keyname> は、ENCKEYS 参照ファイル内のユーザー作成の暗号化鍵の論理名を指定します。ENCRYPT PASSWORD が KEYNAME <keyname> オプションとともに使用された場合に使用し、同じ鍵名を指定します。 ◆ ENCRYPTKEY DEFAULT は、Oracle GoldenGate に、ランダム鍵を使用するように指示します。ENCRYPT PASSWORD が KEYNAME DEFAULT オプションとともに使用された場合に使用します。 Oracle GoldenGate の暗号化オプションの詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。
`GETAPPLOPS | IGNOREAPPLOPS`	Extract に有効です。(Oracle のみ)。 Replicat *以外*のビジネス・アプリケーションが生成した DDL 操作を、Extract がトレイルまたはファイルに書き込むコンテンツに含めるかどうかを制御します。GETAPPLOPS および IGNOREAPPLOPS は、GETREPLICATES および IGNOREREPLICATES オプションとともに使用して、双方向またはカスケード構成で伝播する DDL を制御できます。 ◆ 双方向構成では、GETAPPLOPS と GETREPLICATES を使用します。UPDATEMETADATA オプションも使用する必要があります。 ◆ カスケード構成では、DDL 操作をターゲットにカスケードするシステム上で、IGNOREAPPLOPS と GETREPLICATES を使用します。 デフォルトは GETAPPLOPS です。
`GETREPLICATES | IGNOREREPLICATES`	Extract に有効です (Oracle のみ)。Replicat が生成した DDL 操作を Extract がトレイルまたはファイルに書き込むコンテンツに含めるかどうかを制御します。デフォルトは IGNOREREPLICATES です。詳細は、DDLOPTIONS の GETAPPLOPS | IGNOREAPPLOPS オプションを参照してください。

引数	説明
IGNOREMAPPING	Replicat に有効です。DDL が MAPPED 範囲か UNMAPPED 範囲かを決定する名前マッピングの評価を無効化します。このオプションにより、ソースとターゲットのスキーマ名およびオブジェクト名が一致するためマッピング機能が不要な場合に、like-to-like DDL レプリケーション構成のパフォーマンスが向上します。IGNOREMAPPING を有効にすると、MAPPED または UNMAPPED 範囲を決定できず、すべての DDL 文は OTHER 範囲として処理されます。ソースのスキーマおよびオブジェクト名がターゲット上で異なるスキーマおよびオブジェクト名にマッピングされている場合、このパラメータを使用しないでください。
MAPDERIVED | NOMAPDERIVED	Replicat(Oracle および Teradata) に有効です。導出オブジェクト名のマッピング方法を制御します。 ◆ MAPDERIVED: 導出オブジェクトの MAP 文が存在する場合、名前はその文の TARGET 句に指定されている名前にマッピングされます。それ以外の場合、名前はベース・オブジェクトを含む MAP 文の TARGET 句に指定されている名前にマッピングされます。MAPDERIVED はデフォルトです。 ◆ NOMAPDERIVED: 名前のマッピングを行わないようにします。NOMAPDERIVED は、導出オブジェクトの名前を含むすべての明示的な MAP 文より優先されます。 DDL レプリケーション中の導出オブジェクト処理方法の詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。

引数	説明

MAPSCHEMAS	Replicat(Oracle および Teradata) に有効です。MAPSESSIONSCHEMA を使用している場合のみ使用します。
	◆ MAPSESSIONSCHEMA は、セッション・スキーマのソース・ターゲット間のマッピングを確立し、スキーマが DDL で修飾されていないオブジェクトに使用されます。
	◆ MAPSCHEMAS は、修飾されたスキーマがソース DDL に存在するが、MAP を使用したマッピングは修飾しないオブジェクトを、MAPSESSIONSCHEMA と同じセッション・スキーマ間のマッピングにマップします。そのようなオブジェクトの例には、AS SELECT 句内の導出オブジェクトが含まれている Oracle の CREATE TABLE AS SELECT 文や、Teradata の CREATE REPLICATION RULESET 文があります。
	このマッピングは、MAP 文で指定されたマッピング後に行われます。
	例として、次の DDL 文がソース Oracle データベースで発行されるとします。
	`create table a.t as select from b.t;`
	ターゲット上での MAP 文は次のようになるとします。
	`MAP a.*, TARGET c.*;`
	`DDLOPTIONS MAPSESSIONSCHEMA b, TARGET b1, MAPSCHEMAS`
	このマッピングの結果、Replicat はターゲット上に次の DDL 文を発行します。
	`create table c.t as select from b1.t;`
	◆ ベース表は、TARGET 句 (スキーマ "c") に従ってマップされます。
	◆ 修飾された導出オブジェクト ("SELECT FROM" 内の表 "t") は、MAPSCHEMAS が存在するため、MAPSESSIONSCHEMA (スキーマ "b1") に従ってマップされます。
	MAPSCHEMAS を指定しない場合、MAPSESSIONSCHEMA のみが未修飾のオブジェクトをマップするため、導出オブジェクトは、(TARGET 句で指定した) スキーマ "c" にマップされます。

MAPSESSIONSCHEMA <source_schema> TARGET <target_schema>	Replicat に有効です (Oracle のみ)。ソースのセッション・スキーマを、ターゲット上の異なるセッション・スキーマにマッピング (変換) します。 ◆ <source_schema> は、ソース上で ALTER SESSION set CURRENT_SCHEMA で設定されているセッション・スキーマです。 ◆ <target_schema> は、ターゲット上で ALTER SESSION set CURRENT_SCHEMA で設定されているセッション・スキーマです。 ワイルドカードはサポートされていません。複数の MAPSESSIONSCHEMA パラメータを使用して、異なるスキーマをマップできます。 MAPSESSIONSCHEMA は、マスターまたは導出オブジェクト名に基づくすべてのスキーマ名マッピングより優先されます。 このセクション最後の使用例を参照してください。 「MAPSCHEMAS」も参照してください。

引数	説明
PASSWORD <algorithm> ENCRYPTKEY {<keyname> | DEFAULT}	Extract に有効です (Oracle のみ)。 DDL をトレイルに書き込む前に、Extract にソース DDL のすべてのパスワードを暗号化させます。 ◆ <algorithm> は、パスワードの暗号化に使用する暗号化アルゴリズムを、AES128、AES192、AES256 または BLOWFISH の中から指定します。 ◆ ENCRYPTKEY <keyname> は、ENCKEYS 参照ファイル内のユーザー作成の暗号化鍵の論理名を指定します。 ◆ ENCRYPTKEY DEFAULT は、Oracle GoldenGate に、ランダム鍵を使用するように指示します。
REMOVECOMMENTS {BEFORE | AFTER}	Extract および Replicat に有効です (Oracle のみ)。DDL 操作からコメントを削除するかどうかを制御します。デフォルトでは、DDLSUBST パラメータを使用して文字列の置換で使用できるように、コメントは削除されません (182 ページを参照してください)。 ◆ REMOVECOMMENTS BEFORE では、Extract または Replicat によって DDL 操作が処理される前にコメントを削除します。コメントは、文字列置換で使用できません。 ◆ REMOVECOMMENTS AFTER では、文字列置換で使用された後にコメントを削除します。
REPLICATEPASSWORD | NOREPLICATEPASSWORD	Extract に有効です (Oracle のみ)。{CREATE | ALTER} USER <user> IDENTIFIED BY <password> コマンドのパスワードに適用します。 ◆ デフォルト (REPLICATEPASSWORD) では、Oracle GoldenGate はターゲットの CREATE または ALTER 文でソース・パスワードを使用します。 ◆ ソース・パスワードのターゲットへの送信を防ぐには、NOREPLICATEPASSWORD を使用します。 NOREPLICATEPASSWORD を使用する場合は、Replicat パラメータ・ファイルの DDLOPTIONS 文と DEFAULTUSERPASSWORD オプションを使用して、ターゲット DDL 文用のパスワードを指定します。
REPORT | NOREPORT	Extract および Replicat(Oracle および Teradata) に有効です。レポート・ファイルに詳しい DDL 処理情報を書き込むかどうかを制御します。デフォルトの NOREPORT では、基本的な DDL 統計をレポートします。REPORT では、使用しているパラメータ、および処理された操作のステップバイステップの履歴をレポートに追加します。
UPDATEMETADATA	Replicat に有効です (Oracle のみ)。アクティブアクティブの双方向構成で使用します。このパラメータは、DDL が開始されたシステム上の Replicat に、もう一方のシステムにこの DDL が伝播されたため、新しいメタデータに合わせてすぐにオブジェクト・メタデータ・キャッシュを更新するように通知します。これにより、Replicat のメタデータ・キャッシュと、ローカル・データベースの現在のメタデータの同期性を維持します。

引数	説明
USEOWNERFORSESSION	Replicat に有効です (Oracle のみ)。Replicat DDL 文の未修飾のオブジェクトのスキーマを、ALTER SESSION SET CURRENT_SCHEMA 文のスキーマ (デフォルト動作) ではなく、Replicat のセッション・スキーマのスキーマにします。

例 1

次に、MAPSESSIONSCHEMA を使用して、ソース・セッション・スキーマをターゲット上の別のスキーマにマッピングする方法を示します。

Extract および Replicat で、次のように DDL 取得およびマッピングが構成されているとします。

Extract

```
DDL INCLUDE OBJNAME SRC.* &
INCLUDE OBJNAME SRC1.*
TABLE SRC.*;
TABLE SRC1.*;
```

Replicat

```
DDLOPTIONS MAPSESSIONSCHEMA SRC TARGET DST
DDLOPTIONS MAPSESSIONSCHEMA SRC1 TARGET DST1
MAP SRC.*, TARGET DST.*;
MAP SRC1.*, TARGET DST1.*;
DDL INCLUDE OBJNAME DST.* &
INCLUDE OBJNAME DST1.*
```

ソースのログイン・ユーザー OTH によって、次の DDL 文が発行されるとします。

```
ALTER SESION SET CURRENT_SCHEMA=SRC;
CREATE TABLE tab (X NUMBER);
CREATE TABLE SRC1.tab (X NUMBER) AS SELECT * FROM tab;
```

Replicat は、次のような DDL を実行します (各コード・セグメントの前に説明を記載しています)。

```
-- Set session to DST, because SRC.* is mapped to DST.* in MAP statement.
ALTER SESION SET CURRENT_SCHEMA=DST;
-- Create the first TAB table in the DST schema, using the DST session schema.
CREATE TABLE DST.tab (X NUMBER);
-- Restore Replicat schema.
ALTER SESSION SET CURRENT_SCHEMA=REPUSER
-- Set session schema to DST, per MAPSESSIONSCHEMA, so that AS SELECT succeeds.
ALTER SESION SET CURRENT_SCHEMA=DST;
-- Create the DST1.TAB table AS SELECT * FROM the first table (DST.TAB).
CREATE TABLE DST1.tab (X NUMBER) AS SELECT * FROM tab;
-- Restore Replicat schema.
ALTER SESSION SET CURRENT_SCHEMA=REPUSER
```

MAPSESSIONSCHEMA を指定しない場合、SELECT * FROM TAB は存在しない SRC.TAB 表からの選択を試み、失敗します。デフォルトでは、ソース・スキーマを、ターゲット DDL 文の未修飾のオブジェクトに適用します。その場合、DDL 文は次のようになり、失敗します。

```
-- Set session to DST, because SRC.* is mapped to DST.* in MAP statement.
ALTER SESION SET CURRENT_SCHEMA=DST;
-- Create the first TAB table in the DST schema, using the DST session schema.
CREATE TABLE DST.tab (X NUMBER);
-- Restore Replicat schema.
ALTER SESSION SET CURRENT_SCHEMA=REPUSER
-- Set session schema to SRC, because TAB in the AS SELECT is unqualified and SRC is the
source session schema.
ALTER SESION SET CURRENT_SCHEMA=SRC;
-- Create DST1.TAB AS SELECT * from SRC.TAB (SRC=current session schema).
CREATE TABLE DST1.tab (X NUMBER) AS SELECT * FROM tab;
-- SRC.TAB does not exist.
-- Abend with an error unless the error is handled by a DDLERROR statement.
```

例 2 次に、DEFAULTUSERPASSWORD を使用して、レプリケートされる {CREATE | ALTER} USER <name> IDENTIFIED BY <password> 文にソース文と異なるパスワードを割り当てる様々な方法を示します。

```
DDLOPTIONS DEFAULTUSERPASSWORD ocean

DDLOPTIONS DEFAULTUSERPASSWORD &
AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC, &
AES 256 ENCRYPTKEY mykey

DDLOPTIONS DEFAULTUSERPASSWORD &
AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC, &
BLOWFISH ENCRYPTKEY DEFAULT
```

DDLSUBST

適用対象 Extract および Replicat

DDLSUBST パラメータでは、DDL 操作内の文字列を置換します。たとえば、ある表の名前を別の名前に置換したり、コメント内の文字列を置換したりできます。検索では、大 / 小文字は区別されません。文字列の中で引用符を表すには、二重引用符を使用します。

DDLSUBST 使用のガイドライン

- DDLSUBST は、列名およびデータ型をターゲット上の別の列名やデータ型に変換する目的では使用しないでください。この方法でターゲット・オブジェクトの構造を変更すると、データがターゲットにレプリケートされたときにエラーが発生します。同様に、DDLSUBST は、ターゲット DDL 文の所有者および表名を変更する目的では使用しないでください。レプリケートされる DDL 操作を異なるターゲット・オブジェクトにマップするときは、常に MAP 文を使用してください。

- DDLSUBST は、パラメータ・ファイル内における相対順位にかかわらず、常に DDL パラメータよりも後に実行されます。フィルタリングが先に実行されるため、文字列置換で使用する基準に対応するフィルタリング基準を使用してください。次のパラメータ文を例に取り上げます。

  ```
  DDL INCLUDE OBJNAME fin.*
  DDLSUBST 'cust' WITH 'customers' INCLUDE OBJNAME sales.*
  ```

  この例では、INCLUDE 文と DDLSUBST 文のオブジェクトが異なるため、文字列置換は発生しません。fin に所有されているオブジェクトは Oracle GoldenGate DDL 構成に含まれていますが、sales に所有されているオブジェクトは含まれていません。
- DDLSUBST パラメータは、複数使用できます。これらは、パラメータ・ファイルにリストされた順に実行されます。
- コメントを含む Oracle DDL に対して、コメントの文字列置換を実行する場合、DDLOPTIONS パラメータとともに REMOVECOMMENTS BEFORE オプションを使用しないでください。REMOVECOMMENTS BEFORE を使用すると、文字列置換が発生する前にコメントを削除します。コメントを削除し、かつ文字列置換を行う場合は、REMOVECOMMENTS AFTER オプションを使用してください。
- データベースによって課される制限を除き、置換する文字列に最大サイズの制限はありません。文字列のサイズがデータベースの制限を越えると、この操作を実行している Extract または Replicat プロセスは異常終了します。

デフォルト 置換なし

構文

```
DDLSUBST '<search_string>' WITH '<replace_string>'
[INCLUDE <inclusion clause> | EXCLUDE <exclusion clause>]
```

引数	説明
`'<search_string>'`	ソース DDL 文に含まれる置換対象の文字列。文字列は一重引用符で囲みます。文字列で引用符を表すには、二重引用符を使用します。
`WITH`	必須のキーワード。
`'<replace_string>'`	ターゲット DDL で置換文字列として使用する文字列。文字列は一重引用符で囲みます。文字列で引用符を表すには、二重引用符を使用します。
`INCLUDE <inclusion clause> | EXCLUDE <exclusion clause>`	1 つまたは複数の INCLUDE および EXCLUDE 文を指定して、文字列置換ルールを適用する DDL 操作をフィルタリングします。構文と使用法については、165 ページの「DDL 包含および除外オプション」を参照してください。

例 1 次の例では、"fin" スキーマ内の表の文字列 'cust' を文字列 'customers' に置換します。

```
DDLSUBST 'cust' WITH 'customers'
INCLUDE ALL OBJTYPE 'table' OBJNAME fin.*
```

例 次の例では、DDL コマンドに単語 "logfile" が含まれている場合にのみ新しいディレクトリに置換します。検索文字列が複数回検出された場合、置換文字列は複数回挿入されます。

```
DDLSUBST '/file1/location1' WITH '/file2/location2' INCLUDE INSTR'logfile'
```

例 2 次の例では、複数の DDLSUBST 文を使用します。文はリストされている順番に実行されます。

```
DDLSUBST 'a' WITH 'b' INCLUDE ALL
DDLSUBST 'b' WITH 'c' INCLUDE ALL
```

上記の最終結果では、すべての "a" および "b" 文字列が "c" に置換されます。

DDLTABLE

適用対象 GLOBALS

DDLTABLE パラメータでは、Oracle DDL 同期をサポートする DDL 履歴表の名前をデフォルトの GGS_DDL_HIST 以外にする場合に、その名前を指定します。DDL 履歴表には、Oracle GoldenGate に処理された DDL 操作の履歴が保持されます。

履歴表の名前は、params.sql スクリプトの ddl_hist_table パラメータにも指定する必要があります。このスクリプトは、ルート Oracle GoldenGate インストール・ディレクトリにあります。

このパラメータは、Oracle にのみ有効です。DDL 履歴表および params.sql の詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。

デフォルト GGS_DDL_HIST

構文 `DDLTABLE <table_name>`

引数	説明
`<table_name>`	DDL 履歴表の名前。

例 `DDLTABLE GG_DDL_HISTORY`

DECRYPTTRAIL

適用対象 Extract および Replicat

DECRYPTTRAIL パラメータでは、関連付けられた Extract プロセスがトレイルまたは抽出ファイルを暗号化するために ENCRYPTTRAIL が使用された場合に、トレイルまたは抽出ファイルのデータを復号化します。

デフォルト なし

構文 `DECRYPTTRAIL [{AES128 | AES192 | AES256} KEYNAME <keyname>]`

引数	説明
`DECRYPTTRAIL`	トレイルまたはファイルが ENCRYPTTRAIL でオプション (256 鍵バイト置換) なしで暗号化された場合、DECRYPTTRAIL をオプションなしで使用します。

引数	説明
{AES128 | AES192 | AES256} KEYNAME <keyname>	AES 暗号が ENCRYPTTRAIL で使用された場合、一致する暗号を指定します。keyname には、ENCRYPTTRAIL 文で使用されている論理鍵名を指定します。この鍵は、ローカルの ENCKEYS 参照ファイルに存在する必要があります。トレイルまたはファイルの暗号化の使用の詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。

例 DECRYPTTRAIL AES192, KEYNAME mykey1

DEFERAPPLYINTERVAL

適用対象 Replicat

DEFERAPPLYINTERVAL パラメータでは、Replicat が取得したトランザクションをターゲット・データベースに適用するまでの待機時間を設定します。Replicat は、トランザクションをいつ適用するかを決定するために、ソース・システムのローカル GMT 時間で記録されているソース・ソーストランザクションのコミット・タイムスタンプにこの遅延値を追加します。

DEFERAPPLYINTERVAL は、ソース・データに対して行われた誤った変更の伝播の防止や、異なるタイムゾーン間でのデータ到着の制御、またターゲットにデータを適用する前に他の計画イベントを実行する時間を確保するなどの目的に使用できます。DEFERAPPLYINTERVAL を使用すると、ターゲット・データに意図的に遅延を発生させることになるため、ターゲット・アプリケーションで時間が重視される場合には、慎重に使用する必要があります。

Replicat が操作を延期しているかどうかを判断するには、SEND REPLICAT コマンドと STATUS オプションを使用して、Waiting on deferred apply のステータスを確認します。

注意 **TCPSOURCETIMER パラメータを使用している場合は、ソース・トランザクションとターゲット・トランザクションのタイムスタンプが数秒異なり、Replicat がトランザクションを数秒間オープンにしておく (したがって行ロックしている) 可能性があります。この小さな差異によって、パフォーマンスに大きな影響がおよぶことはないはずです。**

デフォルト 0(遅延なし)

構文 DEFERAPPLYINTERVAL <n><unit>

引数	説明
<n>	遅延時間を示す数値。最小遅延時間は、EOFDELAY パラメータで設定されている値です。最大日数は 7 日間です。
<unit>	遅延時間の単位。次の単位を使用できます。 S | SEC | SECS | SECOND | SECONDS | MIN | MINS | MINUTE | MINUTES | HOUR | HOURS | DAY | DAYS

例 この例では、Replicat にトランザクションの適用まで 10 時間待機させます。

```
DEFERAPPLYINTERVAL 10
```

トランザクションがソースの GTM 時間 08:00:00 に完了している場合、このトランザクションは同日のターゲットの GMT 時間 18:00:00 に適用されます。

DEFSFILE

適用対象 DEFGEN

DEFSFILE パラメータでは、DEFGEN がデータ定義を書き込むファイルの名前を指定します。デフォルトでは、データ定義ファイルは、ローカル・オペレーティング・システムの文字セットで書き込まれます。文字セットを変更するには、CHARSET オプションを使用します。

デフォルト なし

構文 `DEFSFILE <filename> [APPEND | PURGE] [CHARSET <character set>]`

引数	説明
`<filename>`	相対ファイル名または完全修飾ファイル名。このファイルは DEFGEN を実行するときに作成されます。
`APPEND`	指定したファイルがすでに存在している場合、DEFGEN に、既存のコンテンツの末尾に (現在の実行から) 新しいコンテンツを書き込ませます。APPEND が使用され、定義ファイルがすでに存在している場合、DEFGEN は次の方法でデータ定義を追加します。 ◆ 既存のデータ定義ファイルが Oracle GoldenGate 11.2.1 より古いフォーマットの場合、DEFGEN は、マルチバイトおよび特殊文字を含む表名および列名がサポートされていない古いフォーマットで、表定義を追加します。 ◆ 既存のデータ定義ファイルがリリース 11.2.1 で導入されたフォーマットより新しい場合、DEFGEN は、ファイルの既存の文字セットで表定義を追加します。
`PURGE`	現在の実行で新しいコンテンツを書き込む前に、DEFGEN に指定したファイルをパージさせます。これはデフォルトです。
`CHARSET <character set>`	定義ファイルを指定した文字セットで生成します。CHARSET を指定しない場合、オペレーティング・システムのデフォルトの文字セットが使用されます。リリース 11.2.1 以降の定義ファイルに APPEND モードが指定されている場合、CHARSET は無視され、既存の定義ファイルの文字セットが使用されます。

例
```
DEFSFILE ./dirdef/orcldef CHARSET ISO-8859-11
```

DISCARDFILE

適用対象 Extract および Replicat

DISCARDFILE パラメータでは、Oracle GoldenGate が処理できないレコードを記録できる破棄ファイルを生成します。レコードは、複数の理由で破棄されることがあります。たとえば、レコードがトレイルに書き込まれた後に、基盤の表構造が変更されると、レコードは破棄されます。破棄ファイルは、処理エラーの原因の特定にも使用できます。

破棄ファイルの各エントリには、破棄レコード・バッファと、理由を示すエラー・コードが含まれます。Oracle GoldenGate は、指定された破棄ファイルを Oracle GoldenGate インストール・ディレクトリの dirrpt サブディレクトリに作成します。破棄ファイルは、テキスト・エディタか、GGSCI で次のコマンドを使用して表示できます。

```
VIEW REPORT <file name>
```

条件： <file name> は、破棄ファイルの名前です。

破棄ファイルの手動でのメンテナンスを不要にするには、PURGE または APPEND オプションを使用します。Oracle GoldenGate は既存の破棄ファイルに書込みを行わずに終了するため、これらのオプションを使用しない場合は、各プロセスの実行前に異なる破棄ファイルを指定する必要があります。

ファイル・サイズの上限を設定するには、MAXBYTES または MEGABYTES オプションを使用します。指定したサイズを超えると、プロセスは異常終了します。

デフォルト デフォルトでは、Oracle GoldenGate は破棄ファイルを生成しません。

構文

```
DISCARDFILE <file name>
[, APPEND | PURGE]
[, MAXBYTES <n> | MEGABYTES <n>]
```

引数	説明
`<file name>`	破棄ファイルの相対名または完全修飾名。Oracle GoldenGate は Oracle GoldenGate インストール・ディレクトリで名前を修飾するため、ファイルが Oracle GoldenGate のディレクトリにある場合、相対パス名は問題ありません。可能な場合は常に、プロセス・グループと同じ名前を使用してください。
`APPEND`	ファイルがすでに存在する場合に、既存のコンテンツに新しいコンテンツを追加します。
`PURGE`	新しいコンテンツを書き込む前にファイルをパージします。
`MAXBYTES <n>`	ファイルの最大サイズを設定します (バイト)。有効な値は 1 ～ 2147483646 です。デフォルトは 1000000 です。
`MEGABYTES <n>`	ファイルの最大サイズを設定します (MB)。有効な値は 1 ～ 2147 です。デフォルトは 1MB です。

例

```
DISCARDFILE discard.dsc, PURGE, MEGABYTES 2
```

DISCARDROLLOVER

適用対象 Extract および Replicat

DISCARDROLLOVER パラメータでは、破棄ファイルのエージングのスケジュールを設定します。エージング・スケジュールを設定することで、長時間または連続的な実行の場合に、破棄ファイルのサイズが上限に達してプロセスが異常終了することを防止できます。また、アーカイブ・ルーチンに追加されるアーカイブを予測できます。

DISCARDROLLOVER のエージング・ポイントに到達すると、新しい破棄ファイルが作成され、古いファイルは次に説明する <group name><n>.<extension> フォーマットの名前に変更されます。

- <group name> は、Extract または Replicat グループ名です。
- <n> は、新しいファイル作成のたびに 1 ずつ増分される数です。たとえば、myext0.dsc、myext1.dsc、myext2.dsc などのようになります。

時刻、曜日、またはその両方を指定できます。時刻 (AT オプション) のみで曜日 (ON オプション) を指定しない場合、指定した時刻に毎日破棄ファイルが生成されます。

デフォルト 無効。ルールは指定されていません。

構文

```
DISCARDROLLOVER
{AT <hh:mi> |
ON <day of week> |
AT <hh:mi> ON <day of week>}
```

引数	説明
`AT <hh:mi>`	ファイルをエージングする時刻 (24 時間表記)。 有効な値 : ◆ <hh> は、1 ～ 23 までの時間です。 ◆ <mi> は 00 ～ 59 までの分です。

`ON <day of week>`	ファイルをエージングする曜日。
	有効な値 :
	`SUNDAY`
	`MONDAY`
	`TUESDAY`
	`WEDNESDAY`
	`THURSDAY`
	`FRIDAY`
	`SATURDAY`
	大 / 小文字は区別されません。

例 1 `DISCARDROLLOVER AT 05:30`

例 2 `DISCARDROLLOVER ON friday`

例 3 `DISCARDROLLOVER AT 05:30 ON friday`

DOWNCRITICAL

適用対象 Manager

DOWNCRITICAL パラメータでは、DOWNREPORT パラメータによって生成されるレポートに、異常終了または正常終了したプロセスを含めます。

デフォルト なし

構文 `DOWNCRITICAL`

DOWNREPORT

適用対象 Manager

DOWNREPORTMINUTES または DOWNREPORTHOURS パラメータでは、Manager が実行中でない Extract および Replicat プロセスをレポートする間隔を指定します。エラー・ログには、プロセスが開始または終了するたびにイベントが生成されますが、ログが大きい場合には、こうしたメッセージは見逃されてしまいがちです。DOWNREPORTMINUTES および DOWNREPORTHOURS は、停止したプロセスが見逃されないように、定期的にレポートします。

実行中のプロセスをレポートするには、UPREPORT パラメータを使用します。

デフォルト 停止したプロセスをレポートしない。

構文 DOWNREPORTMINUTES <minutes> | DOWNREPORTHOURS <hours>

引数	説明
<minutes>	実行中でないプロセスをレポートする間隔 (分)。
<hours>	実行中でないプロセスをレポートする間隔 (時間)。

例 次の例では、30 分間隔でレポートが生成されます。

```
DOWNREPORTMINUTES 30
```

DSOPTIONS

適用対象 Extract

DSOPTIONS パラメータでは、Teradata Access Module (TAM) を使用する Extract の抽出処理オプションを指定します。Oracle GoldenGate の Teradata 抽出の構成の詳細は、『Oracle GoldenGate *Teradata インストレーションおよびセットアップ・ガイド*』を参照してください。

デフォルト なし

構文

```
DSOPTIONS
[COMMITTEDTRANLOG]
[CREATETRANLOG]
[SORTTRANLOG]
[RESTARTAPPEND]
```

引数	説明
COMMITTEDTRANLOG	(最大パフォーマンス・モード) トランザクション・データをディスクに永続化しないように指定します。TAM が Extract プロセスにトランザクション・データの変更を送信すると、Extract プロセスはトランザクションのコミット順に標準の Oracle GoldenGate トレイルにこれらのデータを保存します。トレイルは、Replicat またはデータ・ポンプ Extract グループが読み取ることができます。
CREATETRANLOG	(最大保護モード)Extract に VAM トレイル (ローカル・トレイル) を作成させます。トランザクション・データの変更は、VAM ソート Extract による次の処理のために、VAM トレイルに永続化されます。データは、様々なトランザクションからの変更レコードをインタリーブするログスタイル・フォーマットで VAM トレイルに書き込まれます。このオプションは、Teradata Access Module(TAM) とインタフェースを取り、VAM トレイルへの書込みを行うプライマリ Extract プロセスに対して使用します。VAM トレイルは、GGSCI の ADD EXTTRAIL コマンドで指定します。

引数	説明
SORTTRANLOG	(最大保護モード)Extract にトランザクションのソート・ファンクションを実行させます。このオプションは、プライマリ Extract プロセスによって移入された VAM トレイルを読み取る VAM ソート Extract グループに対して使用します。VAM ソート Extract は、Oracle GoldenGate による次の処理の前に、インタリーブされた操作を正しい準備 / コミット / ロールバック・トランザクション単位にソートします。
RESTARTAPPEND	(最大パフォーマンス・モード) Extract の再起動時に、前回の実行で書き込んだデータを書き換えるかわりに、Oracle GoldenGate トレイルの末尾にデータを追加させます。このオプションは、COMMITTEDTRANLOG 引数とともに使用します。

DYNAMICPORTLIST

適用対象 Manager

DYNAMICPORTLIST パラメータでは、次のローカル Oracle GoldenGate プロセスがリモート Oracle GoldenGate プロセスとの通信のためにバインドできる、使用可能なポートのリストを指定します。

- Collector: 受信データを受け取るためにリモート Extract と通信します。
- Replicat: 初期ロード・タスク中のデータを受け取るためにリモート Extract と通信します。
- パッシブ Extract: リモート Collector と通信します。
- GGSCI: リモート・コマンドを発行します。

プロセスをリストに追加するための Manager の停止および再起動を伴わないプロセス数の増加に対応するように、十分なポートを指定します。個別のポート、ポートの範囲、またはその両方を指定できます。

デフォルト なし

構文

```
DYNAMICPORTLIST {<port> | <port>-<port>} [ , ...]
```

引数	説明

<port>	割当て可能なポート番号。ポート・エントリの最大数は 5000 です。
	◆ 複数のポートを指定するには、コンマ区切りリストを使用します。次に例を示します。
	`7830, 7833`
	◆ ポート範囲を指定するには、ダッシュ (-) を使用して範囲の最初のポートと最後のポートを区切ります。ダッシュの前後に空白を入力しないでください。次に例を示します。
	`7830-7835`
	◆ ポート範囲および個別ポートを指定するには、範囲と個別ポート番号の間にコンマを置きます。次に例を示します。
	`7830-7835, 7839`

例

```
DYNAMICPORTLIST 7820-7830, 7833, 7835
```

DYNAMICRESOLUTION | NODYNAMICRESOLUTION

適用対象 Extract および Replicat

DYNAMICRESOLUTION および NODYNAMICRESOLUTION パラメータでは、表名の解決方法を制御します。

デフォルトの DYNAMICRESOLUTION では、TABLE または MAP 文で指定された表が多数ある場合に、高速なプロセス起動を可能にします。処理する必要があるトランザクション・レコードのメタデータを取得するために、Oracle GoldenGate はデータベースに問い合せて、関連する表のレコードを構築します。DYNAMICRESOLUTION を指定すると、一度にすべての表でなく、一度に 1 つの表のレコードが構築されます。Extract がトランザクション・ログ内でオブジェクト ID を最初に検出したときに、特定の表のメタデータが追加されるのに対して、他の表のレコード構築は、オブジェクト ID が検出されるまで延期されます。DYNAMICRESOLUTION は、WILDCARDRESOLVE DYNAMIC と同様に動作します。

NODYNAMICRESOLUTION を指定すると、起動時に (すべての表の) オブジェクト・レコード全体が構築され、これはデータベースが大規模な場合、多くの時間がかかることがあります。このオプションは、Teradata ではサポートされていません。NODYNAMICRESOLUTION は、WILDCARDRESOLVE IMMEDIATE と同様に動作します。

WILDCARDRESOLVE の詳細は、424 ページを参照してください。

デフォルト DYNAMICRESOLUTION

構文

```
DYNAMICRESOLUTION | NODYNAMICRESOLUTION
```

DYNSQL | NODYNSQL

適用対象 Replicat

DYNSQL および NODYNSQL パラメータでは、SQL 文の生成方法を制御します。NODYNSQL を指定すると、Replicat はリテラル SQL 文と解決されたバインド変数を使用します。デフォルトの DYNSQL を使用すると、Replicat は動的 SQL を使用して 1 つの文を一度だけコンパイルした後、異なるバインド変数を使用してこの文を何回も実行します。

- DYNSQL を指定した文 :

  ```
  UPDATE <table> ... WHERE ID = :B
  ```

- NODYNSQL を指定した文 :

  ```
  UPDATE <table> ... WHERE ID = '1234'
  ```

DYNSQL を使用すると、ほとんどの環境で最高の効率化と最大限のスループットを達成できます。ただし、分離されたインスタンスでは、NODYNSQL を使用した場合に、さらに高速化および効率化できることが実証されています。NODYNSQL は、Replicat のスループットが十分でない場合にのみ試行してください。

動的 SQL をサポートしていないターゲット・データベースにレプリケートする場合は、DYNSQL を使用しないでください。

NODYNSQL を使用する場合は、NOBINARYCHARS パラメータも使用する必要があります。これらのいずれかのパラメータを使用する前に、Oracle サポートに連絡してください。詳細は、http://support.oracle.com を参照してください。

Oracle GoldenGate for MySQL は、NODYNSQL モードでの LOB レプリケーションをサポートしていません。

デフォルト DYNSQL

構文

```
DYNSQL | NODYNSQL
```

EBCDICTOASCII

適用対象 Extract データ・ポンプおよび Replicat

EBCDICTOASCII パラメータでは、z/OS システム上の DB2 ターゲット・データベースに送信する際に、入力トレイル内の文字データを EBCDIC から ASCII 形式に変換します。このパラメータを指定して、すべての EBCDIC 列およびユーザー・トークン・データの ASCII への変換をリクエストできます。このパラメータは SOURCEDB パラメータの前に指定する必要があります。このパラメータは、入力トレイル・ファイルがリリース v10.0 より前の Extract によって作成された場合にのみ必要です。それ以外の場合、変換は自動的に行われるため、無視されます。

バージョン 11.2.1 より、データ・ポンプによる変換はできません。

デフォルト なし

構文 `EBCDICTOASCII`

ENABLEMONITORING

適用対象 GLOBALS

ENABLEMONITORING パラメータでは、Oracle GoldenGate Monitor から Oracle GoldenGate インスタンスのモニタリングを有効化します。Manager に、ステータスおよびその他の情報を Oracle GoldenGate Monitor クライアントに提供するモニタリング・ポイントを公開するように指示します。モニタリング・ポイントからの情報を Oracle GoldenGate Monitor サーバーに送信する Java エージェントを制御するには、GGSCI で次のコマンドを使用します。

CREATE DATASTORE

REPAIR DATASTORE

START JAGENT

STOP JAGENT

特定のプラットフォームでモニタリングを有効化する前に、『*Oracle GoldenGate Monitor 管理者ガイド*』を参照して、そのオペレーティング・システムがサポートされていることを確認してください。このガイドには、Oracle GoldenGate Monitor のインストールおよび構成に関する方法も記載されています。Oracle GoldenGate Monitor の使用に関するヘルプ情報は、オンライン・ヘルプを参照してください。

このパラメータは、NonStop SQL/MX には無効です。

デフォルト 無効

構文 `ENABLEMONITORING`

ENCRYPTTRAIL | NOENCRYPTTRAIL

適用対象 Extract

ENCRYPTTRAIL および NOENCRYPTTRAIL パラメータでは、トレイルまたは抽出ファイルに書き込むデータを暗号化します。ENCRYPTTRAIL を使用すると、データ・リンクおよびファイル内のすべてのレコードが暗号化されます。デフォルトの NOENCRYPTTRAIL を使用すると、暗号化は行われません。

暗号化は、Extract パラメータ・ファイル内の次のパラメータで指定しているトレイルまたは抽出ファイルに対して使用できます。

EXTTRAIL および RMTTRAIL

EXTFILE および RMTFILE

ENCRYPTTRAIL および NOENCRYPTTRAIL は、トレイルまたはファイルに固有です。一方のパラメータは、もう一方のパラメータが見つかるまで、パラメータ・ファイル内でそれ以降のすべてのトレイルまたは抽出ファイルに影響します。このパラメータは、適用するトレイルのパラメータ・エントリの前に配置する必要があります。

データを復号化するには、次のプロセスのパラメータ・ファイルで DECRYPTTRAIL パラメータを使用します (184 ページを参照してください)。

- 暗号化されたファイルを読み取り、列マッピング、フィルタリング、変換またはその他の操作を実行する必要があるデータ・ポンプ。データ・ポンプのパラメータ・ファイルに ENCRYPTTRAIL を含めて、ダウンストリーム・トレイルまたはファイルに書き込む前に任意またはすべてのデータをもう一度暗号化することも、ENCRYPTTRAIL を省略して、暗号化されていないフォーマットでデータを送信することもできます。
- 暗号化されたトレイルまたはファイルを読み取る Replicat プロセス。

ENCRYPTTRAIL および NOENCRYPTTRAIL は、ASCII フォーマットでファイルにデータを書き込むために FORMATASCII を使用しているときは使用できません。トレイルまたはファイルは、デフォルトの Oracle GoldenGate 正規フォーマットで書き込まれる必要があります。

ENCRYPTTRAIL は、データ・ブロックのみを暗号化します。ユーザー・トークンは暗号化されません。

ENCRYPTTRAIL は、ETOLDFORMAT と互換性がありません。

デフォルト NOENCRYPTTRAIL

構文

```
ENCRYPTTRAIL [{AES128 | AES192 | AES256} KEYNAME <keyname>] | NOENCRYPTTRAIL
```

引数	説明
ENCRYPTTRAIL	オプションなしの ENCRYPTTRAIL は、256 鍵バイト置換を指定します。このフォーマットは安全ではないため、本番環境では使用しないでください。以前の Oracle GoldenGate バージョンとの下位互換性を維持するためにのみ使用します。

引数	説明
{AES128 | AES192 | AES256} KEYNAME <keyname>	Advanced Encryption Standard (AES) 暗号化を指定します。これは、高度なデータ・セキュリティを必要とする政府や他の組織が使用する、対称鍵の暗号化標準です。 ファイルを暗号化する AES 暗号を、次の中から 1 つ指定します。 ◆ AES128 は、ブロック・サイズが 128 ビットでキー・サイズが 128 ビットです。 ◆ AES192 は、ブロック・サイズが 128 ビットでキー・サイズが 192 ビットです。 ◆ AES256 は、ブロック・サイズが 128 ビットでキー・サイズが 256 ビットです。 <keyname> には、ユーザー定義の暗号化鍵の論理名を指定します。Oracle GoldenGate は、鍵名を使用して、ENCKEYS 参照ファイル内の実際の鍵を参照します。AES を使用するには、次を行う必要があります。 ◆ 暗号化鍵を生成します。 ◆ それを ENCKEYS 参照ファイルに保管します。 ◆ 暗号化または復号化(またはその両方)を実行するすべてのシステムに、ENCKEYS をコピーします。 Oracle 以外のデータベースに AES 暗号化を使用するには、プロセスを開始する前に、次のように、環境変数として、Oracle GoldenGate インストール・ディレクトリの lib サブディレクトリのパスを指定する必要があります。 ◆ UNIX: LD_LIBRARY_PATH または SHLIB_PATH 変数への入力として、パスを指定します。次に例を示します。 `setenv LD_LIBRARY_PATH ./lib:$LD_LIBRARY_PATH` ◆ Windows: PATH 変数にパスを追加します。 SETENV パラメータを使用して、プロセスのセッション変数として設定できます。 暗号化の使用の詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』のセキュリティ・ガイドラインを参照してください。

例

次の例では、Extract プロセスが 2 つのトレイルに書き込みます。emp 表のデータは、AES-192 暗号で暗号化されているトレイル "em" に書き込まれます。stores 表のデータは、暗号化されていないトレイル "st" に書き込まれます。

```
ENCRYPTTRAIL AES192, KEYNAME mykey1
RMTTRAIL /home/ggsora/dirdat/em
TABLE hr.emp;
NOENCRYPTTRAIL
RMTTRAIL /home/ggsora/dirdat/st
TABLE ops.stores;
```

例 上記の例のかわりとして、ENCRYPTTRAIL パラメータの前に暗号化されていないすべてのトレイルをリストする場合は、NOENCRYPTTRAIL を省略できます。

```
RMTTRAIL /home/ggsora/dirdat/st
TABLE ops.stores;
ENCRYPTTRAIL AES192, KEYNAME mykey1
RMTTRAIL /home/ggsora/dirdat/em
TABLE hr.emp;
```

詳細な例は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』のセキュリティ・ガイドラインを参照してください。

END

適用対象 Replicat

END パラメータでは、タイムスタンプが指定した時点と一致する、データ・ソース内の最初のレコードを検出したときに、Replicat を終了します。

END を使用しない場合、プロセスは、次の時点まで継続的に実行されます。

- トランザクション・ログまたはトレイルの末尾に到達したとき。この時点で、プロセスは正常に終了します。
- コマンド・シェルから手動で終了したとき。

END を SPECIALRUN パラメータとともに使用して、ターゲット表を継続的に更新するのではなく、ポイントインタイム・スナップショットとしてデータを送信します。

デフォルト 継続処理

構文

```
END {<date> [<time>] | RUNTIME}
```

引数	説明
`<date> [<time>]`	タイムスタンプがこのパラメータの指定を超えるデータ・ソース内のレコードに到達したときに、Replicat を終了させます。 有効な値 : ◆ <date> は、yyyy-mm-dd フォーマットの日付です。 ◆ <time> は、24 時間表記の hh:mi[:ss[.cccccc]] フォーマットの時刻です。
`RUNTIME`	タイムスタンプが現在の日時を超えるデータ・ソース内のレコードに到達したときに、Replicat を終了させます。この時点までのタイムスタンプを持つ未処理レコードは、すべて処理されます。RUNTIME を使用するメリットの 1 つは、実行のたびにパラメータ・ファイルで日時を変更する必要がない点です。かわりに、バッチ・プログラミング内でプロセスの開始時刻を制御できます。

例 1

```
SPECIALRUN
END 2010-12-31 17:00:00
```

例 2

```
SPECIALRUN
END RUNTIME
```

EOFDELAY | EOFDELAYCSECS

適用対象 Extract および Replicat

EOFDELAY または EOFDELAYCSECS パラメータでは、Extract、データ・ポンプまたは Replicat が、データ・ソース内の現在のデータの末尾に到達した後に、新しいデータを確認する間隔を制御します。このパラメータの値を増やすと、新しいデータの読取りによるシステム I/O のオーバーヘッドを低減できます。

注意 **この値を大幅に増やすと、特にソース・データベース上のアクティビティが少ない場合には、ターゲット・データの遅延が増加する可能性があります。**

このパラメータは、SOURCEISTABLE を使用している場合は無効です。

デフォルト 1 秒

構文 `EOFDELAY <seconds> | EOFDELAYCSECS <centiseconds>`

引数	説明
`<seconds>`	処理するデータを探すまでの遅延 (秒)。
`<centiseconds>`	処理するデータを探すまでの遅延 (ミリ秒)。

例 `EOFDELAY 3`

ETOLDFORMAT

適用対象 Extract

ETOLDFORMAT パラメータでは、Oracle GoldenGate リリース 6.0 以前の Replicat バージョンと互換性のあるフォーマットでトレイルを生成します。

Oracle GoldenGate リリース 10 以上では、ETOLDFORMAT は互換性レベルが 0 であることを意味します。ETOLDFORMAT は、すべての出力ファイルにグローバルに適用されます。ETOLDFORMAT の他に、EXTFILE、EXTTRAIL、RMTFILE または RMTTRAIL で互換性レベルが指定されている場合、Extract プロセスは異常終了します。詳細は、これらのパラメータの説明を参照してください。

ETOLDFORMAT は、ENCRYPTTRAIL と互換性がありません。

デフォルト なし

構文 `ETOLDFORMAT`

EXTFILE

適用対象 Extract および Replicat

EXTFILE パラメータでは、ローカル・システム上のデータ・ポンプ Extract グループによって読み取られるローカル・ファイルである抽出ファイルを指定するか、SPECIALRUN を使用する場合は Replicat が読み取るローカル抽出ファイルを指定します。抽出ファイルのサイズの上限は 2GB です。

EXTFILE は、関連するすべての TABLE または MAP 文よりも先に指定する必要があります。異なるファイルを定義するために、複数の EXTFILE 文を使用できます。

ENCRYPTTRAIL パラメータ (192 ページ) を使用して、このファイル内のデータを暗号化できます。

デフォルト なし

構文

```
EXTFILE <file name>
[, MAXFILES <number>]
[, MEGABYTES <megabytes>]
[, FORMAT RELEASE <major>.<minor>]
```

引数	説明
`<file name>`	Extract および Replicat に有効です。抽出ファイルの相対名または完全修飾名を指定します。
`MAXFILES <number>`	Extract に有効です。単一のファイルでなく、連続するファイルを作成させます。1 つのファイルのサイズがオペレーティング・システムに許可される制限を越える可能性があるときに使用します。MAXFILES を使用して、必要な数のファイルを作成できます。エージングされたファイルには、6 桁の順序番号が付けられます (例 : datafile000002)。チェックポイントは、これらのファイルには保持されません。MAXFILES を使用するときは、MEGABYTES も使用して、連続ファイル内の各ファイルの最大サイズを設定します。
`MEGABYTES <megabytes>`	Extract に有効です。ファイル (MAXFILES を使用している場合は各ファイル) の最大サイズを定義します。抽出ファイルのサイズの上限は 2GB です。
`FORMAT RELEASE <major>.<minor>`	Extract から、トレイル、ファイル、または (リモート・タスクの場合) 別のプロセスに送信されるデータのメタデータ・フォーマットを指定します。リーダー・プロセスは、メタデータに基づいて、データ・レコードが自身がサポートしているバージョンかどうかを把握します。メタデータのフォーマットは、Oracle GoldenGate プロセスのリリースによって異なります。古い Oracle GoldenGate リリースには、新しいリリースとは異なるメタデータが含まれます。 ◆ FORMAT は必須のキーワードです。 ◆ RELEASE は Oracle GoldenGate のリリース・バージョンを指定します。<major> はメジャー・バージョン番号で、<minor> はマイナー・バージョン番号です。有効な値は、9.0 から現在の Oracle GoldenGate リリース番号です。(9.0 以前の Oracle GoldenGate リリースを使用している場合は、9.0 または 9.5 を指定してください。) リリース・バージョンは、プログラムによって適切なトレイル・フォーマット互換性レベルにマッピングされます。デフォルトは、このトレイルに書き込むプロセスの現在のバージョンです。 FORMAT と RECOVERYOPTIONS パラメータの間には、依存関係があります。RECOVERYOPTIONS を APPENDMODE に設定する場合、FORMAT は RELEASE 10.0 以上に設定する必要があります。RECOVERYOPTIONS を OVERWRITEMODE に設定する場合、FORMAT は RELEASE 9.5 以下に設定する必要があります。

例 1 `EXTFILE dirdat/datafile`

例 2 `EXTFILE dirdat/extdat, MAXFILES 3, MEGABYTES 2`

例 3 `EXTFILE /ggs/dirdat/extdat, FORMAT RELEASE 10.4`

EXTRACT

適用対象 Extract

EXTRACT パラメータでは、オンライン変更同期用の Extract グループを指定します。このパラメータによって、現在の実行と前回の実行が関連付けられるので、データ変更を継続的に処理し、ソース表とターゲット表の同期性を維持できます。Extract は継続的に実行し、データ・ソースおよびトレイルにチェックポイントを保持することにより、計画的または計画外のプロセス終了、システム停止、ネットワーク障害が発生した場合にもデータ整合性とフォルト・トレランスを確保します。EXTRACT は、パラメータ・ファイルの最初のエントリにする必要があります。

変更同期の実装の詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。

デフォルト なし

構文

```
EXTRACT <group name>
```

引数	説明
`<group name>`	ADD EXTRACT コマンドで定義したグループ名。

例 次の例では、"finance" という名前の Extract グループを指定します。

```
EXTRACT finance
```

EXTTRAIL

適用対象 Extract および Replicat

EXTTRAIL パラメータでは、ADD EXTTRAIL コマンドで作成されたローカル・システム上のトレイルを指定します。このトレイルは、データ・ポンプ Extract グループ、またはローカル・システム上の Replicat グループによって読み取られます。

EXTTRAIL は、関連するすべての TABLE 文よりも先に指定する必要があります。異なるトレイルを定義するために、複数の EXTTRAIL 文を使用できます。

パッシブ・モードで構成されている Extract には、EXTTRAIL を使用しないでください。詳細は、17 ページの「ADD EXTRACT」を参照してください。

ENCRYPTTRAIL パラメータ (192 ページ) を使用して、このトレイル内のデータを暗号化できます。

デフォルト なし

構文

```
EXTTRAIL <file name>
[, FORMAT RELEASE <major>.<minor>]
```

引数	説明
<file name>	トレイルの相対名または完全修飾名。最大文字数 (2 文字) を名前に使用します。エージングされたトレイル・ファイルは、この名前に 6 桁の順序番号が追加されます (例： /ggs/dirdat/rt000001)。
FORMAT RELEASE <major>.<minor>	Extract から、トレイル、ファイル、または (リモート・タスクの場合) 別のプロセスに送信されるデータのメタデータ・フォーマットを指定します。リーダー・プロセスは、メタデータに基づいて、データ・レコードが自身がサポートしているバージョンかどうかを把握します。メタデータのフォーマットは、Oracle GoldenGate プロセスのリリースによって異なります。古い Oracle GoldenGate リリースには、新しいリリースとは異なるメタデータが含まれます。 ◆ FORMAT は必須のキーワードです。 ◆ RELEASE は Oracle GoldenGate のリリース・バージョンを指定します。<major> はメジャー・バージョン番号で、<minor> はマイナー・バージョン番号です。有効な値は、9.0 から現在の Oracle GoldenGate リリース番号です。(9.0 以前の Oracle GoldenGate リリースを使用している場合は、9.0 または 9.5 を指定してください。) リリース・バージョンは、プログラムによって適切なトレイル・フォーマット互換性レベルにマッピングされます。デフォルトは、このトレイルに書き込むプロセスの現在のバージョンです。 FORMAT と RECOVERYOPTIONS パラメータの間には、依存関係があります。RECOVERYOPTIONS を APPENDMODE に設定する場合、FORMAT は RELEASE 10.0 以上に設定する必要があります。RECOVERYOPTIONS を OVERWRITEMODE に設定する場合、FORMAT は RELEASE 9.5 以下に設定する必要があります。 Oracle GoldenGate トレイル・ファイルのバージョニングおよびリカバリ・モードの詳細は、297 ページの付録 3 を参照してください。

例 1 `EXTTRAIL dirdat/ny`

例 2 `EXTTRAIL /ggs/dirdat/ex, FORMAT RELEASE 10.4`

FETCHOPTIONS

適用対象 Extract

FETCHOPTIONS パラメータでは、Oracle GoldenGate が次の状況でデータをフェッチする方法を制御します。

- Extract が更新操作を再構築するために十分な情報がトランザクション・レコードに含まれていない
- TABLE 文の FETCHCOLS 句の結果として、Oracle GoldenGate が列値をフェッチする必要がある

FETCHOPTIONS は表に固有です。1 つの FETCHOPTIONS 文は、別の FETCHOPTIONS 文が見つかるまで、後続のすべての TABLE 文に適用されます。

ほとんどのインストールでは、デフォルトのフェッチ・プロパティで十分に機能します。

デフォルト　行方不明の行を無視し、処理を継続する

構文

```
FETCHOPTIONS
[, FETCHPKUPDATECOLS]
[, MISSINGROW <action>]
[, NOFETCH]
[, USEKEY | NOUSEKEY]
[, USELATESTVERSION | NOUSELATESTVERSION]
[, USESNAPSHOT | NOUSESNAPSHOT]
[, USEROWID | NOUSEROWID]
```

引数	説明
FETCHPKUPDATECOLS	主キーが更新された場合、使用できないすべての列をフェッチします。このオプションは、デフォルトではオフです。オフにすると、有効化されている他の FETCHOPTIONS オプションに従って、列のフェッチが実行されます。 オンにすると、主キー列への更新中のみ有効になります。結果は、TABLE 文で FETCHCOLS (*) を使用した場合と同じです。LOB 列はフェッチに含まれます。 HANDLECOLLISIONS を使用する場合に、このパラメータを使用します。Replicat が行方不明の更新を検出すると、すべての列は、更新を挿入に変更できるようになります。
MISSINGROW <action>	Oracle GoldenGate がフェッチする行を特定できない場合のレスポンスを指定し、行 (変更された値) の一部のみでも処理できるようにします。通常、行が特定できない理由は、変更レコードが作成されてからフェッチがトリガーされるまでの間に行が削除されてしまっているか、必要な行イメージが指定されている UNDO 保存期間以前のものであったためです。 <action> には、次のいずれかを指定できます。 IGNORE 状況を無視して処理を継続します。これはデフォルトです。 REPORT 状況と行のコンテンツを破棄ファイルにレポートしますが、部分的な行の処理を継続します。DISCARDFILE パラメータで、破棄ファイルを指定しておく必要があります。 DISCARD データを破棄し、部分的な行は処理しません。DISCARDFILE パラメータで、破棄ファイルを指定しておく必要があります。 ABEND データを破棄し、処理を中止します。DISCARDFILE パラメータで、破棄ファイルを指定しておく必要があります。
NOFETCH	Extract によるデータベースからの列のフェッチを防ぎます。Extract はトレイルにレコードを書き込みますが、列が行方不明であることを示すトークンを挿入します。

引数	説明
USEKEY | NOUSEKEY	Oracle に有効です。フェッチする行を見つけるために Oracle GoldenGate が主キーを使用するかどうかを決定します。 USEKEY および USEROWID 両方を指定する場合、レコードにより高速にアクセスできる ROWID の方が優先されます。デフォルトは USEROWID です。
USELATESTVERSION | NOUSELATESTVERSION	Oracle に有効です。USESNAPSHOT とともに使用します。デフォルトの USELATESTVERSION では、UNDO 表領域からフェッチできない場合に、Extract にソース表からデータをフェッチさせます。NOUSELATESTVERSION は、スナップショット・フェッチが失敗した場合に、Extract にその状況を無視して処理を継続させます。 スナップショット・フェッチが失敗した場合の代替アクションを指定するには、MISSINGROW オプションを使用します。
USESNAPSHOT | NOUSESNAPSHOT	Oracle に有効です。デフォルトの USESNAPSHOT では、Extract に、REDO レコードから完全には取得できない特定の操作の再構築に必要なデータの適切なスナップショットを、Oracle Flashback メカニズムを使用してフェッチさせます。NOUSESNAPSHOT では、Extract に、フラッシュバック・ログではなくソース表から必要なデータをフェッチさせます。Oracle GoldenGate の Oracle からのデータ・フェッチ方法の詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。
USEROWID | NOUSEROWID	Oracle に有効です。フェッチする行を見つけるために Oracle GoldenGate が行 ID を使用するかどうかを決定します。 USEKEY および USEROWID 両方を指定する場合、レコードにより高速にアクセスできる ROWID の方が優先されます。デフォルトは USEROWID です。

例 1 次の例では、Extract に Flashback Query を使用してデータをフェッチさせ、フェッチが失敗した場合にはその状況を無視してレコードの処理を継続させます。

```
FETCHOPTIONS USESNAPSHOT, NOUSELATESTVERSION
```

例 2 次の例では、Extract に Flashback Query を使用してデータをフェッチさせ、データを使用できない場合には Extract を異常終了させます。

```
FETCHOPTIONS USESNAPSHOT, NOUSELATESTVERSION, MISSINGROW ABEND
```

FILTERDUPS | NOFILTERDUPS

適用対象 Replicat

FILTERDUPS および NOFILTERDUPS パラメータでは、アプリケーションが同一トランザクション内で同一のレコードに複数の操作を実行するときに NonStop システムで発生する可能性がある異常を処理します。このタイプのトランザクションによって、TMF オーディット・トレイルのレコードの順番が不正になり、Replicat が異常終了することがあります。次に例を示します。

- ソース・アプリケーションが削除を実行してから挿入を実行した場合でも、オーディット・トレイルでは、同一の主キーの削除の前に挿入が発生することがあります (結果として、Replicat が挿入を実行するときに重複レコード・エラーが発生します)。

- オーディット・トレイルでは、同一の主キーの挿入の前に更新が発生することがあります (結果として、Replicat が更新を実行するときに行方不明レコード・エラーが発生します)。

FILTERDUPS は、Replicat の異常終了を防止するために、次のようにしてこの状況を解決します。

- 挿入するレコードが重複する場合、Replicat はこの挿入操作をトランザクションの最後に保存します。その後、同一主キーの削除を検出すると、Replicat は削除を実行し、その後に挿入を実行します。
- 更新するレコードが行方不明の場合、Replicat はこの更新操作をトランザクションの最後に保存します。その後、同一の主キーの挿入を検出すると、Replicat は挿入を実行し、その後に更新を実行します。

このような異常は、一般に IDX 医療アプリケーションや一部の BASE24 バンク・アプリケーションなどで見られます (ただし、これらのアプリケーションに限定されません)。FILTERDUPS は、重複するレコードや行方不明のレコードのために Replicat が異常終了し、その原因が NonStop システムで開始された不正な順序のトランザクションであることが判明している場合にのみ使用してください。この状況の診断には、Logdump ユーティリティを使用できます。『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。

FILTERDUPS および NOFILTERDUPS は、必要に応じて異なる MAP 文グループの例外処理を有効化または無効化する個別スイッチとして使用できます。

デフォルト NOFILTERDUPS

構文

```
FILTERDUPS | NOFILTERDUPS
```

例 この例は、FILTERDUPS を Orders に対して有効にしますが、同一のパラメータ・ファイルのそれ以降に定義されているすべての MAP 文に対しては無効にします。

```
FILTERDUPS
MAP $DATA1.SQLDAT.ORDERS, TARGET MASTER.ORDERS;
NOFILTERDUPS
```

FLUSHSECS | FLUSHCSECS

適用対象 Extract

FLUSHSECS および FLUSHCSECS パラメータでは、Oracle GoldenGate がいつ Extract メモリー・バッファをフラッシュするかを制御します。リモート・システムにデータを送信する際、Extract はデータをバッファしてネットワーク・パフォーマンスを最適化します。バッファは、一杯になるか、FLUSHSECS または FLUSHCSECS で指定された時間が経過した後に、ターゲット・システムにフラッシュされます。バッファがフラッシュされてデータが送信されるまで、ターゲットのユーザーは変更データを使用できません。バッファのサイズを制御するには、RMTHOST の TCPBUFSIZE オプションを使用します (314 ページを参照してください)。

FLUSHSECS または FLUSHCSECS の値を増やすと、ネットワーク使用効率が若干高まることがありますが、ソース・システムのアクティビティが少なく、バッファが一杯にならない場合には、ターゲット・データの遅延が増える可能性があります。ソース表のアクティビティが多い場合は、FLUSHSECS および FLUSHCSECS はほとんど効果がありません。

デフォルト 1 秒

構文

```
FLUSHSECS <seconds> | FLUSHCSECS <centiseconds>
```

引数	説明
<seconds>	バッファをフラッシュするまでの遅延 (秒)。
<centiseconds>	バッファをフラッシュするまでの遅延 (センチ秒)。

例 FLUSHSECS 80

FORMATASCII

適用対象 Extract

FORMATASCII パラメータでは、デフォルトの Oracle GoldenGate 正規フォーマットではなく、外部 ASCII フォーマットでデータを出力します。FORMATASCII を使用すると、大半のデータベース・ロード・ユーティリティおよび ASCII 入力を必要とするその他のプログラムと互換性を持つように出力をフォーマットできます。このパラメータは、*File-to-Database-Utility* による初期ロード方法を使用するときに必要です。

FORMATASCII 文は、パラメータ・ファイル内のこの文以降のすべての抽出ファイルまたはトレイルに影響します。パラメータ・ファイル内の文の相対的な順序は重要です。ファイルまたはトレイルの指定の後にリストされた場合、FORMATASCII は効力がありません。

制約事項

- データが Replicat プロセスによって処理される場合は、FORMATASCII を使用しないでください。Replicat はデフォルトの正規フォーマットを受け付けます。
- FORMATSQL または FORMATXML を使用しているときは、FORMATASCII を使用しないでください。
- データに LOB が含まれている場合、FORMATASCII を使用しないでください。
- Extract がマルチバイトの DB2 サブシステムに接続している場合、FORMATASCII を使用しないでください。

デフォルト出力

表名や列名などのデータベース・オブジェクト名、および CHAR および VARCHAR データは、オペレーティング・システムのデフォルトの文字セットで書込みが行われます。

パラメータ・オプションを使用しない場合、FORMATASCII は次のフォーマットでレコードを生成します。

行 1、次のタブ区切りリスト :

- 操作タイプ・インジケータ : I、D、U、V (挿入、削除、更新、圧縮更新)
- ビフォアまたはアフター・イメージ・インジケータ : B または A
- オペレーティング・システムの文字セットでの表名
- 列名、列値、列名、列値など
- 改行文字 (新しい行を開始)

行 2、次のタブ区切り開始トランザクション・レコード :

- 開始トランザクション・インジケータ、B
- トランザクションのコミット時のタイムスタンプ

- コミットが検出されたトランザクション・ログの順序番号
- トランザクション・ログ内のコミット・レコードの相対バイト・アドレス (RBA)

行 3、次のタブ区切りコメント・レコード :

- コミット文字、C
- 改行文字

ソース・トランザクション内のすべてのレコードは、開始インジケータとコミット・インジケータの間に含まれます。コミット・タイムスタンプと RBA の各組合せは一意です。

カスタム出力

出力フォーマットは、オプションの引数でカスタマイズできます。構文の説明を参照してください。

デフォルト 「デフォルト出力」を参照してください。

構文 `FORMATASCII [, <option>] [, ...]`

オプション	説明
`BCP`	SQL Server の BCP、DTS、または SQL Server Integration Services (SSIS) バルクロード・ユーティリティと互換性を持つように出力をフォーマットします。
`DATE | TIME | TS`	次のいずれかを出力します。 ◆ DATE は日付を出力します (年～日)。 ◆ TIME は時間を出力します (年～秒)。 ◆ TS はトランザクションのタイムスタンプを出力します (年～秒の端数)。
`CHARSET <set>`	(Oracle SQL*Loader) Oracle NCHAR 列の ASCII 文字のエンコーディングを指定します。有効な値は UTF8 です。 ソース表に NCHAR データ、および UTF-8 に設定された可変長文字が含まれている場合、CHARSET によって、ロードは文字長セマンテックを含めることができます。 注意 : NCHAR 列および CHAR 列の両方に 8 ビット ASCII 文字が含まれている場合、生成されるファイルには、オペレーティング・システムのネイティブの 8 ビット ASCII 文字コーディングと UTF-8 コーディングが混在して含まれているため、ロードは失敗します。
`DELIMITER <delimiter>`	代替デリミタ文字 (デフォルトはタブ)。 有効な値 : ◆ TAB(タブで区切り) ◆ 一重引用符で囲んだ文字 (例 : `'/'`)
`EXTRACOLS <number>`	各レコードの末尾に追加列のためのプレースホルダを含めます。このオプションは、ターゲット表にソース表よりも多くの列が含まれる場合に使用してください。

オプション	説明
NAMES | NONAMES	出力の一部に列名を包含または除外します。圧縮更新では、PLACEHOLDERS オプションも指定している場合を除き、列名は含まれます。
NOHDRFIELDS [IND], [OP]	次のように出力を抑止します。 ◆ オプションなしの NOHDRFIELDS は、データ値を除くすべてを抑止します。 ◆ IND は、ビフォアまたはアフター・インジケータ (B または A) およびデータ値を除くすべてを抑止します。 ◆ OP は、操作タイプ・インジケータ (I、D、U、V) およびデータ値を除くすべてを抑止します。
NOQUOTE	文字データから引用符を除外します。NOQUOTE を指定しない場合、文字は一重引用符で囲まれます。
NOTRANSTMTS	トランザクション情報を除外します。
NULLISSPACE	NULL 列を空の列として出力します。NULLISSPACE を指定しない場合、NULL 列は単語 "NULL" として出力されます。
PLACEHOLDERS	行方不明の列に対してプレースホルダを出力します。たとえば、4 列の表で 2 番目および 4 番目の列が見つからない場合、データは次のように出力されます。 `'ABC',,123,,`
SQLLOADER	Oracle SQL*Loader ユーティリティまたは IBM Load Utility プログラムと互換性を持つ固定長 ASCII フォーマット・ファイルを生成します。

例 1 次の例は、test.customer という名前のソース表とサンプル・トランザクションに基づいています。この例では、様々な FORMATASCII オプションがどのように出力を構成するかを示します。

表 test.customer:

```
CUSTNAME    CHAR(10)   Primary key
LOCATION    CHAR(10)
BALANCE     INTEGER
```

トランザクション:

```
INSERT INTO CUSTOMER VALUES ("Eric", "San Fran", 550);
UPDATE CUSTOMER SET BALANCE = 100 WHERE CUSTNAME = "Eric";
COMMIT;
```

例 1 オプションなしの FORMATASCII では、次の出力が生成されます。

```
B,2011-01-21:14:09:46.421335,8,1873474,
I,A,TEST.CUSTOMER,CUSTNAME,'Eric',LOCATION,
'San Fran',BALANCE,550,
V,A,TEST.CUSTOMER,CUSTNAME,'Eric',BALANCE,100,
C,
```

例 2 FORMATASCII, NONAMES, DELIMITER '|' では、次の出力が生成されます。

```
B|2011-01-21:14:09:46.421335|8|1873474|
I|A|CUSTOMER|'Eric'|'San Fran'|550|
V|A|CUSTOMER|CUSTNAME|'Eric'|BALANCE|100|
C|
```

レコードは圧縮更新され、PLACEHOLDERS が使用されなかったため、最後のレコードは、CUSTNAME および BALANCE 列の列名を返します。

例 3 FORMATASCII, NOHDRFIELDS, OP, TS, NONAMES, NOQUOTE では、次の出力が生成されます。

```
I,CUSTOMER,2011-01-21:14:09:46.421335,Eric,San Fran,550,
V,CUSTOMER,2011-01-21:14:09:46.421335,Eric,,100,
```

圧縮更新レコードの 2 番目の列の不在の値は、2 つの連続するコンマで示されています。

FORMATSQL

適用対象 Extract

FORMATSQL パラメータでは、デフォルトの Oracle GoldenGate 正規フォーマットではなく、外部 SQL フォーマットでデータを出力します。FORMATSQL は、Oracle GoldenGate Replicat 以外のユーティリティで SQL および Enscribe 表に適用可能な、SQL 文 (INSERT、UPDATE および DELETE) を生成します。

> **注意** **データが Replicat プロセスによって処理される場合は、FORMATSQL を使用しないでください。Replicat はデフォルトの正規フォーマットを受け付けます。FORMATASCII または FORMATXML を使用しているときは、FORMATSQL を使用しないでください。**

FORMATSQL 文は、それ以降に定義されているすべての抽出ファイルまたはトレイルに影響します。

Extract がマルチバイトの DB2 サブシステムに接続している場合、FORMATSQL を使用しないでください。

デフォルト出力

表名や列名などのデータベース・オブジェクト名、および CHAR および VARCHAR データは、オペレーティング・システムのデフォルトの文字セットで書込みが行われます。

オプションなしの FORMATSQL トランザクションは、コンマ区切りフォーマットで次のように出力されます。

- 開始トランザクション・インジケータ、B
- トランザクションのコミット時のタイムスタンプ
- コミットが検出されたトランザクション・ログの順序番号
- トランザクション・ログ内のコミット・レコードの相対バイト・アドレス (RBA)
- SQL 文
- コミット・インジケータ、C
- 改行インジケータ

トランザクション内のすべてのレコードは、開始インジケータとコミット・インジケータの間に含まれます。コミット・タイムスタンプと RBA の各組合せは一意です。

カスタム出力

出力フォーマットは、オプションの引数でカスタマイズできます。構文の説明を参照してください。

デフォルト 「デフォルト出力」を参照してください。

構文 `FORMATSQL [<option>] [, ...]`

オプション	説明
NONAMES	挿入操作の列名を省略します (挿入にはすべての列名が含まれているため)。このオプションにより、ファイル・サイズを低減できます。
NOPKUPDATES	ターゲットの主キーの列に影響する UPDATE 操作を、次に INSERT が続く DELETE に変換します。デフォルト (NOPKUPDATES なし) では、出力は標準の UPDATE 操作です。
ORACLE	次の例のように、日付および時刻列を SQL*Plus が受け付けるフォーマットに変換し、Oracle データベースと互換性を持つようにレコードをフォーマットします。 `TO_DATE('2011-01-01','YYYY-MM-DD')`

例 `FORMATSQL ORACLE, NONAMES`

FORMATXML

適用対象 Extract

FORMATXML パラメータでは、デフォルトの Oracle GoldenGate 正規フォーマットではなく、XML フォーマットでデータを出力します。FORMATXML 文は、それ以降に定義されているすべての抽出ファイルまたはトレイルに影響します。

表名や列名などのデータベース・オブジェクト名、および CHAR および VARCHAR データは、オペレーティング・システムのデフォルトの文字セットで書込みが行われます。

FORMATXML を使用するときは、NOBINARYCHARS パラメータを使用してください。NOBINARYCHARS は、Oracle GoldenGate にバイナリ・データを NULL で終わる文字列として処理させる、ドキュメントに記載されていないパラメータです。NOBINARYCHARS を使用する前に、Oracle サポートに連絡してください。詳細は、http://support.oracle.com を参照してください。

制約事項

- データが Replicat プロセスによって処理される場合は、FORMATXML を使用しないでください。Replicat はデフォルトの正規フォーマットを受け付けます。FORMATASCII または FORMATSQL を使用しているときは、FORMATXML を使用しないでください。
- Extract がマルチバイトの DB2 サブシステムに接続している場合、FORMATXML を使用しないでください。

デフォルト なし

構文 `FORMATXML [<option>] [, ...]`

オプション	説明
`INLINEPROPERTIES |NOINLINEPROPERTIES`	XML タグ内にプロパティを含めるか、別に書き込むかどうかを制御します。デフォルトは INLINEPROPERTIES です。
`TRANS | NOTRANS`	トランザクション境界およびコミットのタイムスタンプを XML 出力に含めるかどうかを制御します。TRANS はデフォルトです。

例 `FORMATXML NOINLINEPROPERTIES, NOTRANS`

FUNCTIONSTACKSIZE

適用対象 Extract および Replicat

FUNCTIONSTACKSIZE パラメータでは、Oracle GoldenGate 列変換ファンクションの処理に使用されるメモリー・スタックのサイズを制御します。メモリー・スタックには、Oracle GoldenGate ファンクションとやり取りする引数が保持されます。このパラメータは、Oracle GoldenGate からスタック・サイズを増やすように通知するメッセージが返されないかぎり、使用する必要はありません。メッセージは次のようになります。

```
Not enough stack space. Specify FUNCTIONSTACKSIZE greater than {0,number,0}
```

このようなメッセージは、多数のファンクションまたは引数を使用している場合に返されることがあります。

FUNCTIONSTACKSIZE なしのデフォルトは、200 引数で、この値では Oracle GoldenGate のパフォーマンスおよびシステム・メモリー使用率が最適化されます。このパラメータの値を増やすと、パフォーマンスおよびシステム・メモリー使用率に悪影響が及ぶことがあります。

FUNCTIONSTACKSIZE は、パラメータ・ファイル内で、ファンクションを含むすべてのパラメータよりも前にリストする必要があります。FUNCTIONSTACKSIZE はグローバル・パラメータです。これはパラメータ・ファイル内のすべての句に影響します。

デフォルト 200 引数

構文 `FUNCTIONSTACKSIZE <stack size>`

引数	説明
`<stack size>`	パラメータ句で許可するファンクション引数の数を示す値 (0 ～ 5000)。

例 `FUNCTIONSTACKSIZE 300`

GENLOADFILES

適用対象 Replicat

GENLOADFILES パラメータでは、*File-to-Database-utility* 初期ロード方法を使用するときに、次のユー

ティリティと互換性のある実行ファイルと制御ファイルを生成します。

- Oracle の SQL*Loader ユーティリティ
- Microsoft の BCP、DTS または SQL Server Integration Services (SSIS) ユーティリティ
- IBM の Load Utility (LOADUTIL)

実行ファイルおよび制御ファイルは、Replicat パラメータ・ファイル内の各 MAP 文で生成されます。Replicat は、制御ファイルおよび実行ファイルを生成後に停止し、データを処理しません。

実行ファイルおよび制御ファイルは、ターゲットにロードするデータを含むデータ・ファイルとともに使用します。データ・ファイルを生成するには、Extract パラメータ・ファイル内で FORMATASCII パラメータを使用します。Oracle および z/OS 上の DB2 のユーティリティを使用する場合は、FORMATASCII の SQLLOADER オプションを使用し、Microsoft ユーティリティを使用する場合は BCP オプションを使用します。

FORMATASCII は、ロード・ユーティリティと互換性を持つ外部 ASCII フォーマットで Oracle GoldenGate トレイルまたはファイルに表データを出力します。複数のファイルを指定すると、複数のデータ・ファイルを生成できます。Oracle GoldenGate のロード・ファイル出力の構成方法および初期ロードの実行方法の詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。

注意 **IBM の Load Utility を使用する場合は、RMTHOST パラメータの PARAMS オプションに、-E および -d <defs file> Collector パラメータを指定する必要があります。これらのパラメータは、ASCII から EBCDIC への変換、およびソース定義ファイルの指定のために必要です。**

デフォルトでは、GENLOADFILES は次のファイル名を生成します。

- SQL*Loader 実行ファイルは <source table>.run、制御ファイルは <source table>.ctl という名前が付けられます。<source table> は、MAP 文で指定されたソース表の名前です。
- BCP/DTS/SSIS 実行ファイルは <target table>.bat、制御ファイルは <target table>.fmt **という名前が付けられます。**<target table> は MAP 文で指定されたターゲット表の名前です。
- Load Utility 実行ファイルは <target table>.run、制御ファイルは <target table>.ctl **という名前が付けられます。**<target table> は MAP 文で指定されたターゲット表の名前です。

制御ファイル

制御ファイルには、テンプレートに基づいて生成されたロード・パラメータが含まれます。Oracle GoldenGate は、SQL*Loader、BCP/DTS/SSIS、および Load Utility 用のデフォルト・テンプレートを提供します。必要に応じてテンプレートを変更してロード・ルールを変更することも、新しいテンプレートを作成することもできます。

次に、Oracle GoldenGate テンプレートの例を示します。テンプレートには、ターゲット表、FORMATASCII によって生成されたデータ・ファイル、およびその他の実行パラメータ用のプレースホルダが含まれています。Oracle GoldenGate は、これらのプレースホルダを Replicat パラメータ・ファイルに指定されたパラメータに基づいた値に置き換えます。

図 15 SQL*Loader テンプレート sqlldr.tpl

```
# File Names
controlfile ?target.ctl
runfile     ?target.run
#
# Run File Template
sqlldr userid=?pw control=?target log=?target direct=true
#
# Control File Template
unrecoverable
load data
infile ?source.dat
truncate
into table ?target
```

図 16 BCP/DTS/SSIS テンプレート bcpfmt.tpl

```
# Run File Template
# Substitute your database name for <db>
bcp <db>..?target in ?source.dat -U ?user -P ?pw -f ?target.fmt -e ?target.err
#
# Control File Template
# The value below must specify the BCP version, not the Sybase Adaptive
# Server or Microsoft SQL Server version."bcp -v" can be used to
# determine the correct version number.
12.0
```

図 17 Load Utility テンプレート db2cntl.tpl

```
# File Names
controlfile ?target.ctl
runfile     ?target.run
#
# Run File Template
odb2 load
#
# Control File Template
LOAD REPLACE INTO TABLE ?target
```

実行ファイル

実行ファイルには、ロードを開始するための入力パラメータが含まれます。実行ファイルを実行するには、次のいずれかのコマンドを発行します。

- UNIX コマンド・シェルから SQL*Loader 実行ファイルを実行します。
  ```
  % <table>.run
  ```
- DOS シェルから BCP 実行ファイルを実行します。
  ```
  > <table>.bat
  ```

- z/OS 上の DB2 表にデータをロードするための JCL スクリプトを使用して、Load Utility 実行ファイルを実行します。必要に応じて、ジョブ・スクリプトの他の環境関連パラメータを追加してください。

  注意 WILDCARDRESOLVE パラメータの DYNAMIC 設定は、GENLOADFILES パラメータと共存できず、Oracle GoldenGate は、GENLOADFILES が指定されていると、デフォルトで IMMEDIATE を使用します。

オペレーティング・システムとデータベースの文字セットが異なる場合、または固定長の出力形式が使用されている場合、Oracle GoldenGate ではマルチバイト文字がサポートされないことに注意してください。

デフォルト なし

構文

```
GENLOADFILES [<template file>]
[CHARSET <value>]
```

引数	説明
`<template file>`	テンプレート・ファイルの完全修飾名。デフォルトのテンプレート・ファイルは、sqlldr.tpl (SQL*Loader)、bcpfmt.tpl (BCP、DTS または SSIS)、および db2cntl.tpl (z/OS 上の DB2) で、すべて Oracle GoldenGate ホーム・ディレクトリにあります。
`CHARSET <set>`	(Oracle SQL*Loader) Oracle NCHAR 列の ASCII 文字のエンコーディングを指定します。有効な値は UTF8 です。FORMATASCII の CHARSET オプションを使用する場合に必要です。 ソース表に NCHAR データ、および UTF-8 に設定された可変長文字が含まれている場合、CHARSET によって、ロードは文字長セマンテックを含めることができます。現在、Oracle SQL*Loader は、バイト長セマンテックを使用しており、文字長セマンテックとは互換性がありません。 注意 : NCHAR 列および CHAR 列の両方に 8 ビット ASCII 文字が含まれている場合、生成されるファイルには、オペレーティング・システムのネイティブの 8 ビット ASCII 文字コーディングと UTF-8 コーディングが混在して含まれているため、ロードは失敗します。

例

```
GENLOADFILES sqlldr.tpl
```

GETAPPLOPS | IGNOREAPPLOPS

適用対象 Extract

GETAPPLOPS または IGNOREAPPLOPS パラメータでは、ローカル Replicat を除くすべてのアプリケーションによって生成された DML 操作を取得または無視します。デフォルトでは、アプリケーション・データは取得されます。

これらのパラメータは、次の目的で GETREPLICATES および IGNOREREPLICATES パラメータとともに有効に活用できます。

- ローカル Replicat によって実行されたデータ操作を、Oracle GoldenGate で抽出するように構成されているビジネス・アプリケーションによって実行されたデータ操作と分離する。IGNOREAPPLOPS および GETREPLICATES を、1 つのトレイルまたはファイルに対して使用して Replicat 操作のみを包含し、GETAPPLOPS および IGNOREREPLICATES を、別のトレイルまたはファイルに対して使用してビジネス・アプリケーションの操作のみを包含します。

- カスケード構成の一部として使用する。カスケード構成では、Replicat がローカルで適用した変更をローカル Extract が取得し、別のシステムに伝播する必要があります。このケースでは、IGNOREAPPLOPS および GETREPLICATES を使用します。
- 双方向レプリケーションを行う場合にループ検出スキームの一部として使用する。デフォルトの GETAPPLOPS および IGNOREREPLICATES の組合せでは、Extract はアプリケーション・データを取得し、同一のデータベース・オブジェクトに送信された Replicat 操作を無視します。このケースでは、これらのパラメータを使用することに加え、Replicat トランザクションを特定できるように Extract を構成する必要があります。

IGNOREREPLICATES の詳細は、213 ページを参照してください。

双方向レプリケーションの構成方法の詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。

Oracle の順序に対する GETAPPLOPS の使用

Replicat によってレプリケートされる順序を取得するには、GETAPPLOPS を有効にする必要があります。Replicat は、トレース表に反映されないように、自律型トランザクションで順序の更新を発行します。順序更新は、アプリケーション操作のように見えます。

DDL 操作に対する GETAPPLEOPS の使用

GETAPPLOPS または IGNOREAPPLOPS 機能を DDL 操作に対して使用する場合は、173 ページの DDLOPTIONS パラメータの項を参照してください。

デフォルト GETAPPLOPS

構文

```
GETAPPLOPS | IGNOREAPPLOPS
```

GETDELETES | IGNOREDELETES

適用対象 Extract および Replicat

GETDELETES および IGNOREDELETES パラメータでは、Oracle GoldenGate が削除操作を処理するかどうかを制御します。これらのパラメータは表に固有です。一方のパラメータは、もう一方のパラメータが見つかるまで、それ以降のすべての TABLE または MAP 文に有効です。

デフォルト GETDELETES

構文

```
GETDELETES | IGNOREDELETES
```

GETENV

適用対象 Extract および Replicat

GETENV パラメータでは、SETENV パラメータで設定された環境変数を表示します。結果は、画面およびレポート・ファイルに出力されます。取得する変数ごとに 1 つの GETENV 文を使用します。

デフォルト なし

構文

```
GETENV (<environment variable>)
```

オプション	説明
<environment variable>	環境変数名。

例 次に、GETENV 文と返される値の例を示します。

```
GETENV (ORACLE_HOME)
ORACLE_HOME = /home/oracle/ora9/product

GETENV (ORACLE_SID)
ORACLE_SID = ora9
```

GETINSERTS | IGNOREINSERTS

適用対象 Extract および Replicat

GETINSERTS および IGNOREINSERTS パラメータでは、Oracle GoldenGate が挿入操作を処理するかどうかを制御します。これらのパラメータは表に固有です。一方のパラメータは、もう一方のパラメータが見つかるまで、それ以降のすべての TABLE または MAP 文に有効です。

デフォルト GETINSERTS

構文 GETINSERTS | IGNOREINSERTS

GETREPLICATES | IGNOREREPLICATES

適用対象 Extract

GETREPLICATES および IGNOREREPLICATES パラメータでは、Replicat が発行した DML トランザクションを、同じシステム上で同じ表を処理している Extract プロセスに取得させるか無視させるかを制御します。

これらのパラメータは、Teradata には無効です。

Replicat トランザクションの無視

デフォルトでは、Extract は IGNOREREPLICATES および GETAPPLOPS の組合せを使用します。この構成では、Extract は Oracle GoldenGate によって同期するように構成されているすべてのアプリケーション・データを取得し、すべての Replicat 操作を無視します。双方向構成では、この設定によって、Replicat が適用するデータが元のシステムにループ・バックする (そのために重複レコード・エラーが発生する) ことを防止できます。

Replicat トランザクションの取得

カスケード構成で GETREPLICATES と IGNOREAPPLOPS を使用すると、レプリケートされたデータをもう一度仲介システム上の Extract が取得し、最終的なターゲットにレプリケートできます。たとえば、データベース A をデータベース B にレプリケートし、データベース B をデータベース C にレプリケートする場合には、データベース B の Extract に対して GETREPLICATES を使用します。

注意 ただし、GETREPLICATES が有効な場合でも、TABLE または MAP 文の WHERE または FILTER 句を使用して、特定のレプリート・データを取得から除外できます。

Replicat トランザクションの識別

一部のデータベースでは、Extract に Replicat トランザクションを無視させる場合、IGNOREREPLICATES または GETREPLICATES を使用するだけでなく、Extract にこれらのトランザクションを識別させる必要があります。Replicat トランザクションの識別の詳細は、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』を参照してください。

これらのパラメータの補足情報

- チェックポイント表作成の詳細は、92 ページの ADD CHECKPOINTTABLE コマンドの項を参照してください。
- トレース表の作成と使用の詳細は、383 ページの TRACETABLE パラメータの項、および 95 ページの ADD TRACETABLE の項を参照してください。
- TRANLOGOPTIONS の詳細は、386 ページを参照してください。
- SQLEXEC の詳細は、333 ページを参照してください。
- カスケードまたは双方向構成の使用の詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。
- GETAPPLOPS および IGNOREAPPLOPS の詳細は、211 ページを参照してください。
- GETAPPLOPS または IGNOREAPPLOPS 機能を DDL 操作に対して使用する場合は、173 ページの DDLOPTIONS パラメータの項を参照してください。

デフォルト IGNOREREPLICATES

構文 GETREPLICATES | IGNOREREPLICATES

GETTRUNCATES | IGNORETRUNCATES

適用対象 Extract および Replicat

GETTRUNCATES および IGNORETRUNCATES パラメータでは、Oracle GoldenGate が表切捨て操作を処理するかどうかを制御します。デフォルトでは、切捨て操作はソースでの取得やターゲットへのレプリケーションが行われません。

GETTRUNCATES および IGNORETRUNCATES は、表に固有です。一方のパラメータは、もう一方のパラメータが見つかるまで、それ以降のすべての TABLE または MAP 文に有効です。

サポートされているデータベース

- GETTRUNCATES および IGNORETRUNCATES は、Teradata ではサポートされていません。
- GETTRUNCATES および IGNORETRUNCATES は、Oracle の Extract、CU6 以降のリリースにアップグレードされた SQL Server 2005、および Sybase よってサポートされています。
- GETTRUNCATES および IGNORETRUNCATES は、Oracle の Replicat、CU6 以降のリリースにアップグレードされた SQL Server 2005、Sybase、DB2 LUW、DB2 z/OS、MySQL、および TRUNCATE コマンドをサポートする他の ODBC ターゲットによってサポートされています。

注意 z/OS 上の DB2 データベースから取得中に、TRUNCATE を無視できません。デフォルトでは、TRUNCATE は z/OS 上の DB2 ソースから常に取得されますが、Replicat パラメータ・ファイルで IGNORETRUNCATES が使用されている場合、Replicat は TRUNCATE を無視できます。

DB2 LUW の制約事項

- DB2 LUW では TRUNCATE コマンドがサポートされていないため、Replicat は NULL (空白) ファイルから IMPORT REPLACE を実行することによって、切捨て操作をレプリケートします。

Oracle の制約事項

- Oracle GoldenGate では、Oracle TRUNCATE TABLE コマンドはサポートされていますが、TRUNCATE PARTITION はサポートされていません。完全な Oracle GoldenGate DDL レプリケーション・サポートの一部として、TRUNCATE PARTITION をレプリケートできます。
- データベースは空の表の切捨て操作のログを取らないため、これらの操作は Oracle GoldenGate に取得されません。この目的には、Oracle GoldenGate の DDL サポートを使用できます。
- データベースは空のパーティションに対する切捨て操作のログを取らないため、表に空のパーティションが含まれている場合、Oracle GoldenGate は TRUNCATE TABLE を確実に処理できません。GETTRUNCATES は、パーティション化された表で使用しないでください。空のパーティションを含む可能性がある表に対する切捨て操作の取得には、Oracle GoldenGate の DDL サポートを使用できます。

Sybase の制約事項

Oracle GoldenGate が Sybase の TRUNCATE TABLE をサポートするには、すべての表名は、特定のデータベース内のすべてのスキーマで一意である必要があります。

デフォルト IGNORETRUNCATES

構文 GETTRUNCATES | IGNORETRUNCATES

GETUPDATEAFTERS | IGNOREUPDATEAFTERS

適用対象 Extract および Replicat

GETUPDATEAFTERS および IGNOREUPDATEAFTERS パラメータでは、Oracle GoldenGate に処理されるレコードに、更新されるレコードのアフター・イメージを含めるかどうかを制御します。アフター・イメージには、更新の結果が含まれます。

これらのパラメータは表に固有です。一方のパラメータは、もう一方のパラメータが見つかるまで、それ以降のすべての TABLE または MAP 文に有効です。

デフォルト GETUPDATEAFTERS

構文 GETUPDATEAFTERS | IGNOREUPDATEAFTERS

GETUPDATEBEFORES | IGNOREUPDATEBEFORES

適用対象 Extract および Replicat

GETUPDATEBEFORES および IGNOREUPDATEBEFORES パラメータでは、Oracle GoldenGate に処理されるレコードに、更新される列のビフォア・イメージを含めるかどうかを制御します。ビフォア・イメージには、レコードが更新される前に存在していた列の詳細情報が含まれます。GETUPDATEBEFORES パラメータは、次のように使用します。

- Extract パラメータ・ファイルでは、データ・ソースからビフォア・イメージを抽出します。
- Replicat パラメータ・ファイルでは、Replicat 操作にビフォア・イメージを含めます。

ビフォア・イメージとアフター・イメージを比較して、トランザクションの最終結果を確認したり、その他の差分計算を実行したりできます。たとえば、更新前の BALANCE フィールドが 100 ドルで更新後が 120 ドルの場合、これらを比較して 20 ドルの差異を確認できます。Oracle GoldenGate の列変換ファンクションを使用して、比較および計算を実行できます。

GETUPDATEBEFORES を使用して、トランザクション履歴表を保持することもできます。差分計算の実行とトランザクション履歴の使用の詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。

- パラメータ・ファイルでビフォア・イメージを参照するには、たとえば次のように "before.<column>" 構文を使用します。

```
COLMAP (previous = before.balance, [...])
```

Oracle GoldenGate がサポートするすべてのプラットフォームで DB2 データベースに競合の検出および解決 (CDR) 機能を使用する場合、GETUPDATEBEFORES が必要です。CDR の詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。

GETUPDATEBEFORES および IGNOREUPDATEBEFORES パラメータは、表に固有です。一方のパラメータは、もう一方のパラメータが見つかるまで、それ以降のすべての TABLE または MAP 文に有効です。

デフォルト IGNOREUPDATEBEFORES

構文 GETUPDATEBEFORES | IGNOREUPDATEBEFORES

GETUPDATES | IGNOREUPDATES

適用対象 Extract および Replicat

GETUPDATES および IGNOREUPDATES パラメータでは、Oracle GoldenGate が更新操作を処理するかどうかを制御します。これらのパラメータは表に固有です。一方のパラメータは、もう一方のパラメータが見つかるまで、それ以降のすべての TABLE または MAP 文に有効です。

デフォルト GETUPDATES

構文 GETUPDATES | IGNOREUPDATES

GGSCHEMA

適用対象 GLOBALS

GGSCHEMA パラメータでは、Oracle GoldenGate による Oracle DDL の同期をサポートするデータベース・オブジェクトなど、Oracle GoldenGate が所有するデータベース・オブジェクトを含むスキーマ名を指定します。このパラメータは、Oracle データベースに有効です。

デフォルト なし

構文 GGSCHEMA <schema_name>

引数	説明
<schema_name>	DDL スキーマ名。

GROUPTRANSOPS

適用対象 Replicat

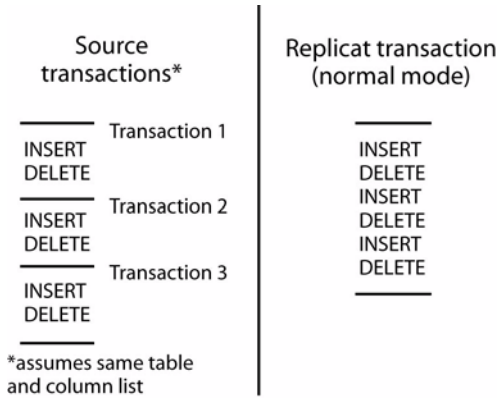
GROUPTRANSOPS パラメータでは、標準モードでの動作時に 1 つの Replicat トランザクションに含まれる SQL 操作数を制御します。Replicat トランザクションに含まれる操作数を増やすと、次のことを通じて Oracle GoldenGate のパフォーマンスが向上します。

- Replicat が実行するトランザクション数の削減。
- チェックポイント・ファイルおよび (使用されている場合) チェックポイント表への I/O アクティビティの削減。Replicat は、定期的なチェックポイントに加えて、ターゲットにトランザクションを適用するたびにチェックポイントを発行します。

Replicat は、ソース・トランザクションからトランザクションの順序で操作を蓄積し、ターゲット上で 1 つのトランザクション内のグループとして操作を適用します。GROUPTRANSOPS では、ソース・トランザクションの分割を回避するために、絶対値ではなく最小値を設定します。Replicat は、グループ内の最後のソース・トランザクションからすべての操作を受信するまで待機してから、ターゲット・トランザクションを適用します。

たとえば、トランザクション A に 500 の操作が含まれ、トランザクション B に 600 の操作が含まれている場合、GROUPTRANSOPS がデフォルトの 1,000 に設定されているとしても、Replicat トランザクションには 1,100 すべての操作が含まれることになります。逆に、トレイルに処理するデータが残っていなければ、Replicat は GROUPTRANSOPS に設定されている値に到達する前にトランザクションを適用することがあります。

図 18 Replicat 標準モード



ソース・トランザクションの境界とターゲット・トランザクションの境界の差によって、ターゲット・データの遅延が長びくことがあるため、GROUPTRANSOPS を不必要に高い値に設定しないでください。

デフォルト 1000 操作

構文

```
GROUPTRANSOPS <min transaction count>
```

引数	説明
<min transaction count>	Replicat トランザクションで適用される最小操作数。値が 1 の場合、ソース・トランザクションと同一のトランザクション境界内の操作を実行します。値は 1 以上にする必要があります。

例 GROUPTRANSOPS 2000

HANDLECOLLISIONS | NOHANDLECOLLISIONS

適用対象 Replicat

HANDLECOLLISIONS および NOHANDLECOLLISIONS パラメータでは、Replicat がターゲットに SQL を適用するときに、重複レコード・エラーおよび行方不明レコード・エラーの解決を試行させるかどうかを制御します。このようなエラーは、Oracle GoldenGate がソース表に対して行われたトランザクション変更をレプリケートしている間に、初期ロード (ソース表のデータをターゲット表にロード) が実行される場合に発生します。初期ロードが終了して Oracle GoldenGate がレプリケートされた変更を適用するとき、HANDLECOLLISIONS によって、これらの衝突に対するエラー処理ロジックが Replicat に提供されます。

HANDLECOLLISIONS および NOHANDLECOLLISIONS は、次の方法で使用できます。

- パラメータ・ファイルのルート・レベルで HANDLECOLLISIONS または NOHANDLECOLLISIONS のいずれかを使用して、すべての MAP 文に適用できます。
- 必要に応じて表グループのエラー処理を有効化または無効化する個別スイッチとして、HANDLECOLLISIONS および NOHANDLECOLLISIONS を使用できます。一方のパラメータは、もう一方のパラメータが見つかるまで、それ以降のすべての MAP 文に有効です。
- 1 つの MAP 文の中で、HANDLECOLLISIONS および NOHANDLECOLLISIONS を使用し、特定の表に対してこの機能を有効化または無効化できます。

上記のどの方法も組み合せて使用できます。このパラメータを MAP 文内で使用する場合は、他の設定よりも優先されます。スイッチとして使用する場合は、グローバル設定よりも優先されます。たとえば、グローバルに NOHANDLECOLLISIONS を設定し、MAP 文の中で HANDLECOLLISIONS を使用して特定の表にのみ有効化できます。

HANDLECOLLISIONS の動作

次の例で、HANDLECOLLISIONS の動作を説明します。

- Replicat が、Oracle GoldenGate がキーとして使用している列への更新を検出すると、処理は次のようになります。
  - 古いキーが含まれる行がターゲットに見つからない場合、トレイル内の変更レコードは挿入に変換されます。
  - 新しいキーが含まれる行がターゲットに存在する場合、Replicat は古いキーが含まれる行を削除し (更新が正常に実行された場合は、このような行は存在しません)、新しいキーが含まれる行は、トレイル値が現在の値を置き換える上書きとして更新されます。

  このロジックでは、デフォルトで、または強制的に (Oracle データベースの ADD TRANDATA の COLS オプションを使用するなど)、表内のすべての列 (変更された列のみではない) をトランザクション・ログに書き込む必要があります。(「行方不明の列値を回避するために考えられる解決策」も参照してください。)

- Replicat が重複レコード・エラーを検出すると、初期ロードで適用された静的レコードは、トレイル内の変更レコードによって上書きされます。操作の観点からすると、重複レコード・エラーは、無視するよりも変更を適用するほうが安全です。
- Replicat は、キー列に影響しない更新または削除操作中に行方不明レコード・エラーを検出すると、トレイル内の変更レコードを破棄します。このようなエラーは、表データが初期ロード・プロセスによって抽出される前に、ソース表でレコードが変更および削除されている場合に発生します。次に例を示します。

1. アプリケーションがソース表のレコード A1 を更新します。
2. Extract がこの更新を抽出します。
3. アプリケーションがソース表 1 のレコード A を削除します。
4. Extract がこの削除を抽出します。
5. Oracle GoldenGate が、レコード A が存在しないソース表 1 から初期ロード・データを抽出します。
6. Oracle GoldenGate がレコード A なしで初期ロードを適用します。
7. Replicat がレコード A の更新の適用を試みます。
8. データベースは " レコード行方不明 " エラーを返します。
9. Replicat がレコード A の削除の適用を試みます。
10. データベースは " レコード行方不明 " エラーを返します。

初期ロード中に取得されたトランザクション変更がターゲット表に適用されたら、HANDLECOLLLIONS を無効化し、その後のエラーが Replicat によって自動的に処理されないようにします。初期同期以降に発生するエラーは異常な状態を意味するため、解決方法を特定できる担当者によって検証される必要があります。たとえば、行方不明エラーは、ソース表に存在するデータがターゲット・システムから誤って削除されたことが原因の可能性があります。

HANDLECOLLISIONS は、次の方法で無効化できます。

- Replicat を停止し、Replicat パラメータ・ファイルから HANDLECOLLISIONS を削除します (ターゲットに遅延が生じる可能性があります)。または、パラメータ・ファイルを編集して、エラー処理を無効化する MAP 文の前に NOHANDLECOLLISIONS を追加できます。
- Replicat の実行中に GGSCI を実行し、SEND REPLICAT コマンドと該当の表に対する NOHANDLECOLLISIONS オプションを使用します (61 ページを参照してください)。SEND REPLICAT を使用する場合は、HANDLECOLLISIONS が再度有効化されないように、Replicat の実行を次に開始する前に、パラメータ・ファイルから HANDLECOLLISIONS を削除するか、NOHANDLECOLLISIONS パラメータを追加してください。

行方不明の列値を回避するために考えられる解決策

データベースがデフォルトでソース表のすべての列値を記録しないと、Replicat が主キー更新を挿入に変換しようとするときにターゲット表に NOT NULL 制約がある場合、エラーが発生します。次の方法で、このシナリオに対処できます。

- Extract パラメータ・ファイルで NOCOMPRESSUPDATES パラメータを使用して、表のすべての列をトレイルに送信し、すべての列値を記録するようにデータベースを構成します。デフォルトでは、Extract は主キーおよび変更された列のみをトレイルに書き込みます。この方法は、操作が実行された時点で現在の値が書き込まれ、フェッチする必要がなくなるため、最も安全です。

- Extract パラメータ・ファイルで FETCHOPTIONS パラメータを FETCHPKUPDATECOLS オプションとともに使用します。この構成では、Extract に、キー列がソース上で更新されたときに使用できない列をフェッチさせます。フェッチは*現在の*値で、特定の更新時の値であるとはかぎらないため、データ整合性の問題が発生する可能性があります。詳細情報、および失敗したフェッチを処理する追加のフェッチ・オプションは、199 ページを参照してください。
- フェッチを回避するには、HANDLECOLLISIONS を _ALLOWPKMISSINGROWCOLLISIONS とともに使用して、更新を挿入に変換するかわりに、更新をスキップします。この構成では、特定の条件でデータ整合性の問題が発生する可能性もあります。詳細は、「キーの更新から挿入への変換の防止」を参照してください。

キーの更新から挿入への変換の防止

場合によっては、ターゲット行が存在しない場合に、キー列を更新する操作を挿入に変換することは適切ではありません。このような場合、_ALLOWPKMISSINGROWCOLLISIONS オプションを使用して、挿入として適用するかわりに、Replicat に操作を*スキップ*させることができます。

次の例では、そのようなケースを示します。このシナリオでは、デフォルトの HANDLECOLLISIONS ロジックを使用して Oracle GoldenGate レプリケーションのインスタンス化を実行して、Replicat が更新を挿入に変換しようとしたときに列値が行方不明の場合にどのような状況が発生するかを示しています。

"s" という名前のソース表およびターゲット表

```
f1  f2                  f3  f4
1   10-01-2011 11:30:45 1   1
2   10-02-2011 14:15:20 2   2
3   10-03-2011 15:12:55 3   3
```

- すべての列は NOT NULL です。
- f1 は主キーです。
- f2 は、レコードが変更されるたびに自動的に更新される日付フィールドです。
- KEYCOLS は、f1 および f2 をキーとして使用するように Oracle GoldenGate に指示するために、パラメータ・ファイルで使用されます。
- 列 f2 を記録するために、ADD TRANDATA が適宜発行されました。(列 f1 は主キーであるため、自動的に記録されます。)

DML の一連のイベント

1. Extract を起動して、進行中のトランザクションを取得します。
2. 次のように、表に UPDATE を実行します。

   ```
   update s set f3=3 where f1=2;
   ```

   この操作では、列 f2 は現在の日時で自動的に更新されます。Oracle GoldenGate では、これはキー更新とみなされます。

   現在、行は次のようになります。

   ```
   2   10-20-2011 08:01:32  3   3
   ```

3. 同じ行の DELETE を実行します。

```
delete s where f1=2
```

現在、表には次の行が含まれています。

```
f1  f2                  f3  f4
1   10-01-2011 11:30:45 1   1
3   10-03-2011 15:12:55 3   3
```

4. HANDLECOLLISIONS を使用して、ターゲットへのソース・データのエクスポート / インポートを実行して、行方不明の行や重複する行を処理します。
5. レプリケートされた更新 (update s set f3=3 where f1=2) は、Replicat によってトレイルから適用される最初の操作です。インポート / エクスポートが実行される前にソースから行が削除されたため、これは失敗します。
6. キー列 (f2 の日時の列) を更新する操作を行う HANDLECOLLISIONS ロジックに従って、Replicat は UPDATE を INSERT に変換します。
7. すべての列値がトレイルで使用できる場合、新しい挿入は成功します。さらに、レプリケートされた削除 (delete s where f1=2) によって行が再度削除されるため、行がソース上で削除されていても、不整合は発生しません。ただし、この例には 2 つの問題があります。
   - 列 f1、f2、および、変更された f3 の値のみが記録されます。f4 の値は記録されないため、この値は挿入されません。
   - すべての列には NOT NULL 制約があります。

f4 の値が行方不明であることにより、挿入は失敗します。_ALLOWPKMISSINGROWCOLLISIONS を使用すると、Replicat は、UPDATE を挿入に変換するかわりにスキップします。これにより、行が存在しないため後続の DELETE は失敗し、そのため、Replicat はデフォルトの HANDLECOLLISIONS ロジックの一部として DELETE レコードをスキップします。現在、データはソースのデータと一貫性があります。

_ALLOWPKMISSINGROWCOLLISIONS からのメッセージ

データ損失のリスクにより、_ALLOWPKMISSINGROWCOLLISIONS を使用する場合は警告が発行されます。警告は次のテキストのようになります。

```
Using _ALLOWPKMISSINGROWCOLLISIONS may cause data corruption under certain
conditions.
```

キーへの UPDATE に、挿入への変換のための完全なアフター・イメージが含まれていない場合、次の警告メッセージも発行されます。

```
A complete after image is not available in SOURCE.x, at RBA 123, in file
.\dirdat\aa000000, while inserting a row into TARGET.x due to a missing target row
for a key update operation. NOCOMPRESSUPDATES or FETCHOPTIONS FETCHPKUPDATECOLS
may be specified in the EXTRACT parameter file to include a complete image for key
update operations.
```

デフォルト なし

構文 `HANDLECOLLISIONS | NOHANDLECOLLISIONS [_ALLOWPKMISSINGROWCOLLISIONS]`

引数	説明
_ALLOWPKMISSINGROWCOLLISIONS	HANDLECOLLISIONS を _ALLOWPKMISSINGROWCOLLISIONS とともに使用すると、対応するターゲット行が存在しない場合に、主キーの UPDATE 操作をスキップします。注意 : スキップ操作によりデータの破損が発生することがあります。このトピックの説明を参照してください。

例 1 次の例では、パラメータ・ファイル内のすべての MAP 文に対して HANDLECOLLISIONS を有効にします。

```
HANDLECOLLISIONS
MAP hr.emp, TARGET hr.emp;
MAP hr.job_hist, TARGET hr.job_hist;
MAP hr.dep, TARGET hr.dep;
MAP hr.country, TARGET hr.country;
```

例 2 次の例では、パラメータ・ファイル内の一部の MAP 文に対してのみ HANDLECOLLISIONS を有効にし、それ以外に対しては無効にします。

```
HANDLECOLLISIONS
MAP hr.emp, TARGET hr.emp;
MAP hr.job_hist, TARGET hr.job_hist;
NOHANDLECOLLISIONS
MAP hr.dep, TARGET hr.dep;
MAP hr.country, TARGET hr.country;
```

例 3 次に、グローバルおよび MAP レベルを組み合せた使用例を示します。指定された表では、MAP レベルの指定がグローバル指定よりも優先されます。

```
HANDLECOLLISIONS
MAP hr.emp, TARGET hr.emp;
MAP hr.job_hist, TARGET hr.job_hist;
MAP hr.dep, TARGET hr.dep, NOHANDLECOLLISIONS;
MAP hr.country, TARGET hr.country, NOHANDLECOLLISIONS;
```

HANDLETPKUPDATE

適用対象 Replicat

HANDLETPKUPDATE パラメータでは、主キーへの更新の結果、一時的な重複が発生するときの制約エラーを防止します。これは Oracle パラメータで、ターゲット・データベースが Oracle リリース 11.2.0.2 より前のリリースである場合に必要です。ターゲットの Oracle データベースがリリース 11.2.0.2 以降である場合、一時的な主キーの重複は、HANDLETPKUPDATE を必要とせずに、自動的に処理されます。

更新によって、トランザクション内の複数の行の主キーに影響を与える場合に、一時的な主キーの重複が発生します。この種類の文は、通常、"SET x = *x*+*n*" の式、または、新しい値が既存の値と同じになるように値をシフトするその他の操作を使用します。

次に、この状況を発生させる可能性がある順序値の変更の例を示します。この例では、表を "ITEM"、主キー列名を "CODE"、この表の行の現在のキーの値を 1、2、3 とします。

```
update item set code = 2 where code = 1;
update item set code = 3 where code = 2;
update item set code = 4 where code = 3;
```

この例では、最初の更新がターゲットに適用されると、表にすでに主キー値 2 が存在するため、エラーが発生します。ターゲット・トランザクションは、制約違反エラーを返します。デフォルトでは、Replicat はこうした違反を検出または処理せずに異常終了します。

HANDLETPKUPDATE を使用すると、ターゲット表に DEFERRABLE INITIALLY IMMEDIATE が制約として作成されます。ターゲットの制約を DEFERRABLE にできない場合、Replicat は HANDLECOLLISIONS および REPERROR パラメータに指定された既存のルールに従ってエラーを処理するか、異常終了します。

デフォルト 一時主キー更新で異常終了する

構文 HANDLETPKUPDATE

INCLUDE

適用対象 Extract および Replicat

INCLUDE パラメータでは、パラメータ・ファイルにマクロ・ライブラリを含めます。

デフォルト なし

構文 `INCLUDE <path name>`

引数	説明
`<path name>`	ライブラリ・ファイルへの相対パスまたは完全パス。

例 次の例では、マクロ・ライブラリ mdatelib.mac を含めます。

```
INCLUDE /ggs/dirprm/mdatelib.mac
```

INSERTAPPEND | NOINSERTAPPEND

適用対象 Replicat

INSERTAPPEND および NOINSERTAPPEND パラメータでは、Replicat が Oracle ターゲット表に INSERT 操作を適用するときに APPEND ヒントを使用するかどうかを制御します。これらのパラメータは、Oracle データベースにのみ有効です。

これらのパラメータは 2 つの方法で使用でき、パラメータ・ファイルのルートで単独パラメータとして使用する場合は、一方のパラメータは、もう一方のパラメータが見つかるまで、それ以降のすべての TABLE または MAP 文に有効です。1 つの MAP 文内で使用する場合は、この MAP 文よりも先に指定されているすべての単独の INSERTAPPEND または NOINSERTAPPEND エントリよりも優先されます。

INSERTAPPEND を指定すると、Oracle ターゲット表に INSERT 操作を適用するときに、Replicat は APPEND_VALUES ヒントを使用します。レプリケートされるトランザクションが大きく、同一の表への複数の挿入が含まれている場合、ヒントの使用は適切なパフォーマンスの向上策です。トランザクションが小さい場合に INSERTAPPEND を使用すると、パフォーマンスが低下することがあります。APPEND ヒント使用の詳細は、Oracle のマニュアルを参照してください。

INSERTAPPEND を使用する場合は、BATCHSQL パラメータを使用する必要があります。BATCHSQL を使用しない場合、Replicat は異常終了します。

MAP 構文の詳細は、233 ページを参照してください。

デフォルト NOINSERTAPPEND

構文 INSERTAPPEND | NOINSERTAPPEND

例 次の Replicat パラメータ・ファイルからの引用では、inventory 表を除き、fin スキーマのすべての表に INSERTAPPEND を使用する方法を示します。

```
BATCHSQL
INSERTAPPEND
MAP fin.*, TARGET fin.*;
MAPEXCLUDE fin.inventory;
NOINSERTAPPEND
MAP fin.inventory, TARGET fin.inventory;
```

INSERTALLRECORDS

適用対象 Replicat

INSERTALLRECORDS パラメータでは、レコードの現在のバージョンのみではなく、ターゲット・レコードに行われたすべての操作のレコードを保持します。INSERTALLRECORDS を使用すると、Replicat はレコードに対するすべての変更操作を新しいレコードとしてデータベースに挿入します。最初の挿入、その後の更新および削除は、ポイントインタイム・スナップショットとして保持されます。

履歴データと特別なトランザクション情報を組み合せることで、より有益なターゲット・レポート・データベースを作成できます。トランザクション履歴表の保持の詳細は、『*Oracle GoldenGate Windows and UNIX 管理者ガイド*』を参照してください。

このパラメータは、MAP 文内でも使用できます。233 ページを参照してください。

デフォルト なし

構文 INSERTALLRECORDS

INSERTDELETES | NOINSERTDELETES

適用対象 Replicat

INSERTDELETES および NOINSERTDELETES パラメータでは、Oracle GoldenGate がソースの削除操作をターゲット・データベース上で挿入操作に変換するかどうかを制御します。これらのパラメータは表に固有です。一方のパラメータは、もう一方のパラメータが見つかるまで、それ以降のすべての MAP 文に有効です。

INSERTDELETES を使用する場合、Extract が削除を圧縮しないように、NOCOMPRESSDELETES パラメータを使用します。

デフォルト NOINSERTDELETES

構文 INSERTDELETES | NOINSERTDELETES

INSERTMISSINGUPDATES | NOINSERTMISSINGUPDATES

適用対象 Replicat

INSERTMISSINGUPDATES および NOINSERTMISSINGUPDATES パラメータでは、ターゲット・レコードが存在しない場合に Oracle GoldenGate がソース・レコードに基づいてレコードを挿入するかどうかを制御します。

INSERTMISSINGUPDATES は、行方不明のレコードに対する更新を挿入しますが、ソース・データベースが非圧縮更新を使用している (つまりすべての列値が記録される) 場合にのみ使用するようにしてください。ターゲット・データベースが、行方不明の列の値に NULL 値の使用を許可する場合には、圧縮更新を使用するデータベースとともに使用できます。

デフォルトの NOINSERTMISSINGUPDATES が有効な場合は、行方不明レコードによってエラーが発生し、REPERROR の設定に応じて、トランザクションは異常終了することがあります。

INSERTMISSINGUPDATES および NOINSERTMISSINGUPDATES パラメータは、表に固有です。一方のパラメータは、もう一方のパラメータが見つかるまで、それ以降のすべての MAP 文に有効です。

デフォルト NOINSERTMISSINGUPDATES

構文

```
INSERTMISSINGUPDATES | NOINSERTMISSINGUPDATES
```

INSERTUPDATES | NOINSERTUPDATES

適用対象 Replicat

INSERTUPDATES および NOINSERTUPDATES パラメータでは、Oracle GoldenGate が非圧縮更新操作を挿入操作に変換するかどうかを制御します。圧縮されない更新の場合、データベースは、デフォルトまたはサプリメンタル・ロギングのいずれかを使用して、すべての列値を記録する必要があります。

これらのパラメータは表に固有です。一方のパラメータは、もう一方のパラメータが見つかるまで、それ以降のすべての MAP 文に有効です。

INSERTUPDATES を使用する場合に、更新が Extract によって圧縮されていないことを確認するには、NOCOMPRESSUPDATES パラメータを使用します。

デフォルト NOINSERTUPDATES

構文

```
INSERTUPDATES | NOINSERTUPDATES
```

LAGCRITICAL

適用対象 Manager

LAGCRITICALSECONDS、LAGCRITICALMINUTES、または LAGCRITICALHOURS パラメータでは、クリティカルとみなすラグしきい値を指定し、しきい値に達したときにエラー・ログに警告メッセージを書き込みます。このパラメータは、ローカル・システムの Extract および Replicat プロセスに影響します。

デフォルト ラグ情報をレポートしない

構文

```
LAGCRITICALSECONDS <seconds> |
LAGCRITICALMINUTES <minutes> |
LAGCRITICALHOURS <hours>
```

引数	説明
<seconds>	ラグしきい値 (秒)。
<minutes >	ラグしきい値 (分)。
<hours>	ラグしきい値 (時間)。

例

```
LAGCRITICALSECONDS 60
```

LAGINFO

適用対象 Manager

LAGINFOSECONDS、LAGINFOMINUTES または LAGINFOHOURS パラメータでは、ラグしきい値を指定します。ラグが指定した値を超えると、Oracle GoldenGate はエラー・ログにラグ情報をレポートします。ラグが LAGCRITICAL パラメータで指定した値を超えると、Manager はこのラグをクリティカルとしてレポートし、それ以外の場合は情報メッセージとしてレポートします。値ゼロ (0) を指定すると、LAGREPORTMINUTES または LAGREPORTHOURS パラメータで指定した間隔でメッセージがレポートされます。

デフォルト ラグ情報をレポートしない

構文

```
LAGINFOSECONDS <seconds> |
LAGINFOMINUTES <minutes> |
LAGINFOHOURS <hours>
```

引数	説明
<seconds>	Oracle GoldenGate がラグ情報をレポートするしきい値 (秒)。
<minutes >	Oracle GoldenGate がラグ情報をレポートするしきい値 (分)。
<hours>	Oracle GoldenGate がラグ情報をレポートするしきい値 (時間)。

例 この例では、Oracle GoldenGate は 1 時間を超えるとラグをレポートします。

```
LAGINFOHOURS 1
```

LAGREPORT

適用対象 Manager

LAGREPORTMINUTES または LAGREPORTHOURS パラメータでは、Manager が Extract および Replicat のラグを確認する間隔を指定します。

デフォルト なし

構文 LAGREPORTMINUTES <minutes> | LAGREPORTHOURS <hours>

引数	説明
<minutes>	ラグ情報を確認する間隔 (分)。
<hours>	ラグ情報を確認する間隔 (時間)。

例 LAGREPORTHOURS 1

LIST | NOLIST

適用対象 Extract および Replicat

LIST および NOLIST パラメータでは、レポート・ファイルにマクロ・ライブラリのマクロをリストするかどうかを制御します。レポートへのリストを有効化または無効化するには、パラメータ・ファイル内かマクロ・ライブラリ・ファイル内に LIST または NOLIST パラメータを配置します。NOLIST を使用すると、レポート・ファイルのサイズが小さくなります。

デフォルト LIST

構文 LIST | NOLIST

例 次の例では、NOLIST によって、hugelib マクロ・ライブラリのマクロをレポートへのリストから除外します。INCLUDE 文の後に LIST を使用し、それ以降のマクロを通常のリスト設定に戻します。

```
NOLIST
INCLUDE /ggs/hugelib.mac
LIST
```

LOBMEMORY

適用対象 z/OS 上の DB2 および NonStop SQL/MX の Extract と Replicat

LOBMEMORY パラメータを使用して、LOB を含むトランザクションのキャッシュに使用するメモリーおよび一時ディスク領域の量を制御します。Oracle GoldenGate はコミットされたトランザクションのみをターゲット・データベースに適用するため、コミットまたはロールバックのインジケータを受信するまで LOB データを保持するための十分なシステム・メモリーが必要です。

このパラメータは、z/OS 上の DB2 データベースおよび NonStop SQL/MX データベースとともに使用します。他のすべてのデータベースでは、CACHEMGR パラメータを使用します。

LOBMEMORY を使用したメモリー管理について

LOBMEMORY を使用して、Oracle GoldenGate の LOB トランザクション用のキャッシュ・サイズを調整し、キャッシュ・サイズを超えるデータを保持するディスク上の一時的な場所を定義できます。合計キャッシュ・サイズ、トランザクション当たりのメモリー・サイズ、初期および増分メモリー割当て、ディスク記憶域領域を定義するためのオプションを使用できます。

LOB トランザクションは、RAM によって指定されたメモリー・プールに追加され、TRANSRAM の値に達したときにディスクにフラッシュされます。初期メモリー量は、INITTRANSRAM に基づいて各トランザクションに割り当てられ、必要に応じて、RAMINCREMENT で指定された量ずつ、TRANSRAM で設定された最大値まで増分されます。したがって、TRANSRAM には (INITTRANSRAM + RAMINCREMENT) の合計によって割り切れる値を指定する必要があります。

デフォルト オプションのデフォルトを参照してください。

構文

```
LOBMEMORY
[RAM <size>]
[TRANSRAM <size>]
[TRANSALLSOURCES <size>]
[INITTRANSRAM <size>]
[RAMINCREMENT <size>]
[DIRECTORY (<directory name>, <max dir size>, <max file size>)]
```

オプション	説明
`RAM <size>`	キャッシュされたすべての LOB トランザクションが使用する合計メモリー量を指定します。デフォルトは 200MB です。この値は、バイト、または次のいずれかの形式の GB、MB または KB で指定できます。 GB | MB | KB | G | M | K | gb | mb | kb | g | m | k
`TRANSRAM <size>`	1 つの LOB トランザクションが使用する合計メモリー量を指定します。デフォルトは 50MB です。この値は、バイト、または次のいずれかの形式の GB、MB または KB で指定できます。 GB | MB | KB | G | M | K | gb | mb | kb | g | m | k TRANSRAM は、最適な成果を得るために、INITTRANSRAM および RAMINCREMENT 両方の値によって割り切れる値に設定する必要があります。
`TRANSALLSOURCES <size>`	1 つの LOB トランザクションが利用するメモリーおよびディスク領域の量の合計を指定します。デフォルトは、使用可能なメモリー (メモリーおよびディスク) 合計の 50% です。この値は、バイト、または次のいずれかの形式の GB、MB または KB で指定できます。 GB | MB | KB | G | M | K | gb | mb | kb | g | m | k
`INITTRANSRAM <size>`	1 つの LOB トランザクションに割り当てる初期メモリー量を指定します。デフォルトは 500KB です。この値は、バイト、または次のいずれかの形式の GB、MB または KB で指定できます。 GB | MB | KB | G | M | K | gb | mb | kb | g | m | k
`RAMINCREMENT <size>`	LOB トランザクションがさらに多くのメモリーを必要とするときに増分するメモリー量を指定します。デフォルトは 500KB です。この値は、バイト、または次のいずれかの形式の GB、MB または KB で指定できます。 GB | MB | KB | G | M | K | gb | mb | kb | g | m |

オプション	説明
DIRECTORY (<directory name>, <max dir size>, <max file size>)	LOB トランザクション・データのサイズが TRANSRAM で指定した最大値を超えたときにこのデータが使用する一時ディスク記憶域を指定します。DIRECTORY は、複数回指定できます。 ◆ <directory> は、ディレクトリの完全修飾名です。デフォルトは、Oracle GoldenGate ディレクトリの dirtmp サブディレクトリです。 ◆ <max dir size> は、ディレクトリ内のすべてのファイルの最大サイズです。デフォルトは 2GB です。指定した領域が使用できない場合は、使用可能なディスク領域の 75% が使用されます。 ◆ <max dir size> は、ディレクトリ内の各ファイルの最大サイズです。デフォルトは 200MB です。 この値は、バイト、または次のいずれかの形式の GB、MB または KB で指定できます。 GB | MB | KB | G | M | K | gb | mb | kb | g | m | k ディレクトリ・サイズおよびファイル・サイズは、RAM で指定したメモリー・サイズより大きい必要があります。 ファイル名は、次のフォーマットを使用します。 <group>_blob_00001.mem または <PID>_blob_00001.mem グループ名は、オンライン処理で使用されます。システム・プロセス ID 番号 (PID) は、SPECIALRUN パラメータで指定したワンタイム実行で使用されます。 スレッド Extract のフォーマットは次のようになります (データベースによって異なります)。 <group>_<thread #>_00001.mem

例 1 次の例では、データをディスクにフラッシュする前に、各トランザクションに 10 回のメモリー追加を許可します (INITTRANSRAM で指定された初期割当て 1 回と、RAMINCREMENT で許可された 9 回)。

```
LOBMEMORY DIRECTORY(c:\test\dirtmp, 3000000000,
300000000), RAM 8000K, TRANSRAM 1000K, INITTRANSRAM 100K,
RAMINCREMENT 100K
```

例 2 次の例は上記の例と同じですが、2 つ目のディレクトリも追加します。

```
LOBMEMORY DIRECTORY(c:\test\dirtmp, 3000000000,
300000000), DIRECTORY (c:\test\dirtmp2, 1000000000,
5000000), RAM 8000K, TRANSRAM 1000K, INITTRANSRAM 100K,
RAMINCREMENT 100K
```

注意 **上記の例では、スペースの制約によってパラメータ指定が複数行にわたっています。実際のパラメータ・ファイルでは、パラメータ指定が複数行にわたる場合は、各行の末尾にアンパサンド (&) を含める必要があります。**

MACRO

適用対象 Extract および Replicat

MACRO パラメータでは、Oracle GoldenGate マクロを作成します。マクロ使用の詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。

デフォルト なし

構文 次に示す順序で使用する必要があります。

```
MACRO <macrochar><macro name>
PARAMS (<macrochar><paramname> [, ...])
BEGIN
<macro body>
END;
```

引数	説明

`<macrochar>`	マクロ文字。マクロおよびパラメータ名は、マクロ文字で開始する必要があります。パラメータ・ファイル内でマクロ文字で始まる要素はすべて、マクロまたはマクロ・パラメータとみなされます。
	デフォルトのマクロ文字は、次の例に示すようにシャープ (#) 文字です。
	`MACRO #macro1`
	`PARAMS (#param1, #param2)`
	マクロ文字を変更するには、MACROCHAR パラメータを使用します。

`<macro name>`	マクロ名。マクロ名は、英数字を使用した 1 単語 (アンダースコアは許可されます) である必要があり、大 / 小文字が区別されません。パラメータ・ファイル内の各マクロ名は、一意である必要があります。引用符は使用しないでください (使用すると、マクロ名はテキストとして扱われ、無視されます)。
`<paramname>`	マクロに対するパラメータを記述します。パラメータ名は大 / 小文字が区別されません。引用符は使用しないでください (使用すると、パラメータ名はテキストとして扱われ、無視されます)。パラメータ句はオプションです。PARAMS 文の最大サイズは 9999 バイトで、99 以下のパラメータを含めることができます。 マクロで使用するすべてのパラメータは、PARAMS 文で宣言する必要があり、マクロを呼び出すときは、呼出し文に各パラメータの値が含まれている必要があります。 例 1 に、パラメータを取るマクロを示します。 パラメータなしのマクロ (頻繁に使用するコマンドを含むマクロなど) も作成できます。例 2 を参照してください。 他のマクロをパラメータとして使用することもできます。例 3 を参照してください。
`BEGIN`	マクロ本文を開始します。マクロ本文の前に指定する必要があります。

引数	説明

<macro body>	マクロ名本文。マクロ本文の最大サイズは 99999 バイトです。マクロ本文には、次のいずれかのタイプの文を含めることができます。
	◆ 次のような単純なパラメータ文 :
	`COL1 = COL2`
	◆ 次のような複雑な文 :
	`COL1 = #val2`
	◆ 次のような他のマクロの呼び出し :
	`#colmap(COL1, #sourcecol)`

END	マクロ定義を終了します。

例 1 次の例では、パラメータを取るマクロを定義します。

```
MACRO #make_date
PARAMS (#year, #month, #day)
BEGIN
@DATE("YYYY-MM-DD", "CC", @IF(#year < 50, 20, 19),
"YY", #year, "MM", #month, "DD", #day)
END;
```

例 2 次の例では、パラメータを必要としないマクロを定義します。

```
MACRO #option_defaults
BEGIN
GETINSERTS
GETUPDATES
GETDELETES
INSERTDELETES
END;
```

例 3 次の例では、#make_date という名前の他のマクロを呼び出す #assign_date という名前のマクロを定義します。

```
MACRO #assign_date
PARAMS (#target_col, #year, #month, #day)
BEGIN
#target_col = #make_date (#year, #month, #day)
END;
```

MACROCHAR

適用対象 Extract および Replicat

MACROCHAR パラメータでは、マクロ文字を # 以外の文字に変更します。指定したマクロ文字で始まるパラメータ・ファイル内の要素はすべて、マクロまたはマクロ・パラメータとみなされます。

たとえば、表名に # 文字が含まれている場合は、マクロ文字を変更する必要があります。

MACROCHAR は一度だけ指定でき、最初のマクロ文の前に指定する必要があります。

デフォルト # (シャープ記号)

構文 `MACROCHAR <character>`

引数	説明
`<character>`	マクロ文字として使用する文字。有効なマクロおよびパラメータ文字は英数字で、アンダースコア文字 (_) も含むことができます。

例 次の例では、$ をマクロ文字として定義します。

```
MACROCHAR $
MACRO $mymac
PARAMS ($p1)
BEGIN
col = $p1
END;
```

Extract 用 MAP

適用対象 Extract

Extract 用の MAP パラメータは、クラシック・キャプチャ・モードで動作しているとき、および、ALTID コンポーネントを使用してオブジェクト ID をオブジェクト名にマップするときに使用します。ALTID では、Extract が接続先データベース以外のデータベースによって生成された Oracle トランザクション・ログを取得する場合に、正しいオブジェクト ID を指定します。この構成は、Oracle GoldenGate が本番環境の (ソース) データベースへの直接接続を許可されていない状況で、本番環境のトランザクションを取得する必要があるときに必要です。

こうしたケースでは、Extract はライブ・スタンバイ・データベースまたはその他の複製データベースに接続しますが、本番環境のデータベースから送信されたトランザクション・ログを読み取ります。代替データベースのカタログへの問合せによって、Extract はトランザクション・データを有効な SQL 文に展開するために必要なメタデータを取得できますが、表のローカル・オブジェクト ID が本番環境のデータベース (およびトランザクション・ログ) のものとは異なるため、この問合せで取得したオブジェクト ID は使用できません。MAP 文と ALTID を使用して、各表名を本番環境 (ソース) のオブジェクト ID に手動でマップする必要があります。

MAP と ALTID の使用方法

- 取得する表ごとに1つのMAP文とALTIDを作成します。ALTIDが含まれているMAPパラメータでは、表名のワイルドカード指定は許可されていません。
- 表に対して、データのフィルタリングや操作などの他の処理を指定するときは、TABLE 文も作成する必要があります。他の処理を指定しない場合は TABLE は必要ありません。
- Replicat パラメータ・ファイルでは、通常どおり標準の Replicat MAP 文を使用します。Extract 用 MAP は、ソース表をターゲット表にマップするために必要な Replicat 用 MAP のかわりには使用できません。
- ALTID を使用するときは、DDL 取得およびレプリケーションはサポートされません。

デフォルト なし

構文 `MAP <table spec>, ALTID <object ID> [, object ID>]`

コンポーネント	説明
<object ID>	本番環境 (ソース) データベース内での表のオブジェクト ID。 表がパーティション化されている場合は、レプリケートするパーティションのオブジェクト ID をコンマで区切ってリストできます。

例 `map QASOURCE.T2, altid 75740;`

例 `map QASOURCE.T_P1, altid 75257,75258;`

Replicat 用 MAP

適用対象 Replicat

Replicat 用の MAP パラメータでは、1 つ以上のソースおよびターゲット・オブジェクト間の関係を確立します。Oracle GoldenGate で同期を取るすべてのターゲット・オブジェクトは、このパラメータを使用してソースにマッピングする必要があります。

注意 MAP でマッピングするソース・オブジェクトを取得するには、ソース・システム上の Extract パラメータ・ファイルで TABLE パラメータ文を使用します。

使用の制限

表に対しては、すべての MAP オプションを使用できます。次のことを実行できます。

- 表の行を選択およびフィルタする
- 表の列をマップする
- データを変換する
- キー列を指定する
- ストアド・プロシージャおよび問合せを実行する
- 例外およびエラー処理を指定する
- 表へのすべての操作を挿入として適用する
- ユーザー・イグジットにパラメータを渡す

表以外のオブジェクトに対しては、MAP ではソース・オブジェクトとターゲット・オブジェクトのマップのみを行い、EXCEPTIONSONLY および REPERROR オプションを使用して処理エラーを処理します。これらのオブジェクトに対しては、これ以外のどの MAP オプションも使用しないでください。

デフォルト なし

構文

```
MAP <table spec>, TARGET <table spec>
[, COLMAP (<column mapping expression>)]
[, DEF <definitions template>]
[, COMPARECOLS (
    {ON UPDATE | ON DELETE}
    {ALL |
    KEY |
    KEYINCLUDING (<col>[,...]) |
    ALLEXCLUDING (<col>[,...]) }
    [,...]
    )]
[, EVENTACTIONS (<action>)]
[, EXCEPTIONSONLY]
[, EXITPARAM "<parameter string>"]
[, FILTER (<filter specification>)]
[, HANDLECOLLISIONS | NOHANDLECOLLISIONS]
[, INSERTALLRECORDS]
[, INSERTAPPEND | NOINSERTAPPEND]
[, KEYCOLS (<column specification>)]
[, MAPEXCEPTION (TARGET <object spec> [, <exception_MAP_options>])]
[, REPERROR (<error> , <response>)]
[, RESOLVECONFLICT (
    {INSERTROWEXISTS |
    UPDATEROWEXISTS |
    UPDATEROWMISSING |
    DELETEROWEXISTS |
    DELETEROWMISSING}
    ({DEFAULT | <resolution name>},
    {USEMAX (<res_col>) |
    USEMIN (<res_col>) |
    USEDELTA |
    DISCARD |
    OVERWRITE |
    IGNORE})
[, COLS (<col>[,...])]
)]
[RESOLVECONFLICT (, ...)]
[, SQLEXEC (<SQL specification>)]
[, TARGETDEF <definitions template>]
[, TRIMSPACES | NOTRIMSPACES]
[, TRIMVARSPACES | NOTRIMVARSPACES]
[, WHERE (<where clause>)]
;
```

表 36　MAP 構文コンポーネントの概要

コンポーネント	説明
MAP	ソース・オブジェクトを指定します。
TARGET	ターゲット・オブジェクトを指定します。
DEF	ソース定義テンプレートを指定します。

表 36　MAP 構文コンポーネントの概要 (続き)

コンポーネント	説明
TARGETDEF	ターゲット定義テンプレートを指定します。
COLMAP	異なるソース列およびターゲット列間でレコードをマップします。
COMPARECOLS	競合の検出および解決に使用する列を指定します。
EVENTACTIONS	指定されているフィルタ・ルールを満たすレコードに基づいて、アクションをトリガーします。
EXCEPTIONSONLY	MAP 文が例外 MAP 文であることを指定します。
EXITPARAM	リテラル文字列形式でパラメータをユーザー・イグジットに渡します。
FILTER	数値演算子に基づいてレコードを選択します。FILTER は、WHERE よりも柔軟に使用できます。
HANDLECOLLISIONS | NOHANDLECOLLISIONS	初期ロード・プロセスでターゲット表に適用された変更の結果と、変更同期グループによって適用された変更を調整します。
INSERTALLRECORDS	すべての行変更を挿入として適用します。
INSERTAPPEND | NOINSERTAPPEND	Replicat が INSERT 文で Oracle APPEND ヒントを使用するかどうかを制御します。
KEYCOLS	行を一意に特定する列を指定します。
MAPEXCEPTION	MAP 文にワイルドカード指定された表を処理する例外が含まれていることを指定します。
REPERROR	MAP 文の実行時に Replicat がエラーに対応する方法を制御します。
RESOLVECONFLICT	競合解決のルールを指定します。
SQLEXEC	ストアド・プロシージャおよび問合せを実行します。
TRIMSPACES | NOTRIMSPACES	CHAR 列を VARCHAR 列にマッピングする際に、末尾の空白を切り捨てるかどうかを制御します。
TRIMVARSPACES | NOTRIMVARSPACES	VARCHAR 列を VARCHAR 列にマッピングする際に、末尾の空白を切り捨てるかどうかを制御します。
WHERE	条件演算子に基づいてレコードを選択します。
;	MAP 文を終了します (必須)。

MAP 句および TARGET 句でのオブジェクト名の指定

MAP 文の MAP 句および TARGET 句にオブジェクト名を指定するには、『Oracle GoldenGate *管理ガイド*』の「Oracle GoldenGate プロセス・インタフェースのスタート・ガイド」を参照してください。

COLMAP の使用

COLMAP では、ソース列を異なる名前のターゲット列に明示的にマップするか、ソース名とターゲット名が同一の場合にデフォルトの列マッピングを指定します。COLMAP は、列データの選択、変換、および移動方法を提供します。

注意　後続の MAP 文のすべての表に適用する列マッピングの *グローバル・ルール*を作成するには、COLMATCH パラメータを使用します。

列名での大 / 小文字の区別および特殊文字のサポート

デフォルトでは、Oracle GoldenGate は二重引用符で囲まれた文字列をリテラルとして処理します。大 / 小文字が区別されるか、特殊文字が含まれる列名をサポートするには、USEANSISQLQUOTES パラメータを使用できます。USEANSISQLQUOTES によって、Oracle GoldenGate は、識別子およびリテラル文字列を区切るための引用符を使用する SQL-92 ルールに従うことができます。USEANSISQLQUOTES を有効にすると、Oracle GoldenGate は、二重引用符で囲まれた文字列を列名として、一重引用符で囲まれた文字列をリテラルとして処理します。このサポートは、Oracle GoldenGate インスタンス内のすべてのプロセスにローバルに適用されます。使用方法および制約事項の詳細は、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』の USEANSISQLQUOTES に関する項を参照してください。

データ定義の生成

構造が同一でないソース表とターゲット表に対して COLMAP を使用する場合は、ソース表のデータ定義を生成し、ターゲットに転送し、SOURCEDEFS パラメータを使用して定義ファイルを特定する必要があります。

ソースおよびターゲットが同一の構造とみなされるためには、同一の列名 (該当する場合は大 / 小文字区別を含む) とデータ型が含まれ、かつ各表の列の順序が同一である必要があります。また、両方の表は、文字用の列の列長セマンテック (バイトまたは文字) が同一である必要があります。

両方の表が同一の構造を持ち、変換などのその他の機能のために COLMAP を使用する場合は、ソース定義ファイルは必要ありません。かわりに ASSUMETARGETDEFS パラメータを使用できます。

詳細は、次を参照してください。

- SOURCEDEFS は 331 ページ
- ASSUMETARGETDEFS は 127 ページ
- 『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』のデータ定義ファイルの作成に関する項

キー列への値のマップ

COLMAP を使用してキー列に値をマップ (主キーの更新操作が発生) する場合、Oracle GoldenGate がターゲット行の特定に使用する WHERE 句で、キー列の正しいビフォア・イメージが使用されません。かわりに、アフター・イメージが使用されます。このため、SQLEXEC 文などのキー列に基づく関数を使用している場合、エラーになります。

次の例で、状況を説明します。この例では、次のオブジェクトがあるものとします。

- ソース表 TCUSTMER1
- ターゲット表 TCUSTMER2
- 2 つの表の列レイアウト

  列 1 = Cust

  列 2 = Name

  列 3 = City

  列 4 = State

- 主キーは、Cust 列、Name 列および City 列です。

MAP 文内の SQLEXEC 文です。

```
SQLEXEC (id mytest, query "select city from TCUSTMER1 WHERE state = 'CA'",
noparams, ERROR RAISE),
```

MAP 文内の COLMAP 文です。

```
COLMAP ( usedefaults, city = mytest.city );
```

イベントの順序です。

1. INSERT 文で次のように挿入します。

   ```
   INSERT into TCUSTMER1 values (Cust = '1234', Name = 'Ace', City = 'SF', State =
   'CA');
   Commit;
   ```

   SQLEXEC 問合せで mytest.city = 'SF' が返されるため、これは成功し、ターゲット表でも City の値が SF に、State の値が CA になります。

2. UPDATE 文でソースの City を SF から LA に変更します。これはターゲットで成功しません。SQLEXEC 問合せは TCUSTMER1 の City 列を参照し、LA という値を返します。COLMAP 句に基づき、City のビフォアとアフターの両バージョンとも LA になります。ターゲット表の City 列に LA という値がないため、ターゲットの WHERE 句を実行すると、SQL エラー 1403 になります。

デフォルトの列マッピングの使用

ソース列と対応するターゲット列の名前が同一の場合は、明示的なマッピング文ではなく、デフォルト・マッピングを使用できます。デフォルト・マッピングでは、Oracle GoldenGate は名前が同じ列を自動的にマップします。適切な場合は、データ変換も自動的に行います。

デフォルト・マッピングを使用するには、USEDEFAULTS オプションを使用します。デフォルト・マッピングは、明示的なマッピング文でマップされていない列にのみ有効です。

デフォルトでは、SQL Server および Sybase の列は、大 / 小文字区別を考慮して比較が行われます。他のすべてのデータベースでは、列名は名前比較のために大文字に変更されます。これらのデータベースで大 / 小文字が区別される列名をサポートするには、GLOBALS ファイルの USEANSISQLQUOTES パラメータを使用します。これにより SQL-92 ルールが適用され、このルールでは、列名を二重引用符で囲み、リテラルを一重引用符で囲む必要があります。

大 / 小文字が認識される場合、USEDEFAULTS は次の方法で大 / 小文字区別をサポートします。

- 名前および大 / 小文字がターゲット列と正確に一致するソース列が見つかった場合、2 つの列はマップされます。
- 大 / 小文字が一致する列が見つからない場合、大 / 小文字の一致にかかわらずターゲット列と名前が一致する最初の候補のソース列を使用してマップが作成されます。

たとえば、次は大 / 小文字を区別する列を含むソース表とターゲット表です。

ソース表 USER1.SM01 の列 :

id

owner

created

changed

creator

modifiedBy

comment

COMMENT

ターゲット表 USER3.SM01 の列 :

ID

owner

id

Creator

comment

ModifiedBy

creationDate

alterationDate

Comment

COMMENT

次に示すこれらの表に対する列マップには、明示的およびデフォルトの列マッピングが含まれています。

```
MAP USER1.SM01, TARGET USER3.SM01,
COLMAP (USEDEFAULTS,
    ID = id,
    creationDate = created,
    alterationDate = changed,
    );
```

このマップの結果を次に示します。デフォルト・マッピングでは、該当する場合には大 / 小文字の区別が識別され、それ以外の場合には名前のみが照合されます。2 つのターゲット列は、明示的にマップされておらず、デフォルト・マッピングも確立できなかったため、マップされません。

マッピングのタイプ	マッピングの結果

明示的マッピング	`ID = id,`
	`creationDate = created,`
	`alterationDate = changed`

デフォルト・マッピング	`owner = owner,`
	`comment = comment,`
	`COMMENT = COMMENT,`
	`Creator = creator,`
	`ModifiedBy = modifiedby`

マッピングされないターゲット列	`id, Comment`

構文

```
MAP <table spec>, TARGET <table spec>,
COLMAP (
[USEDEFAULTS, ]
<target column> = <source expression>
[, BINARYINPUT]
[, ...]
);
```

コンポーネント	説明
`<table spec>`	ソースまたはターゲット表。
`<target column> = <source expression>`	ソースおよびターゲット列間のマップを明示的に定義します。 <target column> はターゲット列名です。オブジェクト名指定のガイドラインは、『Oracle GoldenGate *管理ガイド*』を参照してください。 <source expression> には、次のいずれかを指定できます。 ◆ ソース列名 (例 : ORD_DATE) ◆ 数値定数 (例 : 123) ◆ 引用符で囲まれた文字列定数 (例 : "ABCD") ◆ Oracle GoldenGate 列変換ファンクションを使用する式 (例 : @STREXT (COL1, 1, 3))。列変換ファンクションの詳細は、第 5 章を参照してください。
`BINARYINPUT`	BINARYINPUT は、ターゲット列が RAW または BLOB などのバイナリ・データ型として定義されているが、ソース入力のデータの中にバイナリ・ゼロが含まれている場合に使用します。BINARYINPUT は、単一の列として定義されている完全な Enscribe レコードをターゲット列にレプリケートするときに使用します。ソース入力はバイナリ入力として処理され、データ値の置換は抑止されます。

コンポーネント	説明
USEDEFAULTS	明示的な列マップで指定されていない場合に、同一の名前を持つソース列とターゲット列を自動的にマップします。1 つの列セットには、明示的マップまたは USEDEFAULTS を使用し、両方を使用しないでください。詳細は、237 ページの「デフォルトの列マッピングの使用」を参照してください。USEDEFAULTS は、明示的な列マップの前に指定します。

例 1 `MAP ggs.tran, TARGET ggs.tran2, COLMAP (loc2 = loc, type2 = type);`

例 2 `MAP ggs.tran, TARGET ggs.tran2, COLMAP ( EUROVAL = "\u20ac0" );`

例 3 `MAP ggs.tran, TARGET ggs.tran2, COLMAP (SECTION = @STRCAT("\u00a7", SECTION ));`

COMPARECOLS の使用

COMPARECOLS では、マルチマスター構成で MAP の RESOLVECONFLICT オプションを使用して構成する場合、更新または削除の競合を検出および解決するために Replicat が使用する列を指定します。競合は、トレイル内のレコードのビフォア・イメージと、ターゲット表での現在のデータの間の不一致です。

ビフォア・イメージは、Extract TABLE 文の GETBEFORECOLS パラメータを使用して、トレイル・レコードで使用可能にする必要があります。指定した列は、ターゲット・データベースに存在し、Replicat 構成の一部でもある必要があります (COLMAP 句の有無にかかわらず TARGET 指定を満たします)。

競合解決ルーチンを指定するには、MAP の RESOLVECONFLICT オプションを使用します。COMPARECOLS および RESOLVECONFLICT は、MAP 文に任意の順序で指定できます。競合および競合解決の詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。

構文

```
MAP <source table> , TARGET <target table>,
COMPARECOLS(
{ON UPDATE | ON DELETE}
{ALL |
KEY |
KEYINCLUDING (<col>[,...]) |
ALLEXCLUDING (<col>[,...]) }
[,...]
)
```

引数	説明
{ON UPDATE | ON DELETE}	指定した列のビフォア・イメージを、更新または削除のために比較するかどうかを指定します。ON UPDATE のみ、または ON DELETE のみ、または両方を使用できます。両方使用する場合、同じ COMPARECOLS 句の中に指定します。両方使用する方法は、例を参照してください。

引数	説明
{ALL | KEY | KEYINCLUDING (<col>[,...]) | ALLEXCLUDING (<col>[,...])}	ビフォア・イメージを取得する列を指定します。 ALL: ターゲット表内のすべての列を使用して比較します。対応するビフォア・イメージがトレイルで使用できない場合、エラーが生成されます。ALL を使用すると、Replicat に最も高い処理負荷がかかりますが、すべての列を使用して競合検出の比較を実行し、正確性を最大にすることができます。 KEY: 主キー列のみを比較します。このオプションは最速ですが、キーは一致していても非キー列は異なる可能性があるため、最も正確な競合検出は不可能です。 KEYINCLUDING: 主キー列および指定した列を比較します。これは、速度と検出の正確性との間での妥当な方法です。 ALLEXCLUDING: 指定した列を除くすべての列を比較します。多数の列が含まれる表では、KEYINCLUDING よりも ALLEXCLUDING のほうが効率的な場合があります。キー列は除外*しないでください*。

例 1 次の例では、キー列に加えて、name、address、および salary の各列を競合のために比較します。

```
MAP src, TARGET tgt
COMPARECOLS (
ON UPDATE KEYINCLUDING (name, address,  salary),
ON DELETE KEYINCLUDING (name, address, salary));
```

例 2 次の例では、comment 列は無視して、他のすべての列を競合のために比較します。

```
MAP src, TARGET tgt
COMPARECOLS (ON UPDATE ALLEXCLUDING (comment))
```

DEF の使用

DEF では、ソース定義テンプレートを指定します。定義テンプレートは、特定のソース表に対して DEFGEN が実行されるときに、このオブジェクトの定義に基づいて作成されます。テンプレートが作成されると、この表と同一の定義を持つ新しいソース表は、DEFGEN を実行せずに、かつ Replicat の停止と起動を伴わずに追加できます。DEF で指定されたテンプレートの定義は、定義の参照に使用されます。DEFGEN の詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。

構文

```
MAP <table spec>, TARGET <table spec>, DEF <definitions template>;
```

引数	説明
<definitions template>	DEFGEN パラメータ・ファイルの TABLE の DEF オプションで指定されている定義テンプレート名。テンプレートに含まれる定義は、この MAP 文の表の定義と同一である必要があります。

例

```
MAP acct.cust*, TARGET acct.cust*, DEF custdef;
```

EVENTACTIONS の使用

EVENTACTIONS では、特定のフィルタリング・ルールに適合する、*イベント・レコード*と呼ばれるトレイル内のレコードに基づいて、Replicat に定義済のアクションを実行させます。このシステムを使用すると、データベース・イベントに基づいて処理をカスタマイズできます。たとえば、変換の実行または統計のレポートのために、プロセスを一時停止できます。

ターゲット表へのデータ適用を必要としないアクションのトリガーを目的にイベント・マーカー・システムを使用するために、Replicat 用 TABLE パラメータと EVENTACTIONS をサポートしているフィルタリング・オプションを使用できます。377 ページを参照してください。

警告 **EVENTACTIONS は、ソース・データベースが Teradata で、Extract が最大パフォーマンス・モードで構成されている場合にはサポートされません。**

すべてではありませんが、多くの EVENTACTIONS オプションは (Extract 用) TABLE および (Replicat 用) MAP 両方に適用されるため、ここでは両方のプロセスのすべてのオプションを説明します。一方のみに適用されるオプションには、その旨記載しています。

複数のアクションを組み合せる方法

- すべてではありませんが、多くの EVENTACTIONS オプションは組み合せることができます。目的の達成のために、複数のアクションを組み合せる必要がある場合があります。
- 最初に EVENTACTIONS 文全体が解析されてから、優先度に従って、指定したオプションが実行されます。次のリストでは、**レコードの処理**の前にリストされているアクションは、レコードがトレイルに書き込まれる前またはターゲットに適用される前に発生します (プロセスによって異なります)。**レコードの処理**の後ろにリストされているアクションは、レコードが処理された後に実行されます。
  - TRACE
  - LOG
  - CHECKPOINT BEFORE
  - IGNORE
  - DISCARD
  - SHELL
  - ROLLOVER
  - (レコードの処理)
  - REPORT
  - SUSPEND
  - ABORT
  - CHECKPOINT AFTER
  - FORCESTOP
  - STOP

標準の方法でイベント・レコード自体の処理を防ぐには、IGNORE または DISCARD オプションを使用します。IGNORE および DISCARD は、レコードより先に評価されるため、これらを使用してイベント・レコードの処理を防ぐことができます。これらのオプションが指定されていない場合、Extract はトレイルにレコードを書き込み、Replicat はレコードに含まれている操作をターゲット・データベースに適用します。

1 つのトランザクションに、イベント・アクションをトリガーするレコードが複数含まれている可能性があることに注意してください。そのようなケースでは、指定されている特定の EVENTACTIONS が複数回実行されることがあります。たとえば、2 回の連続する ROLLOVER アクションをトリガーする 2 つのレコードを検出すると、Extract はトレイルを 2 回ロールオーバーするため、実質上 2 つのトレイル・ファイルの 1 つは空になります。

構文

```
EVENTACTIONS (
[STOP | SUSPEND | ABORT | FORCESTOP]
[IGNORE [RECORD | TRANSACTION [INCLUDEVENT]]
[DISCARD]
[LOG [INFO | WARNING]]
[REPORT]
[ROLLOVER]
[SHELL "<command>" |
    SHELL ("<command>", VAR <variable> = {<column name> | <expression>}
    [, ...][, ...]) ]
[TRACE[2] <trace file> [TRANSACTION] [DDL[INCLUDE] | DDLONLY]
    [PURGE | APPEND]]
[CHECKPOINT [BEFORE | AFTER | BOTH]]
[, ...]
)
```

アクション	説明
STOP	指定のイベント・レコードが検出されたときにプロセスを正常に停止させます。プロセスは、オープンしているトランザクションが完了するまで待機してから停止します。Replicat によってグループ化されたトランザクション、またはバッチ・トランザクションの場合、プロセスは、現在のトランザクション・グループが適用されてから正常に停止します。イベント・レコードがトランザクションの終了点であることも示している場合、プロセスはイベント・レコードの次のレコードで再開します。 プロセスは、トランザクションがまだオープンしているために即座に停止できない場合、メッセージを記録します。ただし、長時間オープンしているトランザクション内でイベント・レコードを検出しても、トランザクションがコミットされていないことを通知する警告メッセージは出しません。したがって、STOP イベントが発生しても、プロセスが長時間実行中になる場合があります。 STOP は、ABORT および FORCESTOP を除く、他の EVENTACTIONS オプションと組み合せることができます。

アクション	説明
SUSPEND	現在の実行のアクティブ・コンテキストを保持し、GGSCI で発行される SEND コマンドに応答できるように、プロセスを一時停止します。プロセスが一時停止すると、INFO コマンドによって RUNNING と示され、RBA フィールドに最後のチェックポイントの位置が表示されます。 処理を再開するには、SEND <group> コマンドを RESUME オプションとともに発行します。 CHECKPOINT BEFORE オプションを SUSPEND とあわせて使用するには、イベント・レコードを、SUSPEND が発生するトランザクションの開始にする必要があります。これにより、一時停止状態の間にプロセスが中断した場合、SUSPEND アクションのイベント・レコードは、再起動時に再処理する最初のレコードになります。CHECKPOINT BERORE および SUSPEND の両方を指定しましたが、イベント・レコードがトランザクションの開始ではない場合、SUSPEND が発生する前にプロセスは異常終了します。 CHECKPOINT AFTER オプションを SUSPEND とあわせて使用するには、チェックポイントが発生する前に RESUME コマンドを発行する必要があり、イベント・レコードを COMMIT レコードにする必要があります。SUSPEND 状態の間にプロセスが中断した場合、プロセスは再起動時に最後にチェックポイントが指定された位置からトランザクションを再処理します。 SUSPEND は、ABORT と組み合せることはできませんが、他のすべてのオプションと組み合せることができます。
ABORT	オープンしているトランザクションの有無に関わらず、指定のイベント・レコードが検出されたときに即座にプロセスを終了させます。イベント・レコードは処理されません。ログに致命的なエラーが書き込まれ、DISCARD も指定している場合には破棄ファイルにイベント・レコードが書き込まれます。プロセスの起動時にリカバリが行われます。 ABORT は、CHECKPOINT BEFORE、DISCARD、SHELL および REPORT とのみ組み合せることができます。
FORCESTOP	指定のイベントが検出され、イベント・レコードがトランザクションの最後の操作またはトランザクションの唯一のレコードの場合にのみ、プロセスを正常に停止させます。レコードは正常に書き込まれます。 長時間オープンしているトランザクション内でイベント・レコードが検出された場合、プロセスはログに警告メッセージを記録し、ABORT と同様に即座に終了します。このケースでは、起動時にリカバリが必要です。長時間におよぶトランザクションの途中で FORCESTOP アクションがトリガーされると、プロセスは警告メッセージを出さずに終了します。 FORCESTOP は、ABORT、STOP、CHECKPOINT AFTER および CHECKPOINT BOTH を除く、他の EVENTACTIONS オプションと組み合せることができます。ROLLOVER とともに使用する場合、ロールオーバーはプロセスが正常に停止したときにのみ行われます。

アクション	説明
IGNORE [RECORD | TRANSACTION [INCLUDEVENT]]	選択したアクションに応じて、トランザクションの一部またはすべてを無視します。 ◆ RECORD はデフォルトです。残りのトランザクションではなく、プロセスで指定されたイベント・レコードのみが無視されます。警告またはメッセージはログに書き込まれませんが、Oracle GoldenGate 統計が更新され、レコードが無視されたことが統計に反映されます。 ◆ TRANSACTION では、イベントをトリガーしたレコードを含むトランザクション全体を無視します。TRANSACTION を使用する場合は、イベント・レコードをトランザクション内の最初のレコードにする必要があります。トランザクションを無視する場合、デフォルトではイベント・レコードも無視されます。TRANSACTION は、TRANS に短縮できます。 ◆ INCLUDEEVENT を TRANSACTION とともに指定すると、イベント・レコードはトレイルまたはターゲットに伝播されますが、関連トランザクションの残りの部分は無視されます。 IGNORE は、ABORT および DISCARD を除く、他のすべての EVENTACTIONS オプションと組み合せることができます。 このアクションは、DDL レコードには無効です。DDL 操作は自律的であるため、レコードを無視することはトランザクション全体を無視することと同等です。
DISCARD	プロセスに次のことを実行させます。 ◆ 指定のイベント・レコードを破棄ファイルに書き込む。 ◆ Oracle GoldenGate 統計を更新し、レコードが破棄されたことを統計に反映させる。 プロセスは、トレイルの次のレコードから処理を再開します。このオプションを使用する場合は、DISCARDFILE パラメータを使用して破棄ファイルの名前を指定します。デフォルトでは、破棄ファイルは作成されません。 DISCARD は、IGNORE を除く、他のすべての EVENTACTIONS オプションと組み合せることができます。
LOG [INFO | WARNING]	指定のイベント・レコードが検出されたときに、プロセスにこのイベントを記録させます。レポート・ファイル、Oracle GoldenGate エラー・ログ、およびシステム・イベント・ログにメッセージが書き込まれます。 次のオプションを使用して、メッセージの重大度を指定します。 ◆ INFO では、重大度の低い情報メッセージを指定します。これはデフォルトです。 ◆ WARNING では、重大度の高い警告メッセージを指定します。 LOG は、ABORT を除く、他のすべての EVENTACTIONS オプションと組み合せることができます。ABORT を使用する場合は、プロセスが終了する前に ABORT によって致命的なエラーが記録されるため、LOG は不要です。

アクション	説明
REPORT	指定のイベント・レコードが検出されたときにプロセスにレポート・ファイルを生成させます。GGSCI で SEND コマンドと REPORT オプションを使用する場合と同じ結果になります。 (DISCARD、IGNORE または ABORT を使用している場合を除き) REPORT メッセージは、イベント・レコードが処理された後に発生するので、レポート・データにはイベント・レコードが含まれます。 REPORT は、他のすべての EVENTACTIONS オプションと組み合せることができます。
ROLLOVER	Extract にのみ有効です。指定のイベント・レコードが検出されたときに、Extract にトレイルの新しいファイルにロールオーバーさせます。ROLLOVER アクションは、Extract がトレイル・ファイルにイベント・レコードを書き込む前に発生するので、DISCARD、IGNORE または ABORT も使用されている場合を除き、イベント・レコードが新しいファイルの最初のレコードになります。 ROLLOVER は、ABORT を除く、他のすべての EVENTACTIONS オプションと組み合せることができます。 注意 : ROLLOVER は、次の理由から ABORT とは組み合せることができません。 ◆ ROLLOVER を使用すると、プロセスがチェックポイントの書込みを行わない。 ◆ ROLLOVER は ABORT よりも先に発生する。 ROLLOVER のチェックポイントがない場合に ABORT を使用すると、Extract は再起動時に前回のトレイル・ファイル内の前回のチェックポイントから再開します。そのため、実質上ロールオーバーがキャンセルされます。
SHELL "<command>"	イベント・レコードが検出されたときに、指定したシェル・コマンドをプロセスに実行させます。SHELL "<command>" は、基本的なシェル・コマンドを実行します。コマンド文字列は、リテラル値で取得され、その方法でシステムに送信されます。コマンドは大 / 小文字が区別され、二重引用符で囲む必要があります。たとえば、次のようになります。 `EVENTACTIONS (SHELL "echo hello world! > output.txt")` シェル・コマンドが成功した場合、プロセスはレポート・ファイルおよびイベント・ログに情報メッセージを書き込みます。コマンドの成功は、UNIX シェル言語に従って、コマンドの終了ステータスで判断します。UNIX シェル言語では、0 が成功を表します。 システム・コールが成功しなかった場合、プロセスは致命的なエラーとともに異常終了します。UNIX シェル言語では、0 以外は失敗を表します。エラー・メッセージは、SHELL コマンド自体の実行にのみ関連し、従属するコマンドの終了ステータスには関連しないことに注意してください。たとえば、SHELL はスクリプトを正常に実行できますが、そのスクリプト内のコマンドは失敗することがあります。 SHELL は、他のすべての EVENTACTIONS オプションと組み合せることができます。

アクション	説明
SHELL ("<command>", VAR <variable> = {<column name> | <expression>} [, ...][, ...])	イベント・レコードが検出されたときに、指定したシェル・コマンドをプロセスに実行させ、パラメータの受渡しをサポートします。コマンドおよびパラメータは、大 / 小文字が区別されます。 デフォルトでは、二重引用符の内側の入力値は、リテラル・テキストとして処理されます。ただし、USEANSISQLQUOTES パラメータを GLOBALS ファイル内で使用して、SQL-92 ルールを適用できます。ここでは、一重引用符で囲まれたテキストはリテラルとして処理され、二重引用符で囲まれたテキストは名前として処理されます。「USEANSISQLQUOTES」も参照してください。 SHELL を引数とともに使用する場合、コマンドおよび引数の文字列全体をカッコで囲む必要があります。たとえば、次のようになります。 `EVENTACTIONS (SHELL ("Current timestamp: $1  SQLEXEC result is $2 ",VAR $1 = @GETENV("JULIANTIMESTAMP"),VAR $2 = mytest.description));` 入力内容は次のようになります。 <command> はコマンドで、システムにリテラルに渡されます。 VAR は、パラメータ入力を開始する必須のキーワードです。 <variable> は、ランタイム変数の値が置き換えられる、プレースホルダ変数のユーザー定義名です。コマンドで使用されない余分な変数は無視されます。VAR 変数名に一致する SHELL コマンド内のリテラルは、置換済の VAR の値によって置き換えられることに注意してください。これは、予期しない結果になる可能性があるため、本番環境に移行する前にコードのテストを実行してください。 <column name > は、列値の前または後 (現在) のイメージになります。 <expression> は、列データや DDL の処理の有無に応じて、次のようになります。 **列データの有効な式は、次のとおりです。** ◆ TABLE 文の TOKENS 句からの値。 ◆ 任意の Oracle GoldenGate 列変換ファンクションからの戻り値。 ◆ SQLEXEC 問合せまたはプロシージャからの戻り値。 **DDL の有効な式は、次のとおりです。** ◆ @TOKEN ファンクションからの戻り値 (Replicat のみ)。 ◆ @GETENV ファンクションからの戻り値。 ◆ 列データを参照しないその他のファンクション (@DATENOW など) からの戻り値。 ◆ @DDL ファンクションからの戻り値。

アクション	説明
TRACE[2] <trace file> [TRANSACTION] [DDL[INCLUDE] | DDLONLY] [PURGE | APPEND]	指定のイベント・レコードが検出されたときに、トレース情報をトレース・ファイルに書き込ませます。TRACE は、ステップバイステップの処理情報を提供します。TRACE2 では、プロセスがほとんどの時間を費やしているコード・セグメントを特定します。 デフォルト (オプションなし) では、トランザクション境界を考慮しない標準的な DML トレースは、プロセスが終了するまで有効になります。 ◆ <trace file> にはトレース・ファイル名を指定し、TRACE キーワードの直後に配置します。独自のトレース・ファイルを指定することも、単独の TRACE または TRACE2 パラメータを使用して指定しているデフォルトのトレース・ファイルを使用することもできます。 EVENTACTIONS TRACE が使用されている様々な TABLE または MAP 文で同一のトレース・ファイルを使用できます。複数の TABLE または MAP 文で同一のトレース・ファイル名が指定されていて、TRACE オプションの指定方法が一貫していない場合は、このトレース・ファイルを含む最後に解決された TABLE または MAP のオプションが優先されます。 ◆ TRANSACTION では、プロセスの終了時までではなく、現在のトランザクションの終了時までのトレースのみ有効にします。Replicat のトランザクションの境界は、Replicat によってグループ化されたトランザクションまたはバッチのターゲット・トランザクションではなく、ソース・トランザクションに基づきます。TRANSACTION は、TRANS に短縮できます。このオプションは、DML 操作にのみ有効です。 ◆ DDL[INCLUDE] では、DDL のトレースに加え、DML トランザクション・データ処理もトレースします。DDL または DDLINCLUDE のいずれかが有効です。 ◆ DDLONLY では、DDL をトレースし、DML トランザクション・データはトレースしません。 これらのオプションは Replicat にのみ有効です。デフォルトでは、DDL トレースは無効化されます。 ◆ PURGE では、追加のトレース・レコードを書き込む前にトレース・ファイルを切り捨て、APPEND では、既存のレコードの末尾に新しいトレース・レコードを書き込みます。デフォルトは APPEND です。 TRACE は、ABORT を除く、他のすべての EVENTACTIONS オプションと組み合せることができます。 指定したトレース・ファイルへのトレースを無効にするには、GGSCI の SEND <process> コマンドと TRACE OFF <filename> オプションを発行します。

アクション	説明
CHECKPOINT [BEFORE | AFTER | BOTH]	指定のイベント・レコードが検出されたときに、プロセスにチェックポイントを書き込ませます。チェックポイント・アクションは、TABLE または MAP 文に定義されている処理の前後のコンテキストを提供します。このコンテキストは開始点と終了点を持つため、SQLEXEC およびユーザー・イグジットで実行されるファンクションをマッピングするための同期点を提供します。 ◆ BEFORE Extract プロセス用の BEFORE では、Extract がトレイルにイベント・レコードを書き込む前にチェックポイントを書き込みます。 Replicat プロセス用の BEFORE では、Replicat がレコードに含まれている SQL 操作をターゲットに適用する前にチェックポイントを書き込みます。 BEFORE を使用する場合は、イベント・レコードをトランザクションの最初のレコードにする必要があります。イベント・レコードが最初のレコードでない場合、プロセスは異常終了します。BEFORE を使用することで、イベント・レコードから始まるトランザクション以前のすべてのトランザクションを確実にコミットできます。 DDL レコードに EVENTACTIONS を使用する場合、各 DDL レコードは自律的であるため、DDL レコードはトランザクションの開始であることが保証されることに注意してください。そのため、CHECKPOINT BEFORE イベント・アクションは DDL レコードに対して暗黙的です。 CHECKPOINT BEFORE は、すべての EVENTACTIONS オプションと組み合せることができます。 ◆ AFTER Extract 用の AFTER では、Extract がトレイルにイベント・レコードを書き込んだ後にチェックポイントを書き込みます。 Replicat 用の AFTER では、レコードに含まれている SQL 操作を Replicat がターゲットに適用した後にチェックポイントを書き込みます。 AFTER は、チェックポイント・リクエストに注意としてフラグを立て、プロセスが次の可能な機会にのみチェックポイントを発行することを示します。たとえば、イベント・レコードがマルチレコード・トランザクションの 1 つである場合、チェックポイントは、Oracle GoldenGate データ整合性モデルに従って、次のトランザクションの境界で発生します。 DDL レコードに EVENTACTIONS を使用する場合、各 DDL レコードは自律的であるため、DDL レコードはトランザクションの終了 (境界) であることが保証されることに注意してください。そのため、CHECKPOINT AFTER イベント・アクションは DDL レコードに対して暗黙的です。 CHECKPOINT AFTER は、ABORT を除く、すべての EVENTACTIONS オプションと組み合せることができます。

アクション	説明
	◆ BOTH BOTH は、BEFORE および AFTER の組合せです。Extract または Replicat プロセスは、イベント・レコードの処理前および処理後にチェックポイントを書き込みます。 CHECKPOINT BOTH は、ABORT を除く、すべての EVENTACTIONS オプションと組み合せることができます。 CHECKPOINT は、CP に短縮できます。

例 1 次に、特定のレコードを無視するようにプロセスを構成する方法の例を示します。Replicat は、name = goldengate を持つトレイル・レコードを処理するときに、このレコードを無視します。

```
MAP <owner.table>, TARGET <owner2.table2>, &
WHERE (name = "goldengate"), &
EVENTACTIONS (ignore);
```

例 2 EVENTACTIONS オプションの互換性および優先度ルールでは、DISCARD は ABORT よりも優先されるため、この例では、イベント・レコードはプロセスが異常終了する前に破棄ファイルに書き込まれます。

```
MAP <owner.table>, TARGET <owner2.table2>, &
WHERE (name = "goldengate"), &
EVENTACTIONS (DISCARD, ABORT);
```

例 3 次の例では、SHELL アクションを実行します。SQLEXEC 問合せの結果を取得し、現在のタイムスタンプと組み合せます。

```
MAP src.tab1, TARGET targ.tab1 &
SQLEXEC (id mytest, query "select description from lookup &
where pop = :mycol2", params (mycol2 = col2) ), &
EVENTACTIONS (SHELL ("Current timestamp: $1  SQLEXEC result is $2 ", &
VAR $1 = @GETENV("JULIANTIMESTAMP"), VAR $2 = mytest.description));
```

この例の結果として生じるシェル・コマンドは、次のようになります。

```
"Current timestamp: 212156002704718000  SQLEXEC result is test passed"
```

例 4 次に、プレースホルダ名がコマンド文字列のリテラル・テキストと競合するときに無効な結果が発生する場合の例を示します。この例では、"$1" という名前のプレースホルダは列値と関連付けられ、SHELL コマンドは、"$1" を含むリテラル文字列をエコーします。

```
MAP src.tab1, TARGET targ.tab1 &
EVENTACTIONS (SHELL ("echo Extra charge for $1 is $1", VAR $1 = COL1));
```

列値が "gift wrap" とすると、次は予期しない結果です。

```
"Extra charge for gift wrap is gift wrap"
```

プレースホルダ変数を "$col" に変更すると、結果は次のような正しい出力になります。

```
MAP src.tab1, TARGET targ.tab1 &
EVENTACTIONS (SHELL ("echo Extra charge for $col is $1", VAR $col = COL1));

"Extra charge for gift wrap is $1"
```

次に、予期しない結果に類似した可能性を示します。

```
MAP src.tab1, TARGET targ.tab1 &
EVENTACTIONS (SHELL ("Timestamp: $1  Price is $13 > out.txt ", &
VAR $1 = @GETENV("JULIANTIMESTAMP")));
```

リダイレクトされた出力ファイルには、次のような文字列が含まれることがあります (2 つ目のタイムスタンプには、3 が追加されていることを通知します)。

```
"Timestamp: 212156002704718000 Price is 2121560027047180003"
```

目的の結果は、次のとおりです。

```
"Timestamp: 212156002704718000 Price is $13"
```

例 5 次に、DDL 操作に使用される SHELL の例を示します。@DDL ファンクションは、DDL 文のテキストを返すために使用されます。

```
DDL INCLUDE OBJNAME src.t* &
EVENTACTIONS (SHELL ("echo The DDL text is var1> out.txt ", &
VAR var1 = DDL(TEXT));
```

リダイレクトされた出力ファイルには、次のような文字列が含まれることがあります。

```
"The DDL text is CREATE TABLE src.test_tab (col1 int);"
```

例 6 次に、トレースの異なる構成方法の例を示します。

```
MAP tab1, TARGET tab1 EVENTACTIONS (TRACE ./dirrpt/trace1.txt);
MAP tab2, TARGET tab2 EVENTACTIONS (TRACE ./dirrpt/trace2.txt TRANSACTION);
```

- 最初の MAP 文では、最初の tab1 イベント・レコードがターゲットに適用される直前に trace1.txt トレース・ファイルが生成されます。これには、この時点から Replicat が終了するまで、または GGSCI の SEND REPLICAT コマンドでトレースが無効化されないかぎり、すべてのトレース情報が含まれます。
- 2 つ目の MAP 文には、TRANSACTION オプションが含まれているため、trace2.txt ファイルは最初の tab2 イベント・レコードがターゲットに適用される直前に作成されますが、トレースは tab2 イベント・レコードを含むトランザクションの終了時に自動的に停止します。

イベント・マーカー・システムの他の使用例および詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。

例 7 次に、EVENTACTIONS を SUSPEND とともに使用する方法を示します。

- DDL をレプリケートしていて、新しい表を作成するターゲット・データベースに十分な領域があることを確認するとします。CREATE TABLE DDL 操作をマップする MAP 文で、EVENTACTIONS を SUSPEND とともに使用します。その MAP 文内で SQL 文を実行して、表領域に残っている領域の量を問い合せます。十分な領域がある場合、SEND EXTRACT を RESUME とともに使用して、即座に処理を再開します。十分な領域がない場合、DBA が領域を追加するまで Replicat は一時停止のままになり、処理を再開するには SEND EXTRACT を RESUME とともに使用します。

- 任意の表で一意キー違反が発生した場合、それを修正するとします。Replicat は何千もの表を処理しており、Replicat が再起動時にオブジェクト・キャッシュを再び再構築するのに時間がかかるため、違反があるたびに処理を停止したくないとします。EVENTACTIONS を SUSPEND とともに使用することにより、問題を修正するまで処理を一時停止することができます。
- 1 日の終わりに、日時レポートを実行するために Replicat を一時停止した後、プロセスを停止および再起動せずに処理を即座に再開します。

EXCEPTIONSONLY の使用

EXCEPTIONSONLY は、エラーを処理するための例外 MAP 文で使用します。例外 MAP 文は、エラーが発生する可能性がある MAP 文に続ける必要があります。例外 MAP 文は、先行する標準の MAP 文で処理された最後のレコードでエラーが発生した場合にのみ実行されます。

EXCEPTIONSONLY を使用するには、標準の MAP 文内、またはパラメータ・ファイルのルートで、REPERROR 文と EXCEPTION オプションを使用します。REPERROR の詳細は、299 ページを参照してください。

注意 **Oracle GoldenGate の競合の検出および解決 (CDR) 機能を使用する場合、REPERROR と EXCEPTION は必要ありません。CDR は、エラーを引き起こすすべての操作を、例外 MAP 文に自動的に送信します。**

例外 MAP 文では、標準の MAP 文と同じ SOURCE 表を指定する必要がありますが、例外 MAP 文の TARGET 表は例外表である必要があります。例外 MAP 文を使用する場合は、標準の MAP 文でワイルドカード・オブジェクト名を使用しないでください。

注意 **オブジェクト名のワイルドカードをサポートするには、MAPEXCEPTION オプションを参照してください。**

例外 MAP 文の使用の詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。

構文

```
MAP <table spec>, TARGET <table spec>, EXCEPTIONSONLY
```

EXITPARAM の使用

EXITPARAM では、MAP 文からレコードを検出するたびにユーザー・イグジット・ルーチンにパラメータを渡します。ユーザー・イグジットの詳細は、第 6 章を参照してください。

構文

```
MAP <table spec>, TARGET <table spec>,
EXITPARAM "<parameter string>";
```

コンポーネント	説明
`"<parameter string>"`	リテラル文字列のパラメータ。パラメータは二重引用符で囲みます。パラメータ文字列では、最大で 100 文字まで指定できます。

FILTER の使用

FILTER では、数値に基づいてレコードを選択または除外します。フィルタ式では、条件演算子、Oracle GoldenGate 列変換ファンクション、またはその両方を使用できます。

注意 **文字列に基づいてフィルタするには、文字列ファンクションを使用するか、WHERE オプションを使用します。**

FILTER コンポーネントはすべてコンマで区切ります。FILTER 句には、次を含めることができます。

- 数字
- 数字を含む列
- 数字を返すファンクション
- 算術演算子 :

  +(加算)
  -(減算)
  *(乗算)
  /(除算)
  \(余り)

- 比較演算子 :

  >(より大きい)
  >=(以上)
  <(より少ない)
  <=(以下)
  =(等しい)
  <>(等しくない)

  比較から導出した結果はゼロ (FALSE を示す) またはゼロ以外 (TRUE を示す) になります。

- カッコ (式の結果をグループ化)
- 結合演算子 : AND、OR

構文

```
MAP <table spec>, TARGET <table spec> , FILTER (
[, ON INSERT | ON UPDATE| ON DELETE]
[, IGNORE INSERT | IGNORE UPDATE | IGNORE DELETE]
, <filter clause>
[, RAISEERROR <error>]
);
```

コンポーネント	説明
<filter clause>	次のように、式に基づいてレコードを選択します。 FILTER ((PRODUCT_PRICE*PRODUCT_AMOUNT)>10000)) フィルタ句では、次の例のように、Oracle GoldenGate の列変換ファンクションを使用できます。 FILTER (@COMPUTE (PRODUCT_PRICE*PRODUCT_AMOUNT)>10000) デフォルトでは、Oracle GoldenGate は、FILTER (@STRFIND(NAME, "JOE") > 0) のように、二重引用符で囲まれた入力文字列をリテラルとして処理します。識別子およびリテラルに SQL-92 ルールを使用するには、GLOBALS ファイルで USEANSISQLQUOTES パラメータを使用します。 Oracle GoldenGate では、マルチバイト文字セットまたはローカル・オペレーティング・システムの文字セットと互換性のない文字セットを含む列に対して、FILTER をサポートしていません。 フィルタ句のファイルの最大サイズは 5,000 バイトです。
ON INSERT | ON UPDATE| ON DELETE	レコードのフィルタリングを、指定した操作に限定します。次のように、操作はコンマで区切ります。 FILTER (ON UPDATE, ON DELETE, @COMPUTE (PRODUCT_PRICE*PRODUCT_AMOUNT)>10000) この例では、更新および削除に対してフィルタを実行しますが、挿入に対しては実行しません。
IGNORE INSERT | IGNORE UPDATE | IGNORE DELETE	指定した操作にフィルタを適用しません。次のように、操作はコンマで区切ります。 FILTER (IGNORE INSERT, @COMPUTE (PRODUCT_PRICE*PRODUCT_AMOUNT)>10000) この例では、更新および削除に対してフィルタを実行しますが、挿入は無視します。
RAISEERROR <error>	フィルタが失敗したときにユーザー定義のエラー番号を発行します。REPERROR パラメータの入力として使用し、エラー処理を起動できます。<error> は、データベースまたは Oracle GoldenGate によって使用されているエラー番号範囲外の値にしてください。(例 : RAISEERROR 21000)。

HANDLECOLLISIONS | NOHANDLECOLLISIONS の使用

HANDLECOLLISIONS および NOHANDLECOLLISIONS では、ソース表のターゲット表への初期ロードが行われている間に、Oracle GoldenGate によってレプリケートされた同じソース表へのトランザクション変更を Oracle GoldenGate が調整するかどうかを制御します。HANDLECOLLISIONS は、初期ロード方法を使用し、ソース表がオンライン状態のままでユーザーによってデータが変更されている場合に必要です。HANDLECOLLISIONS を使用すると、初期ロードが終了して Oracle GoldenGate がレプリケートされた変更を適用するときに、Replicat はターゲット表の重複レコードを上書きし、行方不明レコードのエラーには別のエラー処理方法で対処します。

HANDLECOLLISIONS および NOHANDLECOLLISIONS は、パラメータ・ファイルでグローバルに使用することも、表グループに対する個別の有効化 / 無効化スイッチとして使用することもできます。MAP 文で使用される場合は、他の場所の指定よりも優先されます。HANDLECOLLISIONS の詳細は、218 ページを参照してください。

構文

```
MAP <table spec>, TARGET <table spec>,
[HANDLECOLLISIONS | NOHANDLECOLLISIONS];
```

例 1

次に、MAP 文内での基本的な使用例を示します。

```
MAP dbo.tcust, TARGET dbo.tcust, HANDLECOLLISIONS;
```

例 2

次に、グローバルな使用とパラメータ・ファイル内での MAP 固有の使用を組み合せた例を示します。MAP の指定はグローバルな指定よりも優先されるため、tcust 表のペアに対しては衝突は処理されません。

```
REPLICAT fin
USERID ggs, PASSWORD AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC &
    AES128, ENCRYPTKEY securekey1
HANDLECOLLISIONS
ASSUMETARGETDEFS
MAP dbo.torders, TARGET dbo.torders;
MAP dbo.tprod, TARGET dbo.tprod;
MAP dbo.tcust, TARGET dbo.tcust, NOHANDLECOLLISIONS;
```

INSERTALLRECORDS の使用

INSERTALLRECORDS パラメータでは、レコードの現在のバージョンのみではなく、ターゲット・レコードに行われたすべての操作のレコードを保持します。INSERTALLRECORDS を使用すると、Replicat はレコードに対するすべての変更操作を新しいレコードとしてデータベースに挿入します。最初の挿入、その後の更新および削除は、ポイントインタイム・スナップショットとして保持されます。

履歴データと特別なトランザクション情報を組み合せることで、より有益なターゲット・レポート・データベースを作成できます。トランザクション履歴表の保持の詳細は、『*Oracle GoldenGate Windows and UNIX 管理者ガイド*』を参照してください。

INSERTALLRECORDS は、複数の MAP 文に同時に適用するために、パラメータ・ファイルのルート・レベルで単独パラメータとしても使用できます。224 ページを参照してください。

構文

```
MAP <table spec>, TARGET <table spec>, INSERTALLRECORDS;
```

INSERTAPPEND | NOINSERTAPPEND の使用

INSERTAPPEND および NOINSERTAPPEND パラメータでは、Replicat が Oracle ターゲット表に INSERT 操作を適用するときに APPEND ヒントを使用するかどうかを制御します。これらのパラメータは、Oracle データベースにのみ有効です。

これらのパラメータは 2 つの方法で使用でき、パラメータ・ファイルのルートで単独パラメータとして使用する場合は、一方のパラメータは、もう一方のパラメータが見つかるまで、それ以降のすべての TABLE または MAP 文に有効です。1 つの MAP 文内で使用する場合は、この MAP 文よりも先に指定されているすべての単独の INSERTAPPEND または NOINSERTAPPEND エントリよりも優先されます。

INSERTAPPEND を指定すると、Oracle ターゲット表に INSERT 操作を適用するときに、Replicat は APPEND_VALUES ヒントを使用します。レプリケートされるトランザクションが大きく、同一の表への複数の挿入が含まれている場合、ヒントの使用は適切なパフォーマンスの向上策です。トランザクションが小さい場合に INSERTAPPEND を使用すると、パフォーマンスが低下することがあります。APPEND ヒント使用の詳細は、Oracle のマニュアルを参照してください。

INSERTAPPEND を使用する場合は、BATCHSQL パラメータを使用する必要があります。BATCHSQL を使用しない場合、Replicat は異常終了します。

Replicat 用 INSERT 文のデフォルトは、NOINSERTAPPEND です。

単独で使用する場合の構文および例は、223 ページの「INSERTAPPEND | NOINSERTAPPEND」も参照してください。

構文

```
MAP <table spec>, TARGET <table spec>, [INSERTAPPEND | NOINSERTAPPEND];
```

例

次では、inventory 表を除き、MAP 文のすべての表で INSERTAPPEND が使用されます。

```
INSERTAPPEND
MAP fin.orders, TARGET fin.orders;
MAP fin.inventory, TARGET fin.inventory, NOINSERTAPPEND;
MAP fin.customers, TARGET fin.customers;
```

KEYCOLS の使用

KEYCOLS では、ターゲット表の 1 つ以上の列を一意列として定義します。主に KEYCOLS は、ターゲット表で主キーまたは一意索引が使用できないときに、代替主キーを定義するために使用します。

ソースおよびターゲットのキー列または一意索引列は、データベースで定義されている場合も、KEYCOLS によって代替キーが指定されている場合でも、一致する必要があります。ソース表には、少なくともターゲット表と同じ数のキー列または索引列が含まれている必要があります。そうでなければ、ソースのキー列または索引列を更新する際に、Replicat は余剰なターゲット列のビフォア・イメージを取得できません。

キーを定義する際は、次のガイドラインに従ってください。

- ソース表とターゲット表両方にキーまたは一意索引が含まれていない場合は、TABLE および MAP 文両方で KEYCOLS を使用し、一致する列セットを指定します。
- いずれか一方の表にキーまたは一意索引が含まれていない場合は、その表に対して KEYCOLS を使用し、もう一方の表の実際のキーまたは索引列に一致する列を指定します。一致する列セットを定義できない場合は、TABLE および MAP 文両方で KEYCOLS を使用し、一意の値が含まれる一致する列セットを指定します。KEYCOLS はキーまたは一意索引よりも優先されます。
- ターゲット表にソース表よりも大きなキー (または多くの一意索引列) が含まれている場合は、TABLE 文で KEYCOLS を使用し、実際のソースのキーまたは索引列に加え、余剰なターゲット列と一致するソース列を指定する必要があります。表に主キーまたは一意索引が含まれている場合、KEYCOLS の指定はこれらよりも優先されるため、余剰な列のみを指定しないでください。このように KEYCOLS を使用すると、キーまたは索引列は更新のときにビフォア・イメージを利用できます。

KEYCOLS を使用するときは、指定した列をトランザクション・ログに記録し、Replicat がトレイルで使用できるようにしてください。この設定は、データベース・インタフェースを使用するか、ADD TRANDATA コマンドの COLS オプションを使用して行えます (Oracle のみ)。

ターゲット表では、KEYCOLS で定義したキー列に一意索引を作成します。索引によって、Oracle GoldenGate が処理する必要があるターゲット行をより高速に特定できます。

構文

```
MAP <table spec>, TARGET <table spec>, KEYCOLS (<column> [, ... ]);
```

コンポーネント	説明
(<column>)	代替主キーとして使用する列を定義します。複数の列を指定するには、次のようにコンマ区切りリストを作成します。 KEYCOLS (id, name 主キーまたは一意キーが存在する場合は、これらの列を KEYCOLS 指定に含める必要があります。次の列タイプは KEYCOLS でサポートされていません。 **KEYCOLS でサポートされていない Oracle 列タイプ**: 仮想列、UDT、ファンクションベース列、および Oracle GoldenGate 構成から明示的に除外されているすべての列 **KEYCOLS でサポートされていない SQL Server、DB2 LUW、DB2 z/OS、MySQL、SQL/MX、Teradata、TimesTen の列タイプ**: タイプスタンプまたは非マテリアライズド計算結果列を含む列、および Oracle GoldenGate 構成から除外されているすべての列。SQL Server の場合、Oracle GoldenGate は、主キーがないターゲット表の行のデータの長さの合計を、強制的に 8000 バイトより小さくします。 **KEYCOLS でサポートされていない Sybase 列タイプ**: 計算結果列、ファンクションベース列、および GoldenGate 構成から明示的に除外されているすべての列

MAPEXCEPTION の使用

MAPEXCEPTION では、REPERROR パラメータによって例外としてフラグ付けされている操作に対するマッピングおよびその他のオプションを指定します。MAPEXCEPTION によって、Replicat が失敗した操作を書き込み可能な*例外表*を指定し、マッピングおよび処理オプションを許可します。

MAPEXCEPTION は、ソース - ターゲット表マッピングおよびその他の標準の MAP オプションを含む同一の MAP 文内で使用できます。ソースおよびターゲット表名には、ワイルドカードを含めることができます。

MAPEXCEPTION を使用するには、同一の MAP 文内、または Replicat パラメータ・ファイルのルートで、REPERROR 文と EXCEPTION オプションを使用します。

構文

```
MAP <object spec>, TARGET <object spec>,
[<MAP_options>],
MAPEXCEPTION (TARGET <object spec> [, <exception_MAP_options>]);
```

引数	説明
TARGET	必須のキーワード。
<MAP_options>	必要な場合、(成功した) 非例外操作を処理する標準の MAP オプション。

引数	説明
<object spec>	例外表の完全修飾名。オブジェクト名には標準の Oracle GoldenGate ルールが適用されます。このドキュメントの MAP に関するガイドラインを参照してください。
<exception_MAP_options>	MAP パラメータの有効な任意のオプション。エラー処理のためにこれらのオプションが適用されます。非例外データをマップする場合は、MAPEXCEPTION 句の外でも MAP オプションを使用します。

例

次に、例外マッピングでの MAPEXCEPTION の使用例を示します。MAP および TARGET 句には、ワイルドカードのソースおよびターゲット表名が含まれています。名前が TRX で始まるすべての表を処理するときに発生する例外が、指定されたマッピングを使用して fin.trxexceptions 表に取得されます。

```
MAP src.trx*, TARGET trg.*,
MAPEXCEPTION (TARGET fin.trxexceptions,
COLMAP (USEDEFAULTS,
ACCT_NO = ACCT_NO,
OPTYPE = @GETENV ("LASTERR", "OPTYPE"),
DBERR = @GETENV ("LASTERR", "DBERRNUM"),
DBERRMSG = @GETENV ("LASTERR", "DBERRMSG")
)
);
```

REPERROR の使用

REPERROR では、エラーおよびレスポンスを指定し、MAP 文を実行したときに Replicat がエラーに対応する方法を制御します。REPERROR を MAP レベルで使用すると、パラメータ・ファイルのルート・レベルで REPERROR パラメータに設定されているグローバルなエラー処理ルール・セットよりも優先され、このルールを補完できます。自動化された包括的なエラー管理、および無停止のレプリケーション処理を行うために、複数の REPERROR 文を同一の MAP 文に適用できます。

構文

```
MAP <object spec>, TARGET <object spec>,
REPERROR (
{DEFAULT | DEFAULT2 | <SQL error> | <user-defined error>},
{ABEND | DISCARD | EXCEPTION | IGNORE |
RETRYOP [MAXRETRIES <n>] |
TRANSABORT [, MAXRETRIES] [, DELAYSECS <n> | DELAYCSECS <n>] |
TRANSDISCARD | TRANSEXCEPTION }
)
[, ...];
```

引数	説明
処理するエラーを指定するオプション:	
DEFAULT	明示的な REPERROR 文が指定されているエラーを除くすべてのエラーに対するグローバルなレスポンスを設定します。

引数	説明
DEFAULT2	DEFAULT のレスポンスが EXCEPTION に設定されている場合に、バックアップのデフォルト・アクションを提供します。DEFAULT2 は、エラーの発生が予想される MAP 文に例外 MAP 文が指定されていないときに使用します。
<SQL error>	SQL エラー番号。
<user-defined error>	MAP 文の FILTER 句の RAISEERROR オプションで指定されているユーザー定義エラー。
エラーに対するレスポンスを指定するオプション:	
ABEND	トランザクションをロールバックし、処理を異常終了します。ABEND はデフォルトです。
DISCARD	エラーを破棄ファイルに記録しますが、このトランザクションおよび後続のトランザクションの処理を継続します。
EXCEPTION	エラーを例外として処理します。エラーの発生が予想される場合は、このオプションを例外 MAP 文とともに使用するか、MAP の MAPEXCEPTION オプションとともに使用します。このオプションは、同じトランザクション内のエラーのない他の操作に影響しない個別の操作に対する例外を処理します。トランザクション・レベルで例外を処理するには、TRANSEXCEPTION オプションを使用します。エラー処理の構成方法の詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。 *注意*: 競合の検出および解決 (CDR) 機能がアクティブなとき、影響を受ける表への例外 MAP 文が存在する場合、CDR は、競合を引き起こすすべての操作を自動的に例外として処理します。この場合、REPERROR と EXCEPTION は必要ありませんが、CDR が解決できない競合を処理するその他のオプションとともに、または CDR に処理させない競合に対して、REPERROR を使用する必要があります。
IGNORE	エラーを無視します。
RETRYOP [MAXRETRIES <n>]	操作を再試行します。MAXRETRIES オプションでは、再試行回数を制御します。たとえば、表がエクステント不足の場合、RETRYOP と MAXRETRIES によって、トランザクションの失敗を防ぐためにエクステントを追加する時間を確保できます。指定した MAXRETRIES 回数の後、Replicat は異常終了します。 試行間隔を設定するには、312 ページで説明されている RETRYDELAY を設定します。
TRANSABORT [, MAXRETRIES] [, DELAYSECS <n> | DELAYCSECS <n>]	トランザクションを中止し、Replicat をトランザクションの開始位置に再配置します。この動作は、操作の処理が成功するか、MAXRETRIES が終了するまで継続されます。MAXRETRIES が設定されていない場合、TRANSABORT アクションは何度も繰り返されます。 DELAY オプションでは、再試行までの待機時間を指定します。

引数	説明
TRANSDISCARD	トランザクション内の任意の操作 (コミット操作など) が原因で、エラー指定されている Replicat エラーが発生する場合、レプリケートされたそのソース・トランザクション全体を破棄します。レコードでエラーが発生した場合、Replicat はトランザクションを中止し、DISCARDFILE パラメータで指定した破棄ファイルにそのレコードを書き込みます。Replicat はトランザクションをリプレイし、コミット・レコードを含むすべてのレコードを破棄ファイルに書き込みます。Replicat は、破棄処理が原因のエラーの発生時に異常終了します。 破棄レコードがすでにターゲット・レコードにデータ・マッピングされている場合、Replicat はターゲットのフォーマットで破棄ファイルに書き込み、それ以外の場合はソースのフォーマットで書き込みます。リプレイされたトランザクション自体は、常にソースのフォーマットで書き込まれます。 TRANSDISCARD は、コミット・エラーに加え、レコード・レベルのエラーもサポートします。このオプションの詳細は、299 ページの「REPERROR」を参照してください。
TRANSEXCEPTION	例外 MAP 文の MAPEXCEPTION または EXCEPTIONSONLY 句で定義したように、例外マッピング文に従って、レプリケートされたソース・トランザクションのすべてのレコードに、例外マッピングを実行します。レコードに対応する例外マッピング指定がない場合、または例外表への書込み時にエラーがある場合、Replicat はエラー・メッセージとともに異常終了します。 エラーが発生して TRANSEXCEPTION が使用されると、レコードでエラーが発生した場合、Replicat はトランザクションを中止し、DISCARDFILE パラメータで指定した破棄ファイルにそのレコードを書き込みます。 Replicat はトランザクションをリプレイし、ソース・レコードを調べて例外マッピング指定を探し、それを実行します。 TRANSEXCEPTION は、コミット・エラーに加え、レコード・レベルのエラーもサポートします。同じトランザクション内のエラーのない操作に影響せずに、レコード・レベルで (個別の SQL 操作に対して) エラーを処理するには、EXCEPTION オプションを使用します。 このオプションの詳細は、299 ページの「REPERROR」を参照してください。

例

次に、グローバルな REPERROR 文とともに、MAP 文で REPERROR を使用する様々な例を示します。

例 1:

```
REPLICAT <group>
REPERROR (<error1> , <response1>)
MAP <src1>, TARGET <tgt1>, REPERROR (<error1>, <response2>);
MAP <src2>, TARGET <tgt2>, REPERROR (<error2>, <response3>);
```

上記の例では、最初の MAP 文で error1 が発生した場合、優先される文が指定されているため、アクションは response1 ではなく response2 になります。ただし、2 番目の MAP 文で error1 が発生した場合には、グローバル・レスポンスの response1 がレスポンスになります。error2 に対するレスポンスは、MAP 固有の response3 になります。

例 2:

```
REPLICAT <group>
REPERROR (<error1> , <response1>)
MAP <src1>, TARGET <tgt1>, REPERROR (<error2>, <response2>),
REPERROR (<error3>, <response3>);
```

上記の例では、すべての REPERROR 文が異なるエラーに対応するため (MAP 固有の優先なし)、src1 から src2 にレプリケートするときに、すべてのエラーおよびアクション (1 ～ 3) が適用されます。

例 3:

```
REPLICAT <group>
REPERROR (<error1> , <response1>)
MAP <src1>, TARGET <tgt1>, REPERROR (<error1>, <response2>);
MAP <src2>, TARGET <tgt2>, REPERROR (<error2>, <response3>);

REPERROR (<error1> , <response4>)
MAP <src2>, TARGET <tgt2>, REPERROR (<error3>, <response3>);
```

上記の例では、最初の MAP 文で error1 が発生した場合、アクションは response2 になります。2 番目の文ではアクションは response1(グローバル・レスポンス) になり、3 番目の文では response4 になります (2 番目の REPERROR 文が適用されるため)。グローバル REPERROR 文は、パラメータ・ファイル内で別の REPERROR 文で新しいルールが開始されるまで、後続のすべての MAP 文に適用されます。

例 4:

```
REPERROR DEFAULT ABEND
REPERROR 1403 TRANSDISCARD.
MAP src, TARGET tgt, REPERROR(600 TRANSDISCARD);
```

上記の例では、ソース表 src をターゲット表 tgt に適用中にエラー 600 が発生した場合、トランザクション全体が破棄ファイルに書き込まれます。エラー 1403 が発生した場合も、グローバル REPERROR 指定に基づいて、同じアクションが行われます。その他のエラーが発生すると、プロセスは問題のあるレコードのみを破棄し、異常終了します。

RESOLVECONFLICT の使用

RESOLVECONFLICT では、双方向構成またはマルチマスター構成の MAP 文で、表に対して行われた操作における競合を、Replicat が処理する方法を指定します。競合および競合解決の詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』の第 9 章を参照してください。

RESOLVECONFLICT では、次の解決がサポートされています。

- INSERT の一意性の競合の解決。
- 行が存在しても、1 つ以上の列のビフォア・イメージがデータベース内の現在の値と異なる場合の、UPDATE の "no data found" 競合の解決。
- 行が存在しない場合の、UPDATE の "no data found" 競合の解決。
- 行が存在しても、1 つ以上の列のビフォア・イメージがデータベース内の現在の値と異なる場合の、DELETE の "no data found" 競合の解決。
- 行が存在しない場合の、DELETE の "no data found" 競合の解決。

複数の解決を同じ競合タイプに指定して、RESOLVECONFLICT にリストした順序で実行できます。複数の解決は、INSERTROWEXISTS 競合および UPDATEROWEXISTS 競合のみに限定されます。

RESOLVECONFLICT を MAP 文に複数回使用して、異なる競合タイプに異なる解決を指定できます。

BATCHSQL を使用している場合、競合検出はバッチ・モードで実行されますが、Replicat は、解決を実行するために GROUPTRANSOPS モードに戻ります。(詳細は、BATCHSQL パラメータを参照してください。)

サポートされているデータ型およびプラットフォーム

- RESOLVECONFLICT は、Windows および UNIX 用の Oracle GoldenGate でサポートされているすべてのデータベースをサポートしています。
- RESOLVECONFLICT を使用するには、データベースが Windows、Linux または UNIX システムに存在する必要があります。これは NonStop プラットフォーム上のデータベースではサポートされていません。
- CDR は、明示的な変換を行わずに、単純な SQL で比較できるデータ型をサポートしています。詳細は、個別のパラメータ・オプションを参照してください。
- LOB、抽象データ型 (ADT) またはユーザー定義型 (UDT) を含む列には、RESOLVECONFLICT を使用しないでください。

列値を CDR で使用可能にする方法

CDR を使用するには、更新および削除の完全イメージが必要です。

1. DELETE および UPDATE 操作のためにレコードのすべての列がトレイルに書き込まれ、Replicat が CDR で使用できるように、NOCOMPRESSDELETES および NOCOMPRESSUPDATES パラメータを使用します。
2. Extract TABLE パラメータの GETBEFORECOLS オプションを使用して、データベースが書き込むビフォア・イメージを取得し、それをトレイルに書込むように、Extract に指示します。DB2 データベースの場合、GETBEFORECOLS のかわりに、GETUPDATEBEFORES パラメータを使用します。
3. SQL/MX トランザクション・ログ (オーディット・トレイル) から完全イメージ・レコードを取得するには、no auditcompress の属性で表を作成または変更する必要があります。

エラー処理

HANDLECOLLSIONS、INSERTMISSINGUPDATES および REPERROR パラメータを使用する場合は、これらより前に競合解決を実行します。少なくとも、REPERROR を使用して、CDR が解決できないエラーを処理する必要があります。たとえば、CDR 解決が "no data found" エラーを返す場合、REPLICAT を、DISCARD に設定して失敗した操作を破棄ファイルに書き込むことも、EXCEPTION に設定して失敗した操作を同一の例外 MAP 文および CDR 競合に使用する例外表に書き込むこともできます。ビジネス・ルールを満たすのに優れている場合は、CDR を使用せずに、REPERROR を使用してエラーを処理することが適切な場合もあります。次に例を示します。

```
REPERROR (1403, EXCEPTION)
    RESOLVECONFLICT (UPDATEROWEXISTS, &
    (max_resolution_method, USEMAX (Modified_TS), COLS &
    (Address, Modified_TS)),(DEFAULT, OVERWRITE));
MAP source.Order, TARGET target.Order_ExceptionTable, EXCEPTIONSONLY, &
INSERTALLRECORDS;
```

競合解決の構成の詳細手順は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』の第 9 章を参照してください。

構文

```
MAP <source table>, TARGET <target table>,
RESOLVECONFLICT (
    {INSERTROWEXISTS |
    UPDATEROWEXISTS |
    UPDATEROWMISSING |
    DELETEROWEXISTS |
    DELETEROWMISSING}
    ( {DEFAULT | <resolution name>},
        {USEMAX (<res_col>) |
        USEMIN (<res_col>) |
        USEDELTA |
        DISCARD |
        OVERWRITE |
        IGNORE})
            [, COLS (<col>[,...])]
)
[RESOLVECONFLICT (, ...)]
```

引数	説明
INSERTROWEXISTS | UPDATEROWEXISTS | UPDATEROWMISSING | DELETEROWEXISTS | DELETEROWMISSING	この解決で処理する競合のタイプ。 INSERTROWEXISTS 挿入された行がターゲット上の一意性制約を違反しています。 UPDATEROWEXISTS 更新された行がターゲット上に存在しますが、1 つ以上の列には、データベース内の現在の値と異なるトレイル内のビフォア・イメージがあります。 UPDATEROWMISSING 更新された行がターゲットに存在しません。 DELETEROWEXISTS 削除された行がターゲットに存在しますが、1 つ以上の列には、データベース内の現在の値と異なるトレイル内のビフォア・イメージがあります。 DELETEROWMISSING 削除された行がターゲットに存在しません。

引数	説明
`DEFAULT | <resolution name>`	DEFAULT デフォルトの列グループ。DEFAULT 列グループに関連付けられている解決は、明示的に名前が付けられた列グループにないすべての列に使用されます。DEFAULT 列グループを定義する必要があります。 <resolution name> 特定の解決タイプにリンクしている特定の列グループの名前。解決タイプを識別する名前を指定します。有効な値は英数字です。空白および特殊文字は使用できませんが、アンダースコアは使用できます。たとえば、次のようになります。 `delta_res_method` 名前が付けられた解決または DEFAULT のいずれかを使用しますが、両方は使用しないでください。

`USEMAX (<res_col>) |`
`USEMIN (<res_col>) |`
`USEDELTA |`
`DISCARD |`
`OVERWRITE |`
`IGNORE`
	競合を解決するのに使用される競合ハンドラ・ロジック。有効な解決は次のとおりです。
	USEMAX
	トレイル・レコード内の <res_col> の値がデータベース内の列値より大きい場合、適切なアクションが実行されます。(「アクション」を参照してください)
	USEMIN
	トレイル・レコード内の <res_col> の値がデータベース内の列値より小さい場合、適切なアクションが実行されます。(「アクション」を参照してください)
	
	アクション:
	◆ (INSERTROWEXISTS 競合) トレイル・レコードを適用しますが、一意性違反を回避するために挿入を更新に変更し、既存の値を上書きします。
	◆ (UPDATEROWEXISTS 競合) トレイル・レコードから更新を適用します。

引数	説明
	<res_col> 解決列として機能する NOT NULL 列の名前 この列は、この解決に関連付けられている列グループの一部である必要があります。ターゲット・データベース内の現在の値と比較する解決列の値によって、解決の適用方法が決定されます。使用可能な場合、解決列のアフター・イメージが比較に使用されます。それ以外の場合、ビフォア・イメージの値が使用されます。単純な SQL で比較できる、次の列を使用します。 ◆ NUMERIC ◆ DATE ◆ TIMESTAMP ◆ CHAR/NCHAR ◆ VARCHAR/NVARCHAR 最新のタイムスタンプの解決を使用するには、タイムスタンプ列を <res_col> として使用し、行が挿入または更新されたとき、タイムスタンプ列を現在の時刻に設定します。可能な場合、秒の端数をサポートする SYSTIMESTAMP データ型で、解決列を定義します。サブ秒の粒度で比較を実行すると、解決列の値がトレイル内とターゲット内で同じであるケースを解決するタイブレーク競合ハンドラは、ほとんど必要ありません。タイムスタンプ列の値が (解決に応じて) 増加のみまたは減少のみであることが確認されている場合、USEMAX および USEMIN によってデータ相違が発生することはありません。 USEDELTA (UPDATEROWEXISTS 競合のみ) ターゲット・データベース内の列の現在の値に、トレイル・レコード内の前後の値の差異を追加します。いずれかの値が NULL の場合、エラーが発生します。USEDELTA は、NUMERIC データ型を含む列に基づきます。USEDELTA は、複数のノードで行が同時に更新される場合の複数ノード構成で役立ちます。すべてのノードが同期されるように、列値の差異のみを他のノードに伝播します。 DISCARD (すべての競合タイプに有効) ターゲット・データベース内の現在の値を保持し、トレイル・レコード内のデータを破棄ファイルに書き込みます。DISCARDFILE パラメータで、破棄ファイルを指定しておく必要があります。 データ相違が発生する可能性があるため、DISCARD は慎重に使用してください。

引数	説明
	OVERWRITE (DELETEROWMISSING を除くすべての競合タイプに有効) 次のようにトレイル・レコードを適用します。 ◆ (INSERTROWEXISTS 競合) トレイル・レコードを適用しますが、一意性違反を回避するために挿入を更新に変更し、既存の値を上書きします。 ◆ (UPDATEROWEXISTS 競合) トレイル・レコードから更新を適用します。 ◆ (UPDATEROWMISSING 競合) トレイル・レコードを適用しますが、更新を挿入に変更します。更新を挿入に変換するには、行のすべての列のビフォア・イメージがトレイル内で使用できる必要があります。データベースがデフォルトでビフォア・イメージを記録しない場合、サプリメンタル・ロギングを使用し、Extract GETBEFORECOLS パラメータに ALL を指定します。 ◆ (DELETEROWEXISTS 競合) トレイル・レコードから削除を適用しますが、WHERE 句の主キー列のみを使用します。 データ相違が発生する可能性があるため、OVERWRITE は慎重に使用してください。 IGNORE (すべての競合タイプに有効) ターゲット・データベース内の現在の値を保持し、トレイル・レコードを無視します。ターゲット表または破棄ファイルに適用しないでください。
`COLS (<col>[,...])`	非デフォルト列グループ。これは、特定の解決タイプにリンクし、特定の解決タイプによって操作される、ターゲット・データベース内の列のリスト (マッピング後) です。競合に指定される列グループがない場合、すべての列が、特定の競合に指定される解決の影響を受けます。 または、別の列グループにリストされていないすべての列を含める DEFAULT 列グループを指定できます。DEFAULT オプションを参照してください。 複数の列グループに、それぞれ異なる解決を指定できます。たとえば、col2 および col3 に OVERWRITE を使用し、col4 に USEDELTA を使用できます。いずれかのグループに指定されている列は、他のグループには指定できません。異なる列グループでの列の競合は、指定した解決に従って、リストされた順序で、個別に解決されます。

引数	説明
	列グループは次のように機能します。 ◆ INSERTROWEXISTS および UPDATEROWEXISTS 競合の場合、異なる列グループを使用して、表ごとに複数の競合タイプおよび解決を指定できます。異なる列グループでの列の競合は、列グループに指定した競合解決方法に従って、個別に解決されます。 ◆ UPDATEROWMISSING、DELETEROWEXISTS および DELETEROWMISSING の場合、1 つの列グループのみを使用でき、表のすべての列は、この列グループにある必要があります (*DEFAULT* 列グループとみなされます)。

SQLEXEC の使用

SQLEXEC では、Oracle GoldenGate 処理中に MAP 文内から SQL ストアド・プロシージャまたは問合せを実行します。SQLEXEC により、Oracle GoldenGate はデータベースと直接通信し、データベースによってサポートされているファンクションを実行できます。データベース・ファンクションは、列変換のための値取得などの同期プロセスの一部として使用することも、データの抽出またはレプリケーションと関係なく使用することもできます。

MAP 文内で使用する場合、実行されるプロシージャまたは問合せは、ソースまたはターゲット行から入力パラメータを受け付け、出力パラメータを渡すことができます。

注意 **問合せまたはプロシージャは、SQLEXEC 文の実行時に正しく構築されている必要があります。Replicat は、問合せまたはプロシージャの問題を検出すると、設定されているエラー処理ルールにかかわらず、即座に異常終了します。**

サポートされるデータ型

入力パラメータおよび出力パラメータとして SQLEXEC でサポートされるデータ型は次のとおりです。

- 数値データ型
- 日付データ型
- 文字データ型

Oracle GoldenGate でのストアド・プロシージャおよび問合せの使用の詳細は、『Oracle GoldenGate*Windows and UNIX 管理者ガイド*』を参照してください。

SQLEXEC の依存関係および制約事項

- SQL は、Oracle GoldenGate プロセスを実行しているデータベース・ユーザーによって実行されます。このユーザーは、ストアド・プロシージャの実行、およびデータベース提供のプロシージャのコール権限を持っている必要があります。
- 問合せまたはプロシージャは、SQLEXEC 文の実行時に、データベースの有効な SQL 構文を使用して正しく構築されている必要があり、そうでない場合、Replicat は、設定されているエラー処理ルールにかかわらず異常終了します。許可されている SQL 構文の詳細は、データベース・ベンダーによって提供されている SQL リファレンス・ガイドを参照してください。

- SQLEXEC は、主キー列の値を変更するために使用しないでください。主キーの値は、Extract から Replicat に渡されます。主キーの値がない場合、Replicat 操作は完了できません。主キーの値を SQLEXEC で変更する必要がある場合は、元のキーの値を別の列にマッピングした後、KEYCOLS オプションでこの列を代替キーとして定義することにより、エラーを回避できます。256 ページの「KEYCOLS の使用」を参照してください。
- Oracle GoldenGate は、z/OS 上の DB2 に対して、ODBC SQLExecDirect ファンクションを使用して SQL 文を動的に実行します。つまり、接続先のデータベース・サーバーは、SQL 文を動的に準備できる必要があります。ODBC は、実行のたびに (リクエストされる間隔で)SQL 文を準備します。通常は、このことが Oracle GoldenGate ユーザーの問題になることはありません。詳細は、z/OS 上の DB2 のマニュアルを参照してください。

Oracle GoldenGate を使用して DDL をレプリケートする場合は、SQL の実行前に、ストアド・プロシージャまたは問合せに影響を受けるすべてのオブジェクトが、正しい構造で存在している必要があります。したがって、これらのオブジェクトの構造に影響する DDL (CREATE や ALTER など) は、SQLEXEC の実行前に実行される必要があります。

SQLEXEC とストアド・プロシージャの使用

MAP 文内からストアド・プロシージャを実行するには、SPNAME 句を使用します。

構文

```
SQLEXEC (
SPNAME <sp name>
[, ID <logical name>]
{, PARAMS <param spec>| NOPARAMS}
[, <option>] [, ...]
)
```

コンポーネント	説明
`<sp name>`	実行するプロシージャ名を指定します。
`ID <logical name>`	プロシージャの論理名を定義します。このオプションは、MAP 文内でプロシージャを複数回実行するときに使用します。1 つの MAP 文で、最大 20 のストアド・プロシージャを実行できます。プロシージャの実行が 1 回の場合、ID は必要ありません。
`PARAMS <param spec> | NOPARAMS`	プロシージャがパラメータを受け付けるかどうかを定義します。PARAMS <param spec> または NOPARAMS のいずれかを使用する必要があります。<param spec> では、入力パラメータおよび入力のソースを定義します。

`<option>`	ストアド・プロシージャの影響を制御するために、単独または他のオプションと組み合せて使用できる、次のいずれかのオプションを指定します。
	`AFTERFILTER | BEFOREFILTER`
	`ALLPARAMS`
	`DBOP`
	`ERROR`
	`EXEC`
	`MAXVARCHARLEN`
	`PARAMBUFSIZE`
	`TRACE`

SQLEXEC コンポーネントの説明は、アルファベット順に 271 ページから記載されています。

SQLEXEC と問合せの使用

MAP 文内から問合せを実行するには、ID および QUERY 句を使用します。

構文

```
SQLEXEC (
ID <logical name>
, QUERY "<sql query>"
{, PARAMS <param spec>| NOPARAMS}
[, <option>] [, ...]
)
```

コンポーネント	説明
`ID <logical name>`	問合せの論理名を定義します。問合せの結果から値を抽出するには、論理名が必要です。ID <logical name> は、問合せから返された列値を参照します。
`QUERY "<sql query>"`	データベースに対して実行する SQL 問合せの構文を指定します。問合せは、問合せを実行するデータベースの有効な標準問合せ言語を使用している必要があります。 問合せは、SELECT 文の結果を返すか、INSERT、UPDATE、または DELETE 文を実行できます。SELECT 文の出力を生成する問合せの場合は、SELECT によって最初に返される行のみが処理されます。SELECT 文には "INTO ..." 句を指定しないでください。 問合せは 1 行に収め、引用符で囲む必要があります。 SQLEXEC 問合せ内に引用符で囲まれたオブジェクト名を使用するには、SQL 問合せを二重引用符ではなく一重引用符で囲む必要があり、GLOBALS ファイルで USEANSISQLQUOTES パラメータを使用して、オブジェクトおよびリテラル識別子に SQL-92 ルールを施行する必要があります。次に、問合せ内に引用符で囲まれたオブジェクト名を使用する例を示します。 `SQLEXEC 'SELECT "col1" from "schema"."table"'` 最良の結果を得るために、各開始引用符の後ろおよび各終了引用符の前に空白を入力してください。
`PARAMS <param spec>| NOPARAMS`	問合せがパラメータを受け付けるかどうかを定義します。これらのオプションのいずれかを使用する必要があります。<param spec> では、入力パラメータおよび入力ソースを定義します。

`<option>`	問合せの影響を制御するために、単独または他のオプションと組み合せて使用できる、次のいずれかのオプションを指定します。
	`AFTERFILTER | BEFOREFILTER`
	`ALLPARAMS`
	`DBOP`
	`ERROR`
	`EXEC`
	`MAXVARCHARLEN`
	`PARAMBUFSIZE`
	`TRACE`

SQLEXEC コンポーネントの説明は、アルファベット順に 271 ページから記載されています。

入力パラメータ用のプレースホルダの使用

ほとんどの問合せは、入力パラメータ用のプレースホルダを必要とします。問合せ内でのパラメータの指定方法は、データベースのタイプによって異なります。

- Oracle の場合は、次の例のように、入力パラメータはコロン (:) を使用して指定し、その後にパラメータ名を続けます。

  ```
  "SELECT NAME FROM ACCOUNT WHERE SSN = :SSN AND ACCOUNT = :ACCT"
  ```

- 他のデータベースの場合は、次の例のように、入力パラメータは疑問符を使用して指定します。

  ```
  "SELECT NAME FROM ACCOUNT WHERE SSN = ? AND ACCOUNT = ?"
  ```

どのデータベースでも、パラメータ名を引用符で囲む必要はありません。

パラメータの値渡し

Oracle GoldenGate は、入力値および出力値をプロシージャまたは問合せとやり取りするためのオプションを提供しています。

- ストアド・プロシージャまたは問合せ内で入力パラメータにデータ値を渡すには、SQLEXEC の PARAMs オプションを使用します (276 ページを参照してください)。
- ストアド・プロシージャまたは問合せから入力として値を FILTER または COLMAP 句に渡すには、次の構文を使用します。

  ```
  {<procedure name> | <logical name>}.<parameter>
  ```

  条件 :

  - <procedure name> は、ストアド・プロシージャの実際の名前で、SQLEXEC 文の SPNAME に指定している値と一致する必要があります。この引数は、Oracle GoldenGate 実行中にプロシージャを 1 回実行する場合にのみ指定します。
  - <logical name> は、SQLEXEC 文の ID オプションで指定した論理名です。この引数は、MAP 文内でプロシージャを複数回実行する場合に、問合せまたはストアド・プロシージャのインスタンスから値を渡すときに使用します。
  - <parameter> は、参照表の列などのパラメータ名か、返された値を抽出する場合の RETURN_VALUE のいずれかです。

上記の構文のかわりとして、@GETVAL ファンクションを使用できます。詳細は、457 ページを参照してください。

入力パラメータのネーミングには、次のような異なる構成があります。

- Oracle は、次の例のように、入力パラメータに論理名を付けることを許可します。

  ```
  SQLEXEC (ID appphone, QUERY " select per_type from ps_personal_data "
      " where emplid = :vemplid "
      " and per_status = 'N' and per_type = 'A' ",
      PARAMS (vemplid = emplid)),
  TOKENS (applid = @GETVAL(appphone.per_type));
  ```

- 他のデータベースでは、入力パラメータに、P1、P2 のような入力パラメータごとに番号を増分させた名前を付ける必要があります。データベースによっては "p" を大文字または小文字にする必要があることに注意します。このタイプの入力パラメータの例を次に示します。

```
SQLEXEC (ID appphone, QUERY " select per_type from ps_personal_data "
    " where emplid = ? "
    " and per_status = 'N' and per_type = 'A' ",
    PARAMS (p1 = emplid)),
TOKENS (applid = @GETVAL(appphone.per_type));
```

例

次に、Oracle のソース表、ターゲット表、および参照表と、これらの表のパラメータをストアド・プロシージャの単一のインスタンス、およびストアド・プロシージャの複数のインスタンスに渡す方法の例を示します。

ソース表 "cust":

```
custid                    Number
current_residence_state   Char(2)
birth_state               Char(2)
```

ターゲット表 "cust_extended":

```
custid                          Number
current_residence_state_long    Varchar(30)
birth_state_long                Varchar(30)
```

参照表 "state_lookup"

```
abbreviation     Char(2)
long_name        Varchar(30)
```

例 1

次に、参照表から値を取得するために 1 回実行されるストアド・プロシージャの使用例を示します。この値は、COLMAP 文のターゲット列にマッピングされます。

```
MAP sales.cust, TARGET sales.cust_extended, &
SQLEXEC (SPNAME lookup, &
PARAMS (long_name = birth_state)), &
COLMAP (custid = custid, &
birth_state_long = lookup.long_name);
```

例 2

次に、参照表から値を取得するストアド・プロシージャを複数回実行する例を示します。値は、ターゲット列にマッピングされます。

```
MAP sales.cust, TARGET sales.cust_extended, &
SQLEXEC (SPNAME lookup, ID lookup1, &
PARAMS (long_name = current_residence_state)), &
SQLEXEC (SPNAME lookup, ID lookup2, &
PARAMS (long_name = birth_state)), &
COLMAP (custid = custid, current_residence_state_long = lookup1.long_name, &
birth_state_long = lookup2.long_name);
```

AFTERFILTER および BEFOREFILTER の使用

AFTERFILTER および BEFOREFILTER では、MAP 文の FILTER 句との関連でストアド・プロシージャまたは問合せをいつ実行するかを指定します。

構文　AFTERFILTER | BEFOREFILTER

ルール	説明
AFTERFILTER	FILTER 文の後に SQL 文を実行します。これにより、フィルタが成功しない場合に SQL 実行のオーバーヘッドを回避できます。これはデフォルトです。
BEFOREFILTER	SQL の結果をフィルタで使用できるように、FILTER 文の前に SQL を実行します。

例　SQLEXEC (SPNAME check, NOPARAMS, BEFOREFILTER)

ALLPARAMS の使用

ALLPARAMS は、ストアド・プロシージャまたは問合せの実行のために、指定されたすべてのパラメータが存在している必要があるかどうかを決定するグローバル・ルールとして使用します。ALLPARAMS で設定するグローバル・ルールよりも、PARAMS 句内で設定する個別のパラメータのルールのほうが優先されます。

構文　ALLPARAMS {OPTIONAL | REQUIRED}

ルール	説明
OPTIONAL	すべてのパラメータが存在しているかどうかにかかわらず、SQL の実行を許可します。これはデフォルトです。
REQUIRED	SQL を実行するために、すべてのパラメータが存在している必要があります。

例
```
SQLEXEC (SPNAME lookup,
PARAMS (long_name = birth_state, short_name = state),
ALLPARAMS OPTIONAL)
```

DBOP の使用

DBOP では、ストアド・プロシージャまたは問合せ内で実行された INSERT、UPDATE、DELETE、および SELECT 文をコミットします。そうしないと、これらの文はロールバックされる可能性があります。Oracle GoldenGate は、ソース・トランザクションと同じトランザクション境界内でコミットを発行します。

警告　**データベース、特に本番環境のデータベースに対して SQLEXEC プロシージャを実行する場合は、注意して使用してください。プロシージャによってコミットされた変更は、既存のデータを上書きすることがあります。**

構文　DBOP

例　SQLEXEC (SPNAME check, NOPARAMS, DBOP)

ERROR の使用

ERROR では、ストアド・プロシージャまたは問合せに関連するエラーに対するレスポンスを定義します。明示的なエラー処理の指定がない場合、Oracle GoldenGate プロセスはエラーを検出すると異常終了します。プロシージャからプロセスにエラーを返させ、ERROR でレスポンスを指定するようにしてください。

構文　ERROR <action>

アクション	説明
IGNORE	Oracle GoldenGate に、ストアド・プロシージャまたは問合せに関連するすべてのエラーを無視させ、処理を継続させます。このパラメータ抽出結果は、" 列行方不明 " 状態になります。これはデフォルトです。
REPORT	ストアド・プロシージャまたは問合せに、関連するすべてのエラーを破棄ファイルにレポートさせます。(エラーをレポートさせるには、DISCARDFILE パラメータで、破棄ファイルを指定しておく必要があります。) このレポートは、エラーの原因の追跡に役立ちます。ここには、エラーの説明と、プロシージャまたは問合せとやり取りしたパラメータの値両方が含まれます。Oracle GoldenGate で、エラーのレポート後、処理が続けられます。
RAISE	REPERROR パラメータで設定されたルールに従ってエラーを処理します。Oracle GoldenGate は、エラーを処理する前に、現在の MAP 文に関連する他のストアド・プロシージャまたは問合せの処理を継続します。
FINAL	RAISE と似ていますが、プロシージャまたは問合せに関連するエラーを検出したときに、残りのストアド・プロシージャおよび問合せがバイパスされます。エラー処理は、エラーの直後に起動されます。
FATAL	プロシージャまたは問合せに関連するエラーを検出したときに、即座に Oracle GoldenGate を異常終了させます。

例

次に、RAISE オプションを使用する Oracle の例外文の例を示します。Oracle のファンクション、RAISE_APPLICATION_ERROR を使用します。Replicat パラメータ・ファイルで REPERROR (-20000, DISCARD) を使用してエラーが定義されている場合、Oracle GoldenGate はこのレコードを破棄し、処理を継続します。

```
EXCEPTION
WHEN no_match_rec THEN
RAISE_APPLICATION_ERROR(-20000, 'No Matching Update In Target');
```

EXEC の使用

EXEC では、MAP 文のストアド・プロシージャまたは問合せを実行する頻度、および出力パラメータを抽出する場合に結果を有効とみなす期間を制御します。

構文

```
EXEC <frequency>
```

頻度	説明
MAP	プロシージャまたは問合せが指定されている各ソース - ターゲットのマップで、プロシージャまたは問合せを 1 回実行します。MAP を使用する場合、同一のソース表を持つそれ以降のマップでは結果は無効になります。たとえば、ソース表が複数のターゲット表と同期を取る場合、結果は最初のソース - ターゲットのマップでのみ有効です。MAP はデフォルトです。

頻度	説明
ONCE	Oracle GoldenGate の実行中、関連する MAP 文の最初の呼び出し時にプロシージャまたは問合せを 1 回実行します。結果は、プロセスが実行しているかぎり有効です。
TRANSACTION	プロシージャまたは問合せをソース・トランザクションで 1 回実行します。結果は、トランザクションのすべての操作に有効です。
SOURCEROW	プロシージャまたは問合せをソース行操作で 1 回実行します。このオプションは、ソース表を複数のターゲット表と同期し、プロシージャまたは問合せの結果がソース - ターゲット・マッピングのたびに呼び出される場合に使用します。

例 1 次に、ONCE の使用例を示します。

```
MAP sales.cust, TARGET sales.cust_extended, &
SQLEXEC (SPNAME lookup, PARAMS (long_name = birth_state), EXEC ONCE), &
COLMAP (custid = custid, &
birth_state_long = lookup.long_name);
```

例 2 次に、TRANSACTION の使用例を示します。

```
MAP sales.cust, TARGET sales.cust_extended, &
SQLEXEC (SPNAME lookup, PARAMS (long_name = birth_state), EXEC TRANSACTION), &
COLMAP (custid = custid, &
birth_state_long = lookup.long_name);
```

例 3 次に、デフォルト (MAP) の誤った使用例を示します。2 つの MAP 文で、同一のソース表と 2 つの異なるターゲット表を同期します。しかし、プロシージャ lookup の結果は、2 番目の MAP の実行までに無効になってしまい、2 番目の MAP は " 列行方不明 " の状態になります。この機能を正しく実装するには、SOURCEROW を使用する必要があります。

```
MAP sales.srctab, TARGET sales.targtab, &
SQLEXEC (SPNAME lookup, PARAMS (param1 = srccol)), &
COLMAP (targcol = lookup.param2);

MAP sales.srctab, TARGET sales.targtab2, &
COLMAP (targcol2 = lookup.param2);
```

例 4 次に、SOURCEROW の使用例を示します。このケースでは、プロシージャが各ソース行操作で実行されるため、2 番目の MAP は有効な値を返します。

```
MAP sales.srctab, TARGET sales.targtab, &
SQLEXEC (SPNAME lookup, PARAMS (param1 = srccol), EXEC SOURCEROW), &
COLMAP (targcol = lookup.param2);

MAP sales.srctab, TARGET sales.targtab2, &
COLMAP (targcol2 = lookup.param2);
```

ID の使用

ID は、MAP 文内の問合せおよびストアド・プロシージャに対して、次のように使用します。

- 問合せでは、ID <logical name> を使用し、Oracle GoldenGate がこの名前を使用して問合せから返される列値を参照できるようにします。
- ストアド・プロシージャでは、ID <logical name> を使用して、たとえば 2 つの異なる列のマップなどのために、1 つの MAP 文内でプロシージャを複数回呼び出せるようにします。それ以外の場合、ID は必要ありません。1 つの MAP 文で、最大 20 のストアド・プロシージャを実行できます。これらは、パラメータ・ファイルにリストされた順に実行されます。

構文

```
ID <logical name>
```

コンポーネント	説明
<logical name>	ストアド・プロシージャまたは問合せの論理名。たとえば、"lookup" という名前のプロシージャの論理名を "lookup1"、"lookup2" などのようにできます。

例 1 次に、ID <logical name> の使用例を示します。各列マップが、lookup1 および lookup2 を使用して lookup という名前のストアド・プロシージャを個別にコールし、それぞれの結果を参照します。

```
MAP sales.srctab, TARGET sales.targtab, &
SQLEXEC (SPNAME lookup, ID lookup1, PARAMS (param1 = srccol)), &
COLMAP (targcol1 = lookup1.param2), &
SQLEXEC (SPNAME lookup, ID lookup2, PARAMS (param1 = srccol)), &
COLMAP (targcol = lookup2.param2);
```

例 2 次に、lookup という名前のストアド・プロシージャを 1 回実行する例を示します。このケースでは、実際のプロシージャ名を使用します。論理名は必要ありません。

```
MAP sales.tab1, TARGET sales.tab2, &
SQLEXEC (SPNAME lookup), PARAMS (param1 = srccol)), &
COLMAP (targcol = lookup.param1);
```

例 3 次に、Oracle および SQL Server の問合せにそれぞれ ID <logical name> を使用する例を示します。この例では、このドキュメントのスペースの制約によって、SQLEXEC 文が複数の行にまたがっています。実際の SQLEXEC 文は、1 行内に含める必要があります。

```
MAP sales.account, TARGET sales.newacct,
SQLEXEC (ID lookup,
QUERY "select desc_col into desc_param from lookup_table
" where code_col = :code_param ",
PARAMS (code_param = account_code)),
COLMAP (newacct_id = account_id,
newacct_val = lookup.desc_param);

MAP sales.account, TARGET sales.newacct,
SQLEXEC (ID lookup,
QUERY "select desc_col into desc_param from lookup_table
where code_col = ?",
PARAMS (p1 = account_code)),
COLMAP (newacct_id = account_id,
newacct_val = lookup.desc_param);
```

MAXVARCHARLEN の使用

MAXVARCHARLEN では、ストアド・プロシージャまたは問合せの出力パラメータに割り当てる最大長を指定します。この最大値を超える出力値は切り捨てられます。

構文 MAXVARCHARLEN <num bytes>

コンポーネント	説明
<num bytes>	出力パラメータに許可する最大バイト数を定義します。明示的に MAXVARCHARLEN 句を使用しない場合のデフォルトは、255 バイトです。

例 MAXVARCHARLEN 100

NOPARAMS の使用

NOPARAMS は、ストアド・プロシージャまたは問合せがパラメータを必要としない場合に、PARAMS のかわりに使用します。PARAMS 句または NOPARAMS のいずれかを指定する必要があります。

構文 NOPARAMS

例 SQLEXEC (SPNAME check, NOPARAMS)

PARAMBUFSIZE の使用

PARAMBUFSIZE では、入力および出力パラメータを含むパラメータ情報を保持するメモリー・バッファの最大値を指定します。Oracle GoldenGate は、パラメータに割り当てられたメモリーと最大値との差が 500 バイト以内に達するたびに警告を発行します。

構文 PARAMBUFSIZE <num bytes>

コンポーネント	説明
(<num bytes>)	メモリー・バッファに許可する最大バイト数を定義します。明示的に PARAMBUFSIZE 句を使用しない場合のデフォルトは、10,000 バイトです。

例 PARAMBUFSIZE 15000

PARAMS の使用

PARAMS では、入力を受け付けるストアド・プロシージャまたは問合せのパラメータ名と、入力を提供するソース列名または Oracle GoldenGate 列変換ファンクション名を指定します。PARAMS 句または NOPARAMS のいずれかを指定する必要があります。

デフォルトでは、Oracle GoldenGate は二重引用符で囲まれた文字列をリテラルとして処理します。列名に二重引用符、およびリテラルに一重引用符 (SQL-92 ルール) を使用するには、GLOBALS パラメータ・ファイルで USEANSISQLQUOTES パラメータを使用します。

次に、SQLEXEC によってサポートされているデータベースと、入力および出力パラメータとしてサポートされているデータ型を示します。

- 数値データ型
- 日付データ型
- 文字データ型

デフォルトでは、出力パラメータは、パラメータ当たり 255 バイトで切り捨てられます。プロシージャがこれより長いパラメータを必要とする場合は、MAXVARCHARLEN オプションを使用します。

構文

```
PARAMS (
[OPTIONAL | REQUIRED]
<param name> = {<source column> | <GG function>}
[, ...]
)
```

コンポーネント	説明
`OPTIONAL | REQUIRED`	パラメータ値が見つからないときに、プロシージャまたは問合せを実行するかどうかを決定します。 ◆ OPTIONAL は、SQL の実行にパラメータ値が必要ないことを示します。必要なソース列がデータベース操作で見つからない場合、またはソース列が見つからないために列変換ファンクションを完了できない場合でも、SQL は実行されます。 OPTIONAL は、Oracle 以外のすべてのデータベースのデフォルトです。Oracle では、ストアド・プロシージャ定義を取得するときに、パラメータがオプションかどうかが自動的に判別されます。 ◆ REQUIRED は、パラメータ値が存在している必要があることを示します。パラメータ値が存在しない場合、SQL は実行されません。
`<param name> = { <source column> | <GG function }`	パラメータ名を、入力を提供する列またはファンクションにマップします。 条件 : ◆ <param name> は、次のいずれかです。 ストアド・プロシージャの場合、ストアド・プロシージャ内の入力を受け付ける任意のパラメータ名 (参照表の列など) です。 Oracle 問合せの場合、先行するコロンを*除いた*、問合せの任意の入力パラメータ名です。たとえば、:param1 の場合は、PARAMS 句では param1 と指定します。 Oracle 以外の問合せの場合は P*n* (*n* は 1 から始まるパラメータの番号) です。たとえば、パラメータが 2 個の問合せでは、<param name> エントリは p1 および p2 です。データベースによっては "p" を大文字または小文字にする必要があることに注意します。 ◆ <source column> は、ソース列名です。デフォルトでは、指定した列が (圧縮更新であるために) ログに存在しない場合、このパラメータはプロシージャまたは問合せでこのパラメータに指定されているデフォルト値を使用します。 ◆ <GG function> は、Oracle GoldenGate 列変換ファンクション名です。列変換ファンクションの詳細は、第 5 章を参照してください。

例

次に、account 表からターゲット表 newacct にデータをマップする例を示します。account 表からのレコードを処理するとき、Oracle GoldenGate は、列マップを実行する前に lookup ストアド・プロシージャを実行します。プロシージャの code_param パラメータは、account_code ソース列から入力を受け付けます。

```
MAP sales.account, TARGET sales.newacct, &
SQLEXEC (SPNAME lookup, PARAMS (code_param = account_code)), &
COLMAP (newacct_id = account_id, &
newacct_val = lookup.desc_param);
```

TRACE の使用

TRACE では、入力および出力パラメータをレポート・ファイルに記録します。

次に、SQLEXEC トレースを有効化している場合の破棄ファイルの例を示します。

```
    Input parameter values...

    LMS_TABLE: INTERACTION_ATTR_VALUES
       KEY1: 2818249
      KEY2: 1
    Report File:

    From Table MASTER.INTERACTION_ATTR_VALUES to MASTER.INTERACTION_ATTR_VALUES:
           #  inserts:      0
           #       updates:         0
           #       deletes:         0
           #  discards:     1

      Stored procedure GGS_INTERACTION_ATTR_VALUES:
           attempts:        2
           successful:      0
```

構文

```
TRACE {ALL | ERROR}
```

アクション	説明
ALL	呼び出された各プロシージャまたは問合せの入力および出力パラメータをレポート・ファイルに書き込みます。これはデフォルトです。
ERROR	SQL エラーの発生後にのみ、呼び出された各プロシージャまたは問合せの入力および出力パラメータをレポート・ファイルに書き込みます。

例

```
SQLEXEC (SPNAME lookup, PARAMS (long_name = birth_state,
short_name = state), TRACE ERROR)
```

TARGETDEF の使用

TARGETDEF では、ターゲット定義テンプレートを指定します。定義テンプレートは、特定のターゲット表に対して DEFGEN が実行されるときに、このオブジェクトの定義に基づいて作成されます。テンプレートが作成されると、この表と同一の定義を持つ新しいターゲット表は、DEFGEN を実行せずに、かつ Replicat の停止と起動を伴わずに追加できます。TARGETDEF で指定されたテンプレートの定義は、定義の参照に使用されます。DEFGEN の詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。

構文　　MAP <table spec>, TARGET <table spec>, TARGETDEF <definitions template>;

引数	説明
<definitions template>	DEFGEN パラメータ・ファイルの TABLE の DEF オプションで指定されている定義テンプレート名。テンプレートに含まれる定義は、この MAP 文の表の定義と同一である必要があります。

例　　MAP acct.cust*, TARGET acc.cust*, DEF custdef, TARGETDEF tcustdef;

TRIMSPACES および NOTRIMSPACES の使用

TRIMSPACES および NOTRIMSPACES では、ソースの CHAR 列の末尾の空白を、ターゲットの CHAR または VARCHAR 列に適用するときに切り捨てるかどうかを制御します。デフォルトは TRIMSPACES です。

注意　Sybase は、すべての CHAR 型を VARCHAR 型として処理し、このため TRIMSPACES は無効になります。Sybase では、TRIMVARSPACES パラメータを使用します。

TRIMSPACES および NOTRIMSPACES は、異なる MAP 文または文グループに対する切捨て機能を有効化または無効化するために、パラメータ・ファイルのルート・レベルでも使用できます。

構文　　MAP <table spec>, TARGET <table spec>, {TRIMSPACES | NOTRIMSPACES};

例　　次に、最初の 2 つのターゲットに対してはデフォルトの末尾空白の切捨てを行い、最後の 2 つのターゲットでは末尾空白の切捨てを行わない例を示します。

```
MAP fin.src1, TARGET fin.tgt1;
MAP fin.src1, TARGET fin.tgt2;
MAP fin.src1, TARGET fin.tgt3, NOTRIMSPACES;
MAP fin.src1, TARGET fin.tgt4, NOTRIMSPACES;
```

TRIMVARSPACES および NOTRIMVARSPACES の使用

TRIMVARSPACES および NOTRIMVARSPACES では、ソースの VARCHAR 列の末尾の空白を、ターゲットの CHAR または VARCHAR 列に適用するときに切り捨てるかどうかを制御します。

VARCHAR 列の空白は実際はデータの一部であるため、デフォルトは NOTRIMVARSPACES です。TRIMVARSPACES を使用する前に、末尾の空白がターゲット・データの不可欠な部分ではないことを確認してください。

TRIMVARSPACES および NOTRIMVARSPACES は、異なる MAP 文または文グループに対する切捨て機能を有効化または無効化するために、パラメータ・ファイルのルート・レベルでも使用できます。

構文　　MAP <table spec>, TARGET <table spec>, {TRIMVARSPACES | NOTRIMVARSPACES};

例　次に、最後の 2 つのターゲットに対してのみ、末尾の空白を切り捨てる例を示します。

```
MAP fin.src1, TARGET fin.tgt1;
MAP fin.src1, TARGET fin.tgt2;
MAP fin.src1, TARGET fin.tgt3, TRIMVARSPACES;
MAP fin.src1, TARGET fin.tgt4, TRIMVARSPACES;
```

WHERE の使用

WHERE では、条件文に基づいてレコードを選択します。WHERE 句の使用および使用可能な列データの詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。

Oracle GoldenGate では、マルチバイト文字セットまたはローカル・オペレーティング・システムの文字セットと互換性のない文字セットを含む列に対して、WHERE をサポートしていません。

GLOBALS ファイルで USEANSISQLQUOTES パラメータが使用されていない場合、WHERE 内で使用するリテラル文字列は二重引用符で囲む必要があります。このパラメータは、識別子およびリテラルに SQL-92 ルールを施行します。

構文

```
MAP <table spec>, TARGET <table spec>,
WHERE (<where clause>);
```

コンポーネント	説明
<where clause>	次の例のように、条件に基づいてレコードを選択します。 WHERE (branch = "NY") 次の表に、許可されている WHERE 演算子を示します。 WHERE は、主キー更新操作の一部として条件文の主キー列のビフォア・イメージの評価をサポートしていません。

表 37　許可されている WHERE 演算子

演算子	例 :
列名	PRODUCT_AMT
数値	-123, 5500.123
引用符で囲まれたリテラル文字列	"AUTO"、"Ca"
列テスト	@NULL、@PRESENT、@ABSENT(レコードの列が NULL、存在、不在かのテスト)。これらのテストは、Oracle GoldenGate に組み込まれています。
比較演算子	=, <>, >, <, >=, <=
結合演算子	AND、OR

表 37　許可されている WHERE 演算子 (続き)

演算子	例 :
グループ化用カッコ	複数の要素を論理的にグループ化するには、開きおよび閉じカッコを使用します。

例　次の WHERE の例では、AMOUNT 列が 10,000 を超えるとすべてのレコードを返し、AMOUNT が存在しないとレコードは破棄されません。

```
WHERE (amount = @PRESENT AND amount > 10000)
```

MAPEXCLUDE

適用対象　Replicat

MAPEXCLUDE パラメータと MAP パラメータでは、ワイルドカード指定から表を明示的に除外します。MAPEXCLUDE は、除外する表を含むすべての MAP 文よりも先に指定する必要があります。

デフォルト　なし

構文　`MAPEXCLUDE <exclude specification>`

引数	説明
`<exclude specification>`	除外する表の名前またはワイルドカード指定。MAPEXCLUDE には、MAP 文で表をワイルドカード指定するときと同じワイルドカード・ルールが適用されます。

例　次の例では、MAP 文は TEST という名前の表を除くすべての表を取得します。

```
MAPEXCLUDE fin.TEST
MAP fin.*, TARGET fin.*;
```

MARKERTABLE

適用対象　GLOBALS

MARKERTABLE パラメータでは、Oracle DDL 同期をサポートする DDL マーカー表の名前をデフォルトの GGS_DDL_HIST 以外にする場合に、その名前を指定します。マーカー表には、DDL 操作に関する情報が保持されます。このパラメータは、Oracle にのみ有効です。

マーカー表の名前は、params.sql スクリプトの marker_table_name パラメータにも指定する必要があります。このスクリプトは、ルート Oracle GoldenGate インストール・ディレクトリにあります。

マーカー表および params.sql の詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。

デフォルト　GGS_MARKER

構文 MARKERTABLE <table_name>

引数	説明
<table_name>	マーカー表名。

MAXDISCARDRECS

適用対象 Replicat

MAXDISCARDRECS パラメータでは、破棄ファイルにレポートされる MAP 文当たりのエラー数を制限します。

このパラメータは、次の理由で使用します。

- 多数のエラーの発生が予想されるが、レポートする必要がない。
- 破棄ファイルのサイズを管理する。

このパラメータは表に固有で、それ以降のすべての MAP 文に適用されます。1 つのパラメータ・ファイルで、MAXDISCARDRECS の複数のインスタンスを使用できます。

デフォルト なし

構文 MAXDISCARDRECS <number>

引数	説明
<number>	レポートするエラーの最大数。

例 MAXDISCARDRECS 1000

MAXFETCHSTATEMENTS

適用対象 Extract

MAXFETCHSTATEMENTS パラメータでは、Extract が Oracle ソース・データベースから行データをフェッチするために使用可能な、許可される準備済問合せの最大数を制御します。フェッチされたデータは、トランザクション・ログ・レコードから論理 SQL 文を構築するために使用できる情報が不足しているときに使用されます。

問合せは必要に応じて準備およびキャッシュされます。MAXFETCHSTATEMENTS で指定した値に到達すると、最も古い問合せが最も新しい問合せに置き換えられます。このパラメータの値は、Extract がフェッチ問合せのみのために保持しているオープンされているカーソル数を制御します。Extract は、ストアド・プロシージャなど他の目的のために追加のカーソルを使用できます。このパラメータは、Oracle データベースにのみ有効です。

デフォルト 100

構文 MAXFETCHSTATEMENTS <number>

引数	説明
<number>	Extract が準備済問合せに使用する最大カーソル数。データベースが、MAXFETCHSTATEMENTS で指定するカーソル数の他に、別のアプリケーションおよびプロセスが使用するカーソルもサポートできることを確認してください。

例 MAXFETCHSTATEMENTS 150

MAXGROUPS

適用対象 GLOBALS

MAXGROUPS パラメータでは、Oracle GoldenGate のインスタンスで許可されるプロセス・グループの最大数を指定します。Manager プロセスは、このパラメータを確認して、プロセスのリソース割当てを決定します。GGSCI プロセスは、このパラメータを確認して、作成が許可されるグループの最大数を制御します。

特定のシステムで実行できる実際のプロセス数は、使用可能なシステム・リソースによって異なります。システム・リソースを超えると、MAXGROUPS の設定にかかわらず、エラーが返されます。

デフォルト 300 グループの最大値

構文 MAXGROUPS <number_of_groups>

引数	説明
<number_of_groups>	Oracle GoldenGate のインスタンスで許可されるグループ数。300 から 5000 までの値を指定します。

例 MAXGROUPS 600

MAXSQLSTATEMENTS

適用対象 Replicat

MAXSQLSTATEMENTS パラメータでは、Replicat が標準処理モードおよび BATCHSQL モード (129 ページを参照してください) 両方で使用可能な準備済 SQL 文の数を制御します。MAXSQLSTATEMENTS の値によって、Replicat が保持するオープンされているカーソル数が決定されます。データベースが、ここで指定するカーソル数の他に、別のアプリケーションおよびプロセスが使用するカーソルもサポートできることを確認してください。MAXSQLSTATEMENTS を変更する前に、Oracle サポートに連絡してください。詳細は、http://support.oracle.com を参照してください。

MAXSQLSTATEMENTS を使用するには、DYNSQL パラメータが有効である必要があります。DYNSQL はデフォルトです。

デフォルト 250 カーソル

構文 MAXSQLSTATEMENTS <number>

引数	説明
<number>	Replicat が使用する最大カーソル数。最大値は 250 です。最小値は 1 です。

例 MAXSQLSTATEMENTS 200

MAXTRANSOPS

適用対象 Replicat

MAXTRANSOPS パラメータでは、大きなソース・トランザクションを小さく分割してターゲット・システムに送信します。このパラメータは、ターゲット・データベースが大きなトランザクションに対応するように構成されていないときに使用できます。たとえば、ターゲットの Oracle ロールバック・セグメントが百万の削除を実行するソース・トランザクションを再作成できるほど大きくない場合に MAXTRANSOPS 10000 を指定すると、Replicat は各 10,000 削除グループの後にコミットを発行できます。

使用の制限

MAXTRANSOPS は、効率化だけではなく正当なビジネス上の目的を持って使用する必要があります。MAXTRANSOPS を使用する場合、Replicat は正しい順番で操作を適用しますが、ソース・アプリケーションで定義された境界に課せられているトランザクションの整合性を変更することになります。

デフォルトでは、失敗からのリカバリの際、Extract は失敗の発生時に処理中だったソース・トランザクション全体を再送し、古いデータを上書きするのではなく、トレイル・ファイルの末尾にこのトランザクションを追加します。新しいトランザクションには再開レコードによってフラグが立てられ、ロールバックしてこのトランザクションをもう一度開始する必要があることが Replicat に通知されます。ただし、MAXTRANSOPS によって Replicat にこのトランザクションを分割させた場合、Replicat がロールバックできるのは、ターゲット・データベースにコミットされていないトランザクションのみです。Replicat がコミット済操作をもう一度処理した場合、SQL 操作のタイプに応じて、行重複エラーまたは行方不明エラーが発生します。

MAXTRANSOPS を使用しながらこのような状況を回避するには、RECOVERYOPTIONS パラメータを OVERWRITEMODE に設定して Extract を構成し、古いトランザクション・データを新しいデータで上書きします。ただし、上書きによってトレイル・レコードが破損することがあるため、このモードでは特定の失敗の後のリカバリがより困難になる場合があります。このようなケースでは、Oracle サービス・リクエストをオープンすることをお薦めします。詳細は、http://support.oracle.com を参照してください。

注意 Replicat の異常終了エラーをトラブルシューティングする際、GROUPTRANSOPS を 1 に、MAXTRANSOPS を 1 に設定するよう Oracle サポートから求められる場合があります。これは単にトラブルシューティングのための一時的な構成で、本番環境で常時使用しないでください。そうしないと、データ整合性のエラーになります。

デフォルト 100,000,000

構文 MAXTRANSOPS <transaction count>

引数	説明
<transaction count>	単一のトランザクション・グループに分割する操作数。

例 MAXTRANSOPS 10000

MGRSERVNAME

適用対象 GLOBALS

GLOBALS パラメータ・ファイルの MGRSERVNAME パラメータでは、Manager プロセスを Windows サービスとしてインストールする場合にその名前を指定します。このパラメータは、Oracle GoldenGate の複数のインスタンスをインストールするときや、Manager プロセスを使用する Oracle GoldenGate Veridata エージェントもインストールするときなど、同一のシステムにサービスとして複数の Manager のインスタンスをインストールする場合にのみ必要です。

インストールする Manager サービスごとに、MGRSERVNAME を含む GLOBALS ファイルが必要です。システムにサービス名を登録するときに、インストーラが MGRSERVNAME を参照するため、これらのファイルは*サービスのインストール前に*作成する必要があります。

デフォルト なし

構文 MGRSERVNAME <name>

引数	説明
<name>	Manager サービス名 (1 単語)。

例 MGRSERVNAME GoldenGate

NAMEMATCHIGNORECASE | NAMEMATCHNOWARNING | NAMEMATCHEXACT

適用対象 GLOBALS

これらのパラメータでは、フォールバック名マッピングの動作を制御します。Oracle、DB2、および SQL/MX など、ソース・データベースが大 / 小文字を区別し、ターゲット・データベースが大 / 小文字を区別するオブジェクト名と区別しないオブジェクト名の両方をサポートする場合、フォールバック名マッピングはデフォルトで有効になります。

フォールバック名マッチングは次のように行われます。ソース表名が大 / 小文字を区別する場合、Oracle GoldenGate はターゲット・データベースで大 / 小文字を区別するワイルドカード・マッピングを適用して、完全一致するものを検索します。ターゲット・データベースに、大 / 小文字の区別を含む完全に一致するターゲット表名がない場合、フォールバック名マッピングは大 / 小文字を区別しないターゲット表マッピングを実行して、名前の一致を検索します。この動作は、NAMEMATCHIGNORECASE パラメータで制御されます。

フォールバック名マッピングを許可しながら、警告メッセージを出力するには、NAMEMATCHNOWARNING パラメータを使用します。

フォールバック名マッチングを無効化するには、NAMEMATCHEXACT パラメータを使用します。NAMEMATCHEXACT を指定していて、大 / 小文字を区別して完全に一致するものが見つからない場合、Oracle GoldenGate はエラーを返して異常終了します。

デフォルト NAMEMATCHIGNORECASE

構文 NAMEMATCHIGNORECASE | NAMEMATCHNOWARNING | NAMEMATCHEXACT

NOHEADERS

適用対象 Extract

NOHEADERS パラメータでは、抽出ファイルにレコード・ヘッダーが含まれていないことを示します。この場合、Replicat は、入力ファイルには単一の表に関連する挿入レコードのみが含まれ、各レコードの長さとタイプが同一であるとみなします。

NOHEADERS を使用する場合は、Extract パラメータ・ファイルで FORMATASCII パラメータと NOHDRFIELDS オプションを使用する必要があります。NOHEADERS を使用する場合、すべての MAP 文で指定できるソース表は 1 つのみになります。

デフォルト なし

構文 NOHEADERS

NUMFILES

適用対象 Extract および Replicat

NUMFILES パラメータでは、TABLE または MAP 文で指定されている表の情報を保持するために割り当てる初期メモリー構造数を制御します。NUMFILES は、すべての TABLE または MAP エントリが有効になる前に、SOURCEDEFS または TARGETDEFS パラメータより前で指定する必要があります。

NUMFILES の値に到達後に動的に割り当てる追加のメモリー構造数を制御するには、ALLOCFILES パラメータを使用します (123 ページを参照してください)。システム・リソースが許可するメモリーが、プロセスによって必要に応じて割り当てられるので、NUMFILES と ALLOCFILES は、ともにデフォルト値で問題なく機能するはずです。

デフォルト 1000

構文 NUMFILES <number of structures>

引数	説明
<number of structures>	割り当てるメモリー構造数。メモリーの無駄な消費を防ぐため、NUMFILES を不必要に高い数に設定しないでください。Oracle GoldenGate のメモリーは、最大で 200 万表をサポートします。

例 NUMFILES 4000

OBEY

適用対象 Extract および Replicat

OBEY パラメータでは、現在のパラメータ・ファイル以外のファイルからパラメータ設定を取得します。

OBEY を使用するには、次の手順に従います。

1. 取得するパラメータを含むパラメータ・ファイルを作成して保存します。異なる頻度で使用されるパラメータ設定を含むテキスト・ファイルのライブラリを作成できます。
2. アクティブなパラメータ・ファイルで OBEY を使用すると、他のファイルが呼び出されます。OBEY 文は、他の OBEY 文内にネストできません。アクティブなパラメータ・ファイルで OBEY パラメータを検出すると、Oracle GoldenGate は参照ファイルからパラメータを処理し、アクティブなファイルに戻って残りのパラメータを処理します。

頻繁に使用するパラメータを呼び出すために、OBEY のかわりに、または OBEY の使用に加えて、Oracle GoldenGate マクロを使用できます。マクロ使用の詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。

デフォルト なし

構文 `OBEY <file name>`

引数	説明
`<file name>`	パラメータまたはコマンドの取得元ファイルの相対名または完全修飾名。

```
OBEY /home/ggs/myparams
```

OUTPUTFILEUMASK

適用対象 GLOBALS

OUTPUTFILEUMASK パラメータでは、Oracle GoldenGate プロセスがトレイル・ファイルおよび破棄ファイルの作成に使用する 8 進数 umask を指定します。OUTPUTFILEUMASK は、WIN32 システムには有効ではありません。

デフォルト umask 0(すべての権限)

構文 `OUTPUTFILEUMASK <umask>`

引数	説明
`<umask>`	umask 値。0 ～ 0777 に設定する必要があります (それ以外の値にすると、エラー `"Missing or invalid option for OUTPUTFILEUMASK."` が発生します)。

例 `OUTPUTFILEUMASK 066`

OVERRIDEDUPS | NOOVERRIDEDUPS

適用対象 Replicat

OVERRIDEDUPS および NOOVERRIDEDUPS パラメータでは、両方のレコードが同一のキーを持つときに、Replicat がターゲット・データベースの既存のレコードをレプリケートされたレコードで上書きするかどうかを制御します。

- OVERRIDEDUPS では、既存のレコードを上書きします。事前にターゲット表の切捨てを行わない初期ロードや、信頼できるソースとターゲット表の再同期のときに使用できます。
- NOOVERRIDEDUPS (デフォルト) では、既存のレコードを上書きせず、かわりに重複レコード・エラーを生成します。例外 MAP 文を使用して SQLEXEC 句で SQL プロシージャを実行し、エラーに対するレスポンスを開始できます。そうしない場合、トランザクションは異常終了することがあります。例外マップの詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。

例外マップが使用できない場合に Replicat を異常終了させずに重複レコードをバイパスするには、次のような REPERROR パラメータ文を指定します (<duplicate key error> は、主キー制約エラーに対するデータベース・エラー番号です)。

```
REPERROR (<duplicate key error>, IGNORE)
```

たとえば、Oracle データベースの場合、この文は次のようになります。

```
REPERROR (1, IGNORE)
```

重複レコードは、破棄ファイルに出力されます。

OVERRIDEDUPS および NOOVERRIDEDUPS は、TABLE または MAP 文に固有なので、各表または表グループに異なるルールを作成できます。SQLDUPERR パラメータ (333 ページを参照してください) を OVERRIDEUPS とともに使用して、データベースによって返される重複挿入の数値エラー・コードを指定します。

OVERRIDEDUPS は、HANDLECOLLISIONS が指定されたときに自動的に有効化されます。

OVERRIDEDUPS が有効な場合は、すべての Replicat プロセスにわたってレコードが時系列順に処理されない場合があります。

デフォルト NOOVERRIDEDUPS

構文 OVERRIDEDUPS | NOOVERRIDEDUPS

PASSTHRU | NOPASSTHRU

適用対象 Extract

PASSTHRU および NOPASSTHRU パラメータでは、データ・ポンプ Extract がパススルー・モードで表を処理するか、標準モードで処理するかを制御します。パススルー・モードでは、Extract プロセスはデータベースからもデータ定義ファイルからも表定義を参照しません。通常は、Extract プロセスはデータベースにログインしてデータ定義を取得し、ターゲットが NonStop の場合には、データ定義ファイルを読み取ります。データ定義は、マッピングや変換ファンクションの実行に使用されます。

パススルー・モードを使用すると、取得したデータを、データベースがインストールされていない仲介システム上のデータ・ポンプにカスケードできます。フィルタリング、列マッピング、SQLEXEC ファンクション、変換、またはデータの操作や変換を必要とするその他のファンクションが使用できないため、ソースおよびターゲットの表名と構造は同一である必要があります。WILDCARDRESOLVE パラメータは、DYNAMIC (デフォルト) に設定する必要があります。

PASSTHRU および NOPASSTHRU パラメータは、表に固有です。一方のパラメータは、もう一方のパラメータが見つかるまで、それ以降のすべての MAP 文およびトレイルに有効です。したがって、特定の表セットにはパススルー動作を指定し、他の表に対してはデータ操作を含む標準処理を継続できます。操作が必要な表では、フィルタリングを実行する場合にソース定義ファイルが必要になり、列マッピングまたは変換を実行する場合にターゲット定義ファイルが必要になります。これらのファイルは、Oracle GoldenGate がこうしたアクションを実行するために必要なメタデータを提供します。

PASSTHRU モードでは、データ・ポンプは ASCII から EBCDIC、または EBCDIC から ASCII への自動変換を実行しません。

DDL レプリケーションでの PASSTHRU

PASSTHRU モードでは、DDL はデータ・ポンプまたは VAM ソート Extract を通じて自動的に伝播されます。そのため、特定の名前のソース表で実行される DDL (ALTER TABLE TableA... など) は、データ・ポンプまたは VAM ソート Extract によって同一の名前の表に適用されます (ALTER TABLE TableA)。他の表を指定する TABLE 文の有無にかかわらず、このプロセスでは ALTER TABLE TableB としてマップすることはできません。

デフォルト NOPASSTHRU

構文 PASSTHRU | NOPASSTHRU

例 次に、fin.acct からのすべてのデータをパススルーし、fin.sales には通常の処理を許可するパラメータ・ファイルの例を示します。

```
EXTRACT fin
USERID ogg, PASSWORD AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC, &
    AES128, ENCRYPTKEY securekey
RMTHOST sysb, MGRPORT 7809, ENCRYPT AES192 KEYNAME mykey
ENCRYPTTRAIL AES192 KEYNAME mykey2
RMTTRAIL /ggs/dirdat/rt
PASSTHRU
TABLE fin.acct;
NOPASSTHRU
TABLE fin.sales, WHERE (ACCOUNT-CODE < 100);
```

PASSTHRUMESSAGES | NOPASSTHRUMESSAGES

PASSTHRUMESSAGES および NOPASSTHRUMESSAGES パラメータでは、パススルー・モードで処理される表に関するメッセージを Extract レポート・ファイルに書き込むかどうかを制御します。有効にすると、次のようなメッセージが書き込まれます。

```
"PASSTHRU mapping resolved for source table <table name>"
```

デフォルト PASSTHRUMESSAGES

構文 PASSTHRUMESSAGES | NOPASSTHRUMESSAGES

PORT

適用対象 Manager

PORT パラメータでは、動的サービスをリクエストするリモート・プロセス (通常は初期ロードの Replicat または Collector プロセス) とやり取りする Manager プロセス用の TCP/IP ポート番号を指定します。可能な場合にはデフォルト・ポート番号を使用してください。

デフォルト ポート 7809

構文 PORT <number>

引数	説明
<number>	使用可能なポート番号。

例 PORT 7809

PURGEDDLHISTORY

適用対象 Manager

PURGEDDLHISTORY パラメータでは、行をパージすることによって Oracle データベースの DDL 履歴表のサイズを制御します。履歴表をパージするときは注意が必要です。これは DDL 同期プロセスの整合性にとって非常に重要な表であり、パージは Oracle GoldenGate によってリカバリ不能であるため、早期にパージを実行しないようにしてください。DDL データの損失の可能性をなくすために、定期的に履歴表のバックアップを取るようにしてください。

行を保持する最小時間および最大時間は、最終更新日に基づいて指定できます。最大時間および最小時間両方のルールを指定する必要があります。両方の指定がなければ、Manager が行をいつ削除するかの完全な基準がありません。たとえば、MINKEEPHOURS 3 を MAXKEEPHOURS 5 とともに使用した場合、過去 3 時間変更されなかった行を保持し、少なくとも 5 時間変更されなかったときに削除するよう指定できます。

このパラメータには、表名を指定する必要はありません。Oracle GoldenGate は、まず GLOBALS ファイルの DDLTABLE <table> パラメータに指定されている名前を探し、このパラメータが存在しない場合、Oracle GoldenGate はデフォルト名の GGS_DDL_HIST を使用します。

> **注意** DDL 履歴表のパージの詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。

PURGEDDLHISTORY を使用するには、USERID パラメータ、および必要なときは SOURCEDB パラメータも使用してログイン情報を指定する必要があります。

このパラメータは、Oracle にのみ有効です。

デフォルト 1 時間ごとにパージ

構文
```
PURGEDDLHISTORY
{, <max rule>}
[, <min rule>]
[, <frequency>]
```

引数	説明
<max rule>	必須。次のいずれかを使用して行を保持する最大時間を設定します。 MAXKEEPHOURS <n> 行が <n> 時間変更されなかった場合にパージします。

引数	説明
	MAXKEEPDAYS <n> 行が <n> 日間変更されなかった場合にパージします。
<min rule>	オプションですが、指定することをお薦めします。次のいずれかを使用して行を保持する最小時間を設定します。 MINKEEPHOURS <n> 少なくとも指定した時間変更されていない行を保持します。 MINKEEPDAYS <n> 少なくとも指定した日数にわたって変更されていない行を保持します。
<frequency>	DDL 履歴をパージする間隔を設定します。Manager でメンテナンス・タスクを処理するデフォルトの時間は、CHECKMINUTES パラメータで指定されているとおり 10 分です (148 ページを参照してください)。Manager は、10 分ごとに PURGEOLDEXTRACTS の間隔を確認し、指定された間隔の経過後にパージを実行します。<frequency> は次のいずれかです。 FREQUENCYMINUTES <n> DDL 履歴をパージする間隔 (分) を設定します。デフォルトのパージ間隔は 60 分です。 FREQUENCYHOURS <n> DDL 履歴をパージする間隔 (時間) を設定します。

例 次の例では、過去 3 日間変更されなかったすべての行を保持し、少なくとも 5 日間変更されなかったときに削除します。パージ間隔は 30 分です。

```
PURGEDDLHISTORY MINKEEPDAYS 3, MAXKEEPDAYS 5, FREQUENCYMINUTES 30
```

PURGEDDLHISTORYALT

適用対象 Manager

PURGEDDLHISTORYALT パラメータを使用して、指定された表のオブジェクト ID に関連付けられているパーティション化されたオブジェクト ID を追跡する Oracle データベース内の内部 DDL 履歴表のサイズを制御します。このパラメータでは、必要なくなった行をパージします。

この表をパージするときは注意が必要です。パージは Oracle GoldenGate ではリカバリできません。DDL データの損失の可能性をなくすために、定期的に履歴表のバックアップを取るようにしてください。

このパラメータには、入力として表名は必要ありません。デフォルトの名前は、GGS_DDL_HIST_ALT または GLOBALS ファイルで DDLTABLE <table> パラメータが使用されている場合は、このパラメータに指定されたカスタム名です。

構文オプションと使用法については、290 ページの「PURGEDDLHISTORY」を参照してください。

PURGEMARKERHISTORY

適用対象 Manager

PURGEMARKERHISTORY では、行をパージすることによって Oracle GoldenGate マーカー表のサイズを制御します。マーカー表はいつでもパージできます。

このパラメータには、表名を指定する必要はありません。Oracle GoldenGate は、まず GLOBALS ファイルの MARKERTABLE <table> パラメータに指定されている名前を探し、このパラメータが存在しない場合、Oracle GoldenGate はデフォルト名の GGS_MARKER を使用します。

行を保持する最小時間および最大時間は、最終更新日に基づいて指定できます。最大時間および最小時間両方のルールを指定する必要があります。両方の指定がなければ、Manager がいつ行を削除するかの完全な基準がありません。たとえば、MINKEEPHOURS 3 を MAXKEEPHOURS 5 とともに使用した場合、過去 3 時間変更されなかった行を保持し、少なくとも 5 時間変更されなかったときに削除するよう指定できます。

> **注意** **マーカー表のパージの詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。**

PURGEMARKERHISTORY を使用するには、USERID パラメータ、および必要なときは SOURCEDB パラメータも使用してログイン情報を指定する必要があります。

デフォルト 1 時間ごとにパージ

構文

```
PURGEMARKERHISTORY
{, <max rule>}
[, <min rule>]
[, <frequency>]
```

引数	説明
<max rule>	必須。次のいずれかを使用して行を保持する最大時間を設定します。 MAXKEEPHOURS <n> 行が <n> 時間変更されなかった場合にパージします。 MAXKEEPDAYS <n> 行が <n> 日間変更されなかった場合にパージします。
<min rule>	オプションですが、指定することをお薦めします。次のいずれかを使用して行を保持する最小時間を設定します。 MINKEEPHOURS <n> 少なくとも指定した時間変更されていない行を保持します。 MINKEEPDAYS <n> 少なくとも指定した日数にわたって変更されていない行を保持します。

引数	説明
<frequency>	マーカー履歴をパージする間隔を設定します。Manager でメンテナンス・タスクを処理するデフォルトの時間は、CHECKMINUTES パラメータで指定されているとおり 10 分です (148 ページを参照してください)。Manager は、10 分ごとに PURGEOLDEXTRACTS の間隔を確認し、指定された間隔の経過後にパージを実行します。<frequency> は次のいずれかです。 FREQUENCYMINUTES <n> マーカー履歴をパージする間隔 (分) を設定します。デフォルトのパージ間隔は 60 分です。 FREQUENCYHOURS <n> マーカー履歴をパージする間隔 (時間) を設定します。

例 次の例では、過去 3 日間変更されなかったすべての行を保持し、少なくとも 5 日間変更されなかったときに削除します。パージ間隔は 30 分です。

```
PURGEMARKERHISTORY MINKEEPDAYS 3, MAXKEEPDAYS 5, FREQUENCYMINUTES 30
```

PURGEOLDEXTRACTS

適用対象 Manager、Extract、Replicat

このパラメータの実装は、プロセスによって異なります。

Extract および Replicat 用 PURGEOLDEXTRACTS

Extract または Replicat パラメータ・ファイルの PURGEOLDEXTRACTS パラメータでは、Oracle GoldenGate が新しいトレイル・ファイルから処理を開始するたびに、古いトレイル・ファイルを削除します。トレイル・ファイルを蓄積しないのでディスク領域を節約できます。パージは、プロセスがファイルの処理を終了したことがチェックポイントによって示された後に実行されます。

Extract によるパージの実行は、プロセスがデータ・ポンプの場合に適しています。データがターゲット・システムに送信された後にファイルをパージできます。それ以外の場合は、通常は Replicat によってパージを実行します。

Extract または Replicat パラメータ・ファイルでは、プロセスのインスタンスが 1 つの場合にのみ PURGEOLDEXTRACTS を使用するようにしてください。複数のグループが同一のトレイル・ファイル・セットを読み込む場合には、1 つのプロセスが読み込みを完了する前に、別のプロセスによってファイルがパージされてしまう可能性があります。かわりに、一元的にトレイル・ファイルを管理することができ、すべての Oracle GoldenGate 構成での使用が推奨される、Manager 用の PURGEOLDEXTRACTS を使用するようにしてください。

デフォルト 順序の次のファイルに切り替えるときにトレイル・ファイルをパージする。

構文

```
PURGEOLDEXTRACTS
```

Manager 用 PURGEOLDEXTRACTS

Manager パラメータ・ファイルの PURGEOLDEXTRACTS パラメータでは、Oracle GoldenGate がトレイル・ファイルの処理を完了したときにトレイル・ファイルをパージします。PURGEOLDEXTRACTS を使用しない場合、パージは実行されないため、トレイル・ファイルが大量のディスク領域を消費する可能性があります。

Extract または Replicat 用の PURGEOLDEXTRACTS を使用するよりも PURGEOLDEXTRACTS を Manager パラメータとして使用することをお薦めします。Manager パラメータとして PURGEOLDEXTRACTS を使用すると、一元的にトレイル・ファイルを管理し、複数のプロセスを考慮できます。

このパラメータの使用方法

パージを制御するには、次のルールに従います。

- USECHECKPOINTS では、すべてのプロセスがファイルの処理を完了したことがチェックポイントによって示されたときにパージを行います。これはデフォルトですが、NOUSECHECKPOINTS オプションを使用すると無効化できます。チェックポイントを基準にパージする場合、すべてのプロセスがデータの処理を終了するまでデータは削除されません。明示的な NOUSECHECKPOINTS エントリがある場合を除き、USECHECKPOINTS は PURGEOLDEXTRACTS とともに明示的に定義されているかにかかわらず使用されます。本番環境では、データの整合性を確保するために、チェックポイントに基づいてパージを実行することが不可欠です。USECHECKPOINTS では、パージを実行する前に、Extract および Replicat 両方のチェックポイントが考慮されます。
- MINKEEP ルールでは、変更されていないデータを保持する最小時間を設定します。
  - MINKEEPHOURS または MINKEEPDAYS では、<n> 時間または日間データを保持します。
  - MINKEEPFILES では、少なくとも <n> 個のトレイル・ファイル (アクティブなファイルを含む) を保持します。デフォルトは 1 です。

  MINKEEP オプションは *1* つのみ使用します。このオプションが複数使用されている場合、Oracle GoldenGate は次に基づいてそのうち 1 つを選択します。

  - MINKEEPHOURS および MINKEEPDAYS 両方が指定されている場合、最後のオプションを受け付け、もう一方を無視します。
  - MINKEEPHOURS または MINKEEPDAYS が MINKEEPFILES とともに使用されている場合、MINKEEPHOURS または MINKEEPDAYS を受け付け、MINKEEPFILES を無視します。

Manager は、CHECKMINUTES パラメータで設定されている値に基づいてパージを実行します (148 ページを参照してください)。この値に到達すると、次のようにパージ・ルールが評価されます。

1. USECHECKPOINTS のみ。MINKEEP ルールが指定されておらず、USECHECKPOINTS が有効な場合、保持するファイルの最小数は 1 です。チェックポイントによって処理されたことが示されているファイルは、ファイルの最小数 1 を下回らないかぎりパージされます。
2. USECHECKPOINTS と MINKEEP ルール。USECHECKPOINTS が有効化され、チェックポイントによってファイルが処理されたことが示されている場合、適切な MINKEEP ルールを違反しないかぎり、このファイルはパージされます。
3. NOUSECHECKPOINTS のみ。MINKEEP ルールおよび NOUSECHECKPOINTS が指定されていない場合、チェックポイントは考慮されず、デフォルト・ルールの 1 つのファイルの保持ルールを違反しないかぎり、ファイルはパージされます。
4. NOUSECHECKPOINTS と MINKEEP ルール。MINKEEP ルールおよび NOUSECHECKPOINTS が指定されている場合、MINKEEP ルールを違反しないかぎり、ファイルはパージされます。

Manager は、ローカル・システムに構成されている Extract および Replicat プロセスに基づいてパージするファイルを決定します。少なくとも 1 つのプロセスがトレイル・ファイルを読み取っている場

合、Manager は指定されているルールを適用しますが、それ以外の場合はルールは有効になりません。

Manager 用 PURGEOLDEXTRACTS の詳細なガイドライン

- 同一の Manager パラメータ・ファイル内では、500 を超える PURGEOLDEXTRACTS パラメータ文を使用しないでください。
- このパラメータを使用する場合は、ユーザーおよび Oracle GoldenGate 以外のプログラムにトレイル・ファイルの削除を許可しないでください。それにより、PURGEOLDEXTRACTS が正常に機能しなくなります。
- トレイルが NFS に格納されている場合、NFS ドライブと Manager が実行中のローカル・システムの間に、システム時間差があります。トレイルは NFS 時間で作成されますが、トレイル内のレコードのタイムスタンプはローカル・システム時間で比較され、パージするかどうかを決定します。MINKEEP ルールを作成すると、すべての時間差を考慮します。

デフォルト USECHECKPOINTS

構文

```
PURGEOLDEXTRACTS <trail name>
[, USECHECKPOINTS | NOUSECHECKPOINTS]
[, <min rule>]
[, <frequency>]
```

引数	説明
<trail name>	パージするトレイル名。相対名または完全修飾名を使用します。
USECHECKPOINTS	指定されている MINKEEP ルールに従い、すべての Extract および Replicat プロセスが処理を終了したことがチェックポイントによって示された後、データのパージを許可します。
NOUSECHECKPOINTS	次のいずれかの最小値を維持しながら、チェックポイントを考慮せずにパージを許可します。 ◆ MINKEEP ルールが使用されていない場合は 1 ファイル または ◆ MINKEEP ルールで指定されたファイル数
<MINKEEP rule>	データの最小保持時間を設定する次のいずれかのルールを使用できます。 MINKEEPHOURS <n> 少なくとも指定した時間変更されていないファイルを保持します。 MINKEEPDAYS <n> 少なくとも指定した日数にわたって変更されていないファイルを保持します。 MINKEEPFILES <n> 少なくとも <n> 個の変更されていないトレイル・ファイル (アクティブなファイルを含む) を保持します。
<frequency>	古いトレイル・ファイルをパージする間隔を設定します。Manager でメンテナンス・タスクを処理するデフォルトの時間は、CHECKMINUTES パラメータで指定されているとおり 10 分です (148 ページを参照してください)。Manager は、10 分ごとに PURGEOLDEXTRACTS の間隔を確認し、指定された間隔の経過後にパージを実行します。<frequency> は次のいずれかです。

引数	説明
	FREQUENCYMINUTES <n> 古いトレイル・ファイルをパージする間隔 (分) を設定します。デフォルトのパージ間隔は 60 分です。 FREQUENCYHOURS <n> 古いトレイル・ファイルをパージする間隔 (時間) を設定します。

例 1 状況 : トレイル・ファイルとして AA000000、AA000001、および AA000002 が存在します。Replicat は 4 時間にわたって停止しており、いずれのファイルの処理も完了していません。Manager パラメータには、次が含まれています。

```
PURGEOLDEXTRACTS /ggs/dirdat/AA*, USECHECKPOINTS, MINKEEPHOURS 2
```

結果 : 変更されていないファイルの保持時間が経過しました。しかし、チェックポイントによって、Replicat がこれらのファイルの処理を完了していないことが示されているため、どのファイルもパージされません。

例 2 状況 : トレイル・ファイルとして AA000000、AA000001、および AA000002 が存在します。Replicat は 4 時間にわたって停止しており、いずれのファイルの処理も完了していません。Manager パラメータには、次が含まれています。

```
PURGEOLDEXTRACTS /ggs/dirdat/AA*, NOUSECHECKPOINTS, MINKEEPHOURS 2
```

結果 : ファイルの最小保持時間のルールが満たされているため、すべてのトレイル・ファイルがパージされます。

例 3 状況 : Replicat および Extract はデータの処理を完了しています。過去 5 時間にわたってトレイル・ファイルへのアクセスは発生していません。トレイル・ファイルとして AA000000、AA000001、および AA000002 が存在します。Manager パラメータには、次が含まれています。

```
PURGEOLDEXTRACTS /ggs/dirdat/AA*, USECHECKPOINTS, MINKEEPHOURS 4, &
MINKEEPFILES 4
```

結果 : これは、MINKEEP オプションの使用を 1 つのみにする理由を示す例です。USECHECKPOINTS の要件は満たされているため、AA000002 をパージするかどうかを決定するときに最少数のルールが考慮されます。AA000002 がパージされると、残りのファイルが 2 つになり、MINKEEPFILES ルール違反になります。しかし、MINKEEPFILES および MINKEEPHOURS 両方が指定されているため、MINKEEPFILES は無視されます。5 時間変更されておらず、かつ MINKEEPHOURS の 4 時間の条件も満たすため、このファイルはパージされます。

PURGEOLDTASKS

適用対象 Manager

PURGEOLDTASKS パラメータでは、指定した時間の経過後または Extract および Replicat が正常に停止した後に、Extract および Replicat タスクをパージします。次のルールに従って、いつタスクを削除するかを指定できます。

- 特定の時間または日数前に最後に開始されたタスク。開始されたことがないタスクの場合は、このルールの適用基準として作成日時が使用されます。
- 正常に停止したか、開始されたことがないタスク。このルールは、最後に開始された時刻よりも優先されます。このルールを使用して、異常終了したタスクのパージを防止します。

同一の Manager パラメータ・ファイルで使用できる PURGEOLDTASKS パラメータ文は、300 未満です。

デフォルト なし

構文

```
PURGEOLDTASKS <process> <group name>
[, <purge option>]
[USESTOPSTATUS]
```

引数	説明
<process>	有効な値 : ◆ EXTRACT ◆ REPLICAT ◆ ER(両方のプロセス)
<group name>	グループ名、または複数のグループを指定するワイルドカード。
<purge option>	指定した数の時間または日数にわたって更新されていない場合にタスクをパージします。 有効な値 : ◆ AFTER <number> DAYS ◆ AFTER <number> HOURS
USESTOPSTATUS	正常に停止したか、開始されたことがない場合にタスクをパージします。

例 次の例では、少なくとも 3 日間更新されていないすべての Extract タスクを削除し、正常に停止したか少なくとも 2 時間更新されていない場合に test_rep Replicat タスクを削除します。

```
PURGEOLDTASKS EXTRACT *, AFTER 3 DAYS
PURGEOLDTASKS REP test_rep, AFTER 2 HOURS, USESTOPSTATUS
```

RECOVERYOPTIONS

適用対象 Extract

RECOVERYOPTIONS パラメータでは、Extract が再起動時に既存のトレイル・ファイルのコンテンツを上書きするか、再起動後にファイルの既存のデータに新しいレコードを追加するかを制御します。

追加モード

Extract がデフォルトで動作する*追加モード*では、プロセスが失敗した場合にリカバリ・マーカーがトレイルに書き込まれ、Extract がファイルにリカバリ・データを追加するので、以前のすべてのデータの履歴がリカバリのために保持されます。

追加モードでは、起動時の Extract 初期化の際に、トレイルに書き込まれた最新の完了レコードのアイデンティティを識別します。Extract はこの情報を使用して、データ・ソースでこのトランザクションのコミット・レコードを検出したときにリカバリを終了し、抽出対象の次のコミット済トランザクションから新しいデータ取得を開始し、トレイルへの新しいデータの追加を開始します。データ・ポンプまたは Replicat は、このリカバリ・ポイントから読み取りを再開します。

上書きモード

*上書きモード*は、Oracle GoldenGate リリース 10.0 以前のリリースで使用されていたもう 1 つの Extract リカバリ方法です。これらのリリースでは、Extract は新しいデータを追加するのではなく、最新の書込みチェックポイント位置以降のトレイルの既存のトランザクション・データを上書きします。最初に書き込まれるトランザクションは、データ・ソースの最新の読取りチェックポイント以降の最初の抽出対象トランザクションです。

上書きモードでは、上書き前と同じレコード・イメージが上書き後に精密に正確な順序で配置されない可能性があります。この変化は、大きなオブジェクトのフェッチの実行が必要な場合や、Extract の構成パラメータが変更されたときに発生する可能性があります。上書きアクティビティが開始されてからトレイル・ファイルの最後に到達するまでの間に、Extract 処理に変更がある場合、ファイルの上書き部分の先端にずれが生じ、結果として再び書き込まれた最後のレコードの末尾が、以前の Extract インスタンスに書き込まれたレコードの最初の部分と共有する境界に配置されなくなります。1 つのトレイル・レコードから次のトレイル・レコードにかけての読取りを試行する Replicat またはデータ・ポンプは、破損したレコードの途中に到達すると、エラーとともに異常終了します。

推奨事項

Oracle Support のアナリストに指示されないかぎり、RECOVERYOPTIONS をデフォルトから変更しないでください。Extract がトレイルの既存データの上書きを許可されている場合、Oracle GoldenGate にとって失敗後のリカバリは困難になり、ターゲットに送信する必要があるデータが失われる可能性があります。

一部のケースでは、ターゲットで使用されている Oracle GoldenGate のリリースが Oracle GoldenGate リリース 10 よりも古い場合、Extract は下位互換性をサポートするために自動的に上書きモードに戻ります。古いバージョンでは、追加モードはサポートされていません。

パラメータの依存関係

EXTFILE、EXTTRAIL、RMTFILE および RMTTRAIL の RECOVERYOPTIONS パラメータと FORMAT オプションの間には、依存関係があります。RECOVERYOPTIONS を APPENDMODE に設定する場合、FORMAT オプションは RELEASE 10.0 以上に設定する必要があります。RECOVERYOPTIONS を OVERWRITEMODE に設定する場合、FORMAT オプションは RELEASE 9.5 以下に設定する必要があります。

デフォルト APPENDMODE

構文 `RECOVERYOPTIONS {APPENDMODE | OVERWRITEMODE}`

引数	説明
`APPENDMODE`	新しいデータをトレイル・ファイルの既存のデータに追加します。これはデフォルトです。
`OVERWRITEMODE`	最新のチェックポイントの位置から、古いデータを新しいデータで上書きします。

例 `RECOVERYOPTIONS OVERWRITEMODE`

REPERROR

適用対象 Replicat

REPERROR パラメータでは、Replicat のエラーへの対応方法を制御します。Replicat のエラーへのデフォルトのレスポンスは、異常終了です。

1 つの REPERROR 文を使用して大半のエラーをデフォルトの方法で処理しながら、1 つまたは複数の別の REPERROR 文を使用して、特定のエラーを別の方法で処理できます。たとえば、重複レコード・エラーは無視し、他のすべてのケースでは処理を異常終了させることができます。

次に示す構文では、<error> および <response> をカッコ内に指定する必要があることに注意してください。次に例を示します。

```
REPERROR (DEFAULT, ABEND)
REPERROR (-1, IGNORE)
```

ただし、RESET オプションはカッコ内に含めることができません。

```
REPERROR RESET
```

レコード・レベルのエラー処理のオプション

TRANSDISCARD および TRANSEXCEPTION を除くすべての REPERROR オプションは、個別のレコードの個別の SQL 操作に対するレスポンスに、エラー処理アクションを適用します。また、規定どおりに、MAP 文およびパラメータ・ファイルのその他のパラメータで構成されているのと同じトランザクション内のエラーのないレコードを処理します。

トランザクション・レベルのエラー処理のオプション

TRANSDISCARD、TRANSEXCEPTION および ABEND オプションは、トランザクション全体にエラー処理アクションを適用します。トランザクション内の個別のレコードまたはコミット操作で、トリガー・エラーが発生することがあります。(コミット・エラーには、関連付けられている特定のレコードがありません。) これらのオプションは、次の目的で使用できます。

- エラーが関連付けられている場合に、ソース・トランザクション全体がターゲットにレプリケートされるのを回避する。
- 延期された制約チェックがターゲット上で有効になっている場合、コミット・エラーに対応する。

TRANSDISCARD と TRANSEXCEPTION は、相互に排他的です。

トランザクション・レベルのオプションでの他のパラメータの影響

TRANSDISCARD および TRANSEXCEPTION では、ソース・トランザクションの境界が適用されます。ただし、トランザクションの境界を変更する他のパラメータがパラメータ・ファイルに存在する場合、それがエラー処理ロジックや結果に影響を与える可能性があります。

BATCHSQL および GROUPTRANSOPS

BATCHSQL または GROUPTRANSOPS (デフォルト) は、パフォーマンス向上のため、トランザクションの順序を維持しながら、異なるトランザクションの SQL 操作を、より大きいトランザクションにグループ化します。これらのパラメータが有効なときにエラーが発生すると、Replicat は、まず別の処理モードを入力してエラーの解決を試行します (各パラメータのドキュメントを参照してください)。エラーが解決しない場合、TRANSDISCARD または TRANSEXCEPTION が有効になり、Replicat は次のようにソース処

理モードに戻ります。

1. グループ化または配列化されたトランザクションをロールバックします。
2. ソース・トランザクションと同じトランザクション境界を使用して、問題のあるトランザクションの SQL 操作を、一度に 1 つリプレイします。
3. 破棄ロジック (TRANSDISCARD) または例外マッピング (TRANSEXCEPTION) を実行します。(詳細は、各オプションの説明を参照してください。)
4. TRANSDISCARD エラー処理が完了したら、BATCHSQL または GROUPTRANSOPS モードを再開します。

MAXTRANSOPS

TRANSDISCARD および TRANSEXCEPTION のトランザクション・レベルのエラー処理の整合性は、MAXTRANSOPS パラメータ設定の悪影響を受けることがあります。MAXTRANSOPS によって、Replicat は、ターゲットでのトランザクションの適用時に、非常に大きなレプリケートされたソース・トランザクションを小さいトランザクションに分割します。

TRANSDISCARD および TRANSEXCEPTION ロジックによって、Replicat は、最後に成功したコミット後の最初のレコードまでロールバックします。これは、問題のあるトランザクションの、実際の最初である場合とそうでない場合があります。そのトランザクションが分割されているか、およびトランザクションの一部が以前にコミットしたトランザクションにあるかどうかによって異なります。その場合、TRANSDISCARD または TRANSEXCEPTION アクションはソース上に発行されましたが、ターゲットからロールバックされた部分にのみ発行されたため、Replicat はこれらのアクションをトランザクション全体には適用できません。

MAXTRANSOPS を使用する場合、TRANSDISCARD および TRANSEXCEPTION が処理する可能性がある最大のトランザクションより大きい値に設定されていることを確認してください。これにより、トランザクションがターゲット上で小さいトランザクションに分割されていないことを確認できます。

統計でのトランザクション・レベルのオプションの影響

STATS REPLICAT など、GGSCI の情報コマンドの出力では、TRANSDISCARD または TRANSEXCEPTION ロジックで処理されたトランザクションの合計レコード数が表示されます。この数は、次のことを表します。

- Replicat は、FILTER または WHERE 句によって Oracle GoldenGate 処理から除外されたすべてのレコードを含む、トランザクションのすべてのレコードを破棄ファイルに書き込みます。
- トランザクション内のソース表に複数のターゲットがある場合、破棄トランザクションには、各レコードの複数のコピーが各ターゲットに 1 つ含まれます。
- トランザクションを破棄する場合、Replicat は、例外マッピング文 (EXCEPTIONSONLY または MAPEXCEPTION で指定) を無視します。

破棄処理 (TRANSDISCARD) または例外マッピング (TRANSEXCEPTION) が原因のエラーが発生すると、Replicat は異常終了します。

デフォルト TRANSABORT (デッドロックの場合)、ABEND (その他すべての場合)

構文

```
REPERROR { (
{DEFAULT | DEFAULT2 | <SQL error> | <user-defined error>},
{ABEND | DISCARD | EXCEPTION | IGNORE |
RETRYOP [MAXRETRIES <n>] |
TRANSABORT [, MAXRETRIES] [, DELAYSECS <n> | DELAYCSECS <n>] |
TRANSDISCARD |
TRANSEXCEPTION
}) |
RESET }
```

表 38　エラー指定

引数	説明
DEFAULT	明示的な REPERROR 文が指定されているエラーを除くすべてのエラーに対するグローバルなレスポンスを設定します。
DEFAULT2	DEFAULT のレスポンスが EXCEPTION に設定されている場合に、バックアップのデフォルト・アクションを提供します。DEFAULT2 は、エラーの発生が予想される MAP 文に例外 MAP 文が指定されていないときに使用します。
<SQL error>	SQL エラー番号。ここには、TRANSDISCARD および TRANSEXCEPTION を使用している場合、レコード・レベルのエラーまたはコミット・レベルのエラーを指定できます。
<user-defined error>	MAP 文の FILTER 句の RAISEERROR オプションで指定されているユーザー定義エラー。

表 39　エラー・レスポンス・オプション

ABEND	トランザクションをロールバックし、処理を異常終了します。ABEND はデフォルトです。
DISCARD	問題のある操作を破棄ファイルに記録しますが、このトランザクションおよび後続のトランザクションの処理を継続します。DISCARDFILE パラメータを使用して破棄ファイルを指定します。

表 39　エラー・レスポンス・オプション (続き)

EXCEPTION	エラーの原因となる個別の操作を例外として処理しますが、トランザクションのその他の操作を正常に処理します。例外後にのみ実行し、失敗した操作を例外表にマップする、例外 MAP 文を使用します。たとえば、失敗した更新文の列を " 行方不明の更新 " 表にマップできます。パラメータ・ファイルでは、エラーの発生が予想される MAP 文の後ろに例外 MAP 文を指定します。 EXCEPTION は、個別のレコードでの個別の SQL 操作にのみ例外処理を適用します。例外処理をトランザクション全体に適用するには、TRANSEXCEPTION オプションを使用します。 *注意*: 競合の検出および解決 (CDR) 機能がアクティブなとき、影響を受ける表への例外 MAP 文が存在する場合、CDR は、エラーを引き起こすすべての操作を自動的に例外として処理します。この場合、REPERROR と EXCEPTION は必要ありませんが、CDR が解決できない競合を処理するその他のオプションとともに、または CDR に処理させない競合に対して、REPERROR を使用する必要があります。 エラー処理の詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。
IGNORE	エラーを無視します。
RETRYOP [MAXRETRIES <n>]	問題のある操作を再試行します。MAXRETRIES オプションでは、再試行回数を制御します。たとえば、表がエクステント不足の場合、RETRYOP と MAXRETRIES によって、トランザクションの失敗を防ぐためにエクステントを追加する時間を確保できます。指定した MAXRETRIES 回数の後、Replicat は異常終了します。試行間隔を設定するには、312 ページで説明されている RETRYDELAY を設定します。
TRANSABORT [, MAXRETRIES <n>] [, DELAYSECS <n> | DELAYCSECS <n>]	トランザクションを中止し、Replicat をトランザクションの開始位置に再配置します。この動作は、レコードの処理が成功するか、MAXRETRIES が終了するまで継続されます。MAXRETRIES が設定されていない場合、TRANSABORT アクションは何度も繰り返されます。 DELAY オプションでは、再試行を遅延させます。デフォルトの遅延は 60 秒です。 TRANSABORT オプションは、タイムアウトやデッドロックをサポートしているデータベースで、このような状況に対処するときに活用できます。

表 39　エラー・レスポンス・オプション (続き)

TRANSDISCARD	トランザクション内の任意の操作 (コミット操作など) が原因で、REPERROR エラー指定されている Replicat エラーが発生する場合、ソース・トランザクション全体を破棄します。レコードでエラーが発生した場合、Replicat はトランザクションを中止し、DISCARDFILE パラメータで指定した破棄ファイルにそのレコードを書き込みます。Replicat はトランザクションをリプレイし、コミット・レコードを含むすべてのレコードを破棄ファイルに書き込みます。Replicat は、破棄処理が原因のエラーの発生時に異常終了します。 破棄レコードがすでにターゲット・レコードにデータ・マッピングされている場合、Replicat はターゲットのフォーマットで破棄ファイルに書き込み、それ以外の場合はソースのフォーマットで書き込みます。リプレイされたトランザクション自体は、常にソースのフォーマットで書き込まれます。 TRANSDISCARD は、コミット・エラーに加え、レコード・レベルのエラーもサポートします。 詳細情報は、このトピックの最初に記載されています。
TRANSEXCEPTION	トランザクションの任意のレコードで、REPERROR で指定されたエラーが発生した場合、例外 MAP 文の MAPEXCEPTION または EXCEPTIONSONLY 句で定義したように、対応する例外マッピング指定に従って、トランザクションのすべてのレコードに例外マッピングを実行します。レコードに対応する例外マッピング指定がない場合、または例外表への書込み時にエラーがある場合、Replicat はエラー・メッセージとともに異常終了します。 エラーが発生して TRANSEXCEPTION が使用されると、レコードでエラーが発生した場合、Replicat はトランザクションを中止し、DISCARDFILE パラメータで指定した破棄ファイルにそのレコードを書き込みます。Replicat はトランザクションをリプレイし、ソース・レコードを調べて例外マッピング指定を探し、それを実行します。 TRANSEXCEPTION は、コミット・エラーに加え、レコード・レベルのエラーもサポートします。詳細情報は、このトピックの最初に記載されています。

例 1　次に、大半のエラーでは処理を停止し、重複レコード・エラーは無視する方法の例を示します。

```
REPERROR (DEFAULT, ABEND)
REPERROR (-1, IGNORE)
```

例 2 次に、account 表のエラーを処理するために作成された例外 MAP 文を呼び出す例を示します。product 表には例外 MAP 文が定義されていないため、この表のエラーによって Replicat は異常終了します。

```
REPERROR (DEFAULT, EXCEPTION)
REPERROR (DEFAULT2, ABEND)
MAP sales.product, TARGET sales.product;
MAP sales.account, TARGET sales.account;
INSERTALLRECORDS
MAP sales.account, TARGET sales.account_exception,
EXCEPTIONSONLY,
COLMAP (account_no = account_no,
optype = @GETENV ("lasterr", "optype"),
dberr = @GETENV ("lasterr", "dberrnum"),
dberrmsg = @GETENV ("lasterr", "dberrmsg"));
```

例 3 次に、最初の MAP 文にエラー・ルールを適用した後、2 つ目の文ではデフォルトの ABEND に戻す例を示します。

```
REPERROR (-1, IGNORE)
MAP sales.product, TARGET sales.product;
REPERROR RESET
MAP sales.account, TARGET sales.account;
```

例 4 次に、トランザクション内のレコードでの操作によってエラー 1403 が生成される場合、問題のあるレコードを破棄し、トランザクション全体をリプレイする例を示します。他のエラー・タイプでは Replicat は異常終了します。

```
REPERROR DEFAULT ABEND
REPERROR 1403 TRANSDISCARD
```

例 5 次に、"tgtexception" という名前の例外表に書き込む例外マッピング指定を探すために、問題のあるレコードを破棄し、トランザクション全体をリプレイする例を示します。他のエラーによって、Replicat は問題のあるレコードを破棄し (該当する場合)、異常終了します。

```
REPERROR DEFAULT ABEND
REPERROR 1403 TRANSEXCEPTION
MAP src, TARGET tgt, &
MAPEXCEPTION (TARGET tgtexception, INSERTALLRECORDS, COLMAP () );
```

REPFETCHEDCOLOPTIONS

適用対象 Replicat

REPFETCHEDCOLOPTIONS パラメータでは、Replicat によるソース・データベースからのフェッチが必要とされていた操作に対応する方法を制御します。Extract プロセスは、トランザクション・レコードに SQL 文を構築するための十分な情報が含まれていないとき、または FETCHCOLS 句が使用されている (355 ページを参照してください) ときに、列データをフェッチします。

デフォルト なし

構文

```
REPFETCHEDCOLOPTIONS
[, INCONSISTENTROW]
[, LATESTROWVERSION {IGNORE | REPORT | DISCARD | ABEND}]
[, MISSINGROW {IGNORE | REPORT | DISCARD | ABEND}]
[, NOFETCH <action>]
[, REDUNDANTROW]
[, SETIFMISSING [<string>]]
[, SNAPSHOTROW]
```

引数	説明
INCONSISTENTROW	行 ID による列データのフェッチは成功したものの、キーが一致しなかったことを示します。行 ID が再利用されたか、この操作の後 (およびフェッチの前) に主キー更新が行われました。デフォルトでは、Replicat はこの行を破棄ファイルに記録し、以降のデータ処理を継続します。
LATESTROWVERSION <action>	表の現在の行から列データがフェッチされた場合のレスポンスを指定します。有効な値 : IGNORE 状況を無視して処理を継続します。 REPORT 状況と行のコンテンツを破棄ファイルにレポートしますが、行の処理を継続します。DISCARDFILE パラメータで、破棄ファイルを指定しておく必要があります。 DISCARD データを破棄し、行は処理しません。DISCARDFILE パラメータで、破棄ファイルを指定しておく必要があります。 ABEND データを破棄し、処理を中止します。DISCARDFILE パラメータで、破棄ファイルを指定しておく必要があります。
NOFETCH <action>	フェッチを防ぎます。このオプションの 1 つの用途は、データベースがスタンバイで、Oracle GoldenGate がデータベースに接続していないときです。このケースでは、データベースからのフェッチが試行されるとエラーが発生します。他のシナリオでも、このパラメータの使用が必要になることがあります。 Oracle GoldenGate が通常フェッチしているデータをフェッチできない場合、ターゲットでデータ整合性の問題が発生する可能性があります。

引数	説明
	次に、NOFETCH が検出された場合に実行可能な有効なアクションを示します。 ABEND 操作を破棄ファイルに書き込み、Replicat プロセスを異常終了させます。これはデフォルトです。 ALLOW レコード長が 0 でないかぎり、操作を処理します。 IGNORE 操作を無視します。(STATOPTIONS 設定に基づいて) フェッチ統計がプロセス・レポートにレポートされている場合、レポートはこの結果によって更新されます。 REPORT レコードを破棄ファイルに書き込み、操作を処理します。 DISCARD レコードを破棄ファイルに書き込み、操作を処理しません。(STATOPTIONS 設定に基づいて) フェッチ統計がプロセス・レポートにレポートされている場合、レポートはこの結果によって更新されます。
`MISSINGROW <action>`	Replicat が処理で行の一部 (変更された値) のみ使用できる場合のレスポンスを指定します。トレイルで行方不明の列データは、変更レコードが作成されてからフェッチがトリガーされるまでの間に行が削除されてしまっているか、必要な行イメージが指定されている UNDO 保存期間以前のものであったため、通常はフェッチできません。 有効な値 : IGNORE 状況を無視して処理を継続します。 REPORT 状況と行のコンテンツを破棄ファイルにレポートしますが、部分的な行の処理を継続します。DISCARDFILE パラメータで、破棄ファイルを指定しておく必要があります。 DISCARD データを破棄し、部分的な行は処理しません。DISCARDFILE パラメータで、破棄ファイルを指定しておく必要があります。 ABEND データを破棄し、処理を中止します。DISCARDFILE パラメータで、破棄ファイルを指定しておく必要があります。
`REDUNDANTROW`	このレコードの列データがフェッチ済のため、列データがフェッチされなかったことを示します。

引数	説明
SETIFMISSING [<string>]	フェッチは成功しなかった (およびトレイル・レコードで値が行方不明である) ものの、ターゲット列が Not NULL 制約を持つ場合に、値を指定します。CHAR および BINARY データ型の値としてオプションの ASCII 文字列を受け付けるか、次のデフォルトを受け付けます。 CHAR、VARCHAR: 単一の空白 BINARY、VARBINARY: NULL バイト TIMESTAMP: 現在の日付 / 時刻 FLOAT、INTEGER: ゼロ SETIFMISSING に加え、MAP 文の COLMAP 句を使用してターゲット列に値をマップできます。(236 ページを参照してください。)
SNAPSHOTROW	列データがスナップショットからフェッチされたことを示します。通常、このオプションは操作のレポートまたは破棄のためにのみ使用されます。

REPLACEBADCHAR

適用対象 Extract および Replicat

REPLACEBADCHAR パラメータでは、文字用の列をマップするときに検出される無効な文字データの代替値を指定します。代替値が指定されていない場合、出力不可能な文字が含まれる文字用の列は、16 進数文字列として出力されます。REPLACEBADCHAR はグローバルに適用されます。

REPLACEBADCHAR は、すべての出力不可能なシングルバイトの非 ASCII 文字をシングルバイトの値に置換することに注意してください。これは、マルチバイトまたは 8 ビットの文字をサポートしていません。

デフォルト UNPRINTABLE

構文

```
REPLACEBADCHAR {<char> | SPACE | NULL | UNPRINTABLE | NONE}
```

引数	説明
<char>	指定したシングルバイト文字に置換します。
SPACE	空白に置換します。
NULL	ターゲット列が NULL 値を受け付ける場合は NULL に置換し、それ以外の場合は空白に置換します。
UNPRINTABLE	無効なデータを含むすべての列を拒否します。
NONE	ダブルバイト文字セット値からデフォルト文字への変換を抑止します。

例 1 次に、無効な文字を空白に置換する例を示します。

```
REPLACEBADCHAR SPACE
```

例 2 次に、出力不可能な文字をカレット記号に置換する例を示します。

```
REPLACEBADCHAR ^
```

REPLACEBADNUM

適用対象 Replicat

REPLACEBADNUM パラメータでは、数字列をマップするときに検出される無効な数字データの代替値を指定します。REPLACEBADNUM はグローバルに適用されます。

デフォルト 無効な数字を NULL に置き換えます。

構文

```
REPLACEBADNUM {<number> | NULL | UNPRINTABLE}
```

引数	説明
<number>	指定した数字に置換します。
NULL	ターゲット列が NULL 値を受け付ける場合は NULL に置換し、それ以外の場合はゼロに置換します。
UNPRINTABLE	出力不可能なデータを含むすべての列を拒否します。プロセスは停止し、不正な値をレポートします。

例 1

```
REPLACEBADNUM 1
```

例 2

```
REPLACEBADNUM NULL
```

REPLICAT

適用対象 Replicat

REPLICAT パラメータでは、オンライン変更同期用の Replicat グループを指定します。このパラメータによって、現在の実行と前回の実行が関連付けられるので、データ変更を継続的に処理し、ソース表とターゲット表の同期性を維持できます。Replicat は継続的に実行し、データ・ソースおよびトレイルにチェックポイントを保持することにより、計画的または計画外のプロセス終了、システム停止、ネットワーク障害が発生した場合にもデータ整合性とフォルト・トレランスを確保します。

Replicat パラメータ・ファイルには REPLICAT または SPECIALRUN のいずれかの指定が必要で、パラメータ・ファイルの最初のエントリにする必要があります。SPECIALRUN の詳細は、332 ページを参照してください。

デフォルト なし

構文

```
REPLICAT <group name>
```

引数	説明
<group name>	ADD REPLICAT コマンドで定義したグループ名。

例

```
REPLICAT finance
```

REPORT

適用対象 Extract および Replicat

REPORT パラメータでは、Extract または Replicat がプロセス・レポートにいつ一時的な実行時統計を生成するかを制御します。統計は既存のレポートに追加されます。デフォルトでは、実行時統計は、プロセスが意図的に中断されないかぎり実行終了時に表示されます。

REPORT の統計は、前回のレポートから繰り越されます。たとえば、プロセスが初日に 1000 万の挿入、2 日目に 2000 万の挿入を実行し、レポートが毎日 3:00 に生成される場合、最初のレポートは最初の 1000 万の挿入をレポートし、次のレポートは前日の 1000 万件と当日の 2000 万件の合計、3000 万件をレポートします。新しいレポートが生成されたときに統計をリセットするには、STATOPTIONS パラメータと RESETREPORTSTATS オプションを使用します。336 ページを参照してください。

プロセス・レポート使用の詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。

デフォルト 各実行の終了時に実行時統計を生成する。

構文

```
REPORT
{AT <hh:mi> |
ON <day> |
AT <hh:mi> ON <day>}
```

引数	説明
AT <hh:mi>	指定の時刻にレポートを生成します。AT を ON なしで指定すると、指定した時刻に毎日レポートが生成されます。
ON <day>	指定の曜日にレポートを生成します。有効な値 : SUNDAY MONDAY TUESDAY WEDNESDAY THURSDAY FRIDAY SATURDAY 大 / 小文字は区別されません。

例 1
```
REPORT AT 17:00
```

例 2
```
REPORT ON SUNDAY AT 1:00
```

REPORTCOUNT

適用対象 Extract および Replicat

REPORTCOUNT パラメータでは、Extract または Replicat が起動以降に処理したトランザクション・レコード数をレポートします。各トランザクション・レコードは、Oracle GoldenGate によって取得されたトランザクション内で実行された論理データベース操作を表します。レコード数は、レポート・ファイルおよび画面に出力されます。

注意 この数は、Oracle GoldenGate トレイルに含まれているレコード数とは異なる場合があります。データが 4K を上回る操作は、複数のトレイル・レコードに保持する必要があります。したがって、レコード数は 1,000 レコード (データベース操作) を示していても、トレイル数ではそれより多くのレコード数を示す場合があります。トレイルのレコード数を取得するには、Logdump ユーティリティを使用します。

レコード数のレポートは、定期的な間隔、または特定のレコード数に到達後の生成をスケジュールできます。レコード数は、レポートからレポートに繰り越されます。

REPORTCOUNT は、パラメータ・ファイルで 1 回のみ使用できます。REPORTCOUNT の複数のインスタンスがある場合、Oracle GoldenGate は最後に指定されているインスタンスを使用します。

デフォルト なし

構文

```
REPORTCOUNT [EVERY] <count>
{RECORDS | SECONDS | MINUTES | HOURS} [, RATE]
```

引数	説明
<count>	レコード数を出力する間隔。
RECORDS | SECONDS | MINUTES | HOURS	<count> の測定単位 (レコード、秒、分、または時間)。
RATE	1 秒当たりの操作数、およびパフォーマンス指標として変更レートをレポートします。例 2 を参照してください。"rate" 統計は、レコード数合計をプロセス起動からの合計時間で割った値です。"delta" 統計は、最後のレポート以降のレコード数を最後のレポート出力時からの時間で割った値です。 注意 : この計算では、マイクロ秒の粒度が使用されます。時間間隔では秒の端数なしで表示され、レート値は整数で表示されます。

例 1 次に、5,000 レコードごとにレコード数を生成する例を示します。

```
REPORTCOUNT EVERY 5000 RECORDS
```

例 2 次に、10 分間隔でレコード数のレポートを生成し、処理統計もレポートする例を示します。

```
REPORTCOUNT EVERY 10 MINUTES, RATE
```

処理統計は次のようになります。

```
12000 records processed as of 2011-01-01 12:27:40 (rate 203,delta 308)
```

REPORTROLLOVER

適用対象 Extract および Replicat

REPORTROLLOVER パラメータでは、プロセスの起動時ではなく、定期的なスケジュールでレポート・ファイルをエージングさせます。エージング・スケジュールを設定することで、長時間または連続的な実行の場合に、アクティブなレポート・ファイルのサイズを制御でき、またアーカイブ・ルーチンに追加されるアーカイブを予測できます。

注意 レポート統計は、レポートからレポートに繰り越されます。新しいレポートで統計をリセットするには、STATOPTIONS パラメータと RESETREPORTSTATS オプションを使用します。

時刻、曜日、またはその両方を指定できます。時刻 (AT オプション) のみで曜日 (ON オプション) を指定しない場合、指定した時刻に毎日レポートが生成されます。

このパラメータによって発生するロールオーバーでは、プロセス・レポートに実行時統計が生成されません。

- レポート・ファイルに実行時統計をいつ生成するかを制御するには、REPORT パラメータを使用します。
- 新しい実行時統計をオンデマンドで生成するには、SEND EXTRACT または SEND REPLICAT コマンドとともに REPORT オプションを使用します。

デフォルト 起動時にレポートをロールオーバーする。

構文

```
REPORTROLLOVER
{AT <hh:mi> |
ON <day> |
AT <hh:mi> ON <day>}
```

引数	説明
`AT <hh:mi>`	ファイルをエージングする時刻。 有効な値 : ◆ hh は 24 時間表記の時間で、1 ～ 23 までの値を受け付けます。 ◆ mi は 00 ～ 59 までの値を受け付けます。

`ON <day>`	ファイルをエージングする曜日。有効な値 :
	`SUNDAY`
	`MONDAY`
	`TUESDAY`
	`WEDNESDAY`
	`THURSDAY`
	`FRIDAY`
	`SATURDAY`
	大 / 小文字は区別されません。

例 1
```
REPORTROLLOVER AT 05:30
```

例 2
```
REPORTROLLOVER ON friday
```

例 3
```
REPORTROLLOVER AT 05:30 ON friday
```

RESTARTCOLLISIONS | NORESTARTCOLLISIONS

適用対象 Replicat

RESTARTCOLLISIONS および NORESTARTCOLLISIONS パラメータでは、競合が原因で Oracle GoldenGate が停止した後に、Replicat が HANDLECOLLISIONS ロジックを適用するかどうかを制御します。デフォルトでは、NORESTARTCOLLISIONS が適用されます。ただし、Oracle GoldenGate 起動後の最初のトランザクションに HANDLECOLLISIONS ロジックを適用する必要がある場合もあります。たとえば、サーバーが強制的にシャット・ダウンされ、データベースが最後の Replicat トランザクションをコミットしたものの、Oracle GoldenGate が確認を受信していない場合です。このケースでは、Replicat は起動時にトランザ

クションを再試行します。HANDLECOLLISIONS は、結果として発生するエラーを自動的に処理します。

RESTARTCOLLISIONS は、最初の Replicat チェックポイント (トランザクション) が完了するまで HANDLECOLLISIONS 機能を有効化します。パラメータ・ファイルに HANDLECOLLISIONS パラメータを指定する必要はありません。最初のチェックポイントの完了後、HANDLECOLLISIONS は自動的に無効化されます。

HANDLECOLLISIONS の詳細は、218 ページを参照してください。

デフォルト NORESTARTCOLLISIONS

構文 RESTARTCOLLISIONS | NORESTARTCOLLISIONS

RETRYDELAY

適用対象 Replicat

RETRYDELAY パラメータでは、失敗した操作を再試行するまでの間隔を指定します。このパラメータは、REPERROR パラメータの RETRYOP オプションを使用する場合に使用します (299 ページを参照してください)。

デフォルト 60 秒

構文 RETRYDELAY <seconds>

引数	説明
<seconds>	再試行までの間隔 (秒)。

例 REPERROR (100, RETRYOP MAXRETRIES 3) RETRYDELAY 30

RMTFILE

適用対象 Extract

RMTFILE パラメータでは、抽出データを書き込むリモート・システム上の抽出ファイル名を定義します。このパラメータは、初期ロード構成に使用します。オンライン変更同期では、RMTTRAIL パラメータを使用します。

抽出ファイルのサイズの上限は 2GB です。

RMTFILE は、RMTHOST 文の後に指定する必要があり、すべての TABLE 文より先に指定する必要があります。

ENCRYPTTRAIL パラメータ (192 ページ) を使用して、このファイル内のデータを暗号化できます。

デフォルト なし

構文

```
RMTFILE <file name>
[, APPEND]
[, PURGE]
[, MAXFILES <number>]
[, MEGABYTES <megabytes>]
[, FORMAT RELEASE <major>.<minor>]
```

引数	説明
`<file name>`	ファイルの相対名または完全修飾名。
`APPEND`	現在のデータをファイルの既存のデータに追加します。APPEND を使用する場合、PURGE は使用しないでください。
`PURGE`	新しいファイルを作成する前に、既存のファイルを削除します。PURGE を使用する場合、APPEND は使用しないでください。
`MAXFILES <number>`	単一のファイルでなく、連続するファイルを作成します。このオプションは、1 つのファイルのサイズがオペレーティング・システムに許可される制限を越える可能性があるときに使用します。 MAXFILES を使用して、必要な数のファイルを作成できます。エージングされたファイルには、6 桁の順序番号が付けられます (例 : datafile000002)。 MAXFILES を使用するときは、MEGABYTES も使用して、連続ファイル内の各ファイルの最大サイズを設定します。 チェックポイントは、これらのファイルには保持されません。
`MEGABYTES <megabytes>`	ファイル (MAXFILES を使用している場合は各ファイル) の最大サイズを定義します。リモート・ファイルのサイズの上限は 2GB です。
`FORMAT RELEASE <major>.<minor>`	Extract から、トレイル、ファイル、または (リモート・タスクの場合) 別のプロセスに送信されるデータのメタデータ・フォーマットを指定します。リーダー・プロセスは、メタデータに基づいて、データ・レコードが自身がサポートしているバージョンかどうかを把握します。メタデータのフォーマットは、Oracle GoldenGate プロセスのリリースによって異なります。古い Oracle GoldenGate リリースには、新しいリリースとは異なるメタデータが含まれます。 ◆ FORMAT は必須のキーワードです。 ◆ RELEASE は Oracle GoldenGate のリリース・バージョンを指定します。<major> はメジャー・バージョン番号で、<minor> はマイナー・バージョン番号です。有効な値は、9.0 から現在の Oracle GoldenGate リリース番号です。(9.0 以前の Oracle GoldenGate リリースを使用している場合は、9.0 または 9.5 を指定してください。) リリース・バージョンは、プログラムによって適切なトレイル・フォーマット互換性レベルにマッピングされます。デフォルトは、このトレイルに書き込むプロセスの現在のバージョンです。

引数	説明
	FORMAT と RECOVERYOPTIONS パラメータの間には、依存関係があります。RECOVERYOPTIONS を APPENDMODE に設定する場合、FORMAT は RELEASE 10.0 以上に設定する必要があります。RECOVERYOPTIONS を OVERWRITEMODE に設定する場合、FORMAT は RELEASE 9.5 以下に設定する必要があります。 Oracle GoldenGate トレイル・ファイルのバージョニングおよびリカバリ・モードの詳細は、297 ページの付録 3 を参照してください。

例 1 `RMTFILE /ggs/dirdat/salesny, MEGABYTES 2, PURGE`

例 2 `RMTFILE /ggs/dirdat/salesny, MEGABYTES 2, FORMAT RELEASE 10.4`

RMTHOST

適用対象 Extract

RMTHOST パラメータでは、次のことを行います。

- ローカル Extract プロセスの接続先のリモート・システムを特定する
- Manager プロセスが実行されているシステムの TCP/IP ポート番号を指定する
- TCP/IP 接続の様々な属性を制御する

このパラメータでは、圧縮、データ暗号化、バッファ属性、TCP/IP ストリーミング、接続タイムアウトしきい値、および接続リクエストの待機時間を制御します。Collector パラメータの設定にも使用できます。

パラメータ・ファイルで複数のリモート・システムを特定するには、次に示す例のように、各システムに対して 1 つの RMTHOST 文を指定し、関連するトレイルおよび表マップの指定を続けます。

```
EXTRACT sales
USERID ggs, PASSWORD AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC &
    AES128, ENCRYPTKEY securekey1
RMTHOST ny, MGRPORT 7888, ENCRYPT AES 192 KEYNAME mykey
RMTTRAIL /ggs/dirdat/aa
TABLE ora.orders;
RMTHOST la, MGRPORT 7888, ENCRYPT AES 192 KEYNAME mykey2
RMTTRAIL /ggs/dirdat/bb
TABLE ora.orders;
```

パッシブ・モードで作成されている Extract には、RMTHOST を使用しないでください。パッシブ Extract の詳細は、17 ページを参照してください。

最適なバッファ・サイズの決定

TCPBUFSIZE オプションでは、大きなパケット・サイズをターゲット・システムに送信できるように、Extract が保持を試みる TCP ソケット・バッファのサイズを制御します。お使いのネットワークに最

適なバッファ・サイズの決定のガイドラインとして、次の式を使用できます。

1. コマンド・シェルから ping コマンドを実行し、次の例に示すような平均ラウンド・トリップ時間 (RTT) を取得します。

```
C:\home\ggs>ping ggsoftware.com
Pinging ggsoftware.com [192.168.116.171] with 32 bytes of data:
Reply from 192.168.116.171: bytes=32 time=31ms TTL=56
Reply from 192.168.116.171: bytes=32 time=61ms TTL=56
Reply from 192.168.116.171: bytes=32 time=32ms TTL=56
Reply from 192.168.116.171: bytes=32 time=34ms TTL=56
Ping statistics for 192.168.116.171:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 31ms, Maximum = 61ms, Average = 39ms
```

2. この値にネットワーク帯域幅を掛けます。たとえば、平均 RTT が 0.08 秒で帯域幅が 100MB/ 秒の場合の最適なバッファ・サイズは次のようになります。

```
0.08 second * 100 megabits per second = 8 megabits
```

3. この結果を 8 で割り、バイト数を決定します (8 ビットから 1 バイトに変換します)。次に例を示します。

```
8 megabits / 8 = 1 megabyte per second
```

TCPBUFSIZE で必要な単位はバイトなので、この値を 1000000 に設定します。

Windows 以外のシステムの最大ソケット・バッファ・サイズは、通常デフォルトで制限されています。Oracle GoldenGate が TCPBUFSIZE で構成するバッファ・サイズを増やすことができるように、システム管理者にソースおよびターゲット・システムのデフォルト値を増やすように依頼してください。

注意 **パフォーマンスが改善するのは、ターゲットの Oracle GoldenGate リリースが 8.0.4 以上の場合のみです。**

サポートされているインターネット・プロトコル

Oracle GoldenGate は、IPv4 および IPv6 プロトコルをサポートしています。インターネット・プロトコルの選択の詳細は、USEIPV6 パラメータを参照してください。

デフォルト なし

構文

```
RMTHOST
{<host name> | <IP address>}
{, MGRPORT <port> | PORT <port>}
[, COMPRESS]
[, COMPRESSTHRESHOLD]
[, ENCRYPT <algorithm> KEYNAME <keyname>
[, KEYNAME <keyname>]
[, PARAMS <collector parameters>]
[, STREAMING | NOSTREAMING]
[, TCPBUFSIZE <bytes>]
[, TCPFLUSHBYTES <bytes>]
[, TIMEOUT <seconds>]
```

引数	説明
{<host name> | <IP address>}	ターゲット・システムの DNS ホスト名または IP アドレス。いずれか 1 つを使用してホストを定義できます。IP アドレスを使用している場合、宛先システムのスタックに応じて、IPv6 または IPv4 のいずれかのマップ・アドレスを使用します。
COMPRESS	送信レコードのブロックを圧縮して、必要な帯域幅を低減します。Oracle GoldenGate は、トレイルに書き込む前にデータを解凍します。COMPRESS での通常の圧縮率は少なくとも 4:1 で、それより高くなる場合もあります。ただし、データを圧縮すると CPU リソースを消費します。
COMPRESSTHRESHOLD	圧縮を行うための最小ブロック・サイズを設定します。有効な値は、0 から 28000 までです。デフォルトは 1,000 バイトです。
ENCRYPT <algorithm> KEYNAME <keyname>	ターゲット・システムに TCP/IP を介して送信されるデータ・ストリームを暗号化します。 ◆ <algorithm> は、使用する暗号化アルゴリズムを次の中から指定します。 ◆ AES128 は、キー・サイズが 128 ビットの AES-128 暗号を使用します。 ◆ AES192 は、キー・サイズが 192 ビットの AES-192 暗号を使用します。 ◆ AES256 は、キー・サイズが 256 ビットの AES-256 暗号を使用します。 ◆ BLOWFISH は、ブロック・サイズが 64 ビットで可変長のキー・サイズが 32 ビットから 128 ビットの Blowfish 暗号化を使用します。BLOWFISH は、以前の Oracle GoldenGate バージョンとの下位互換性を維持するためにのみ使用します。これは AES より安全ではありません。 アルゴリズムを指定しない場合、デフォルトは AES128 です。 ◆ KEYNAME <keyname> は、ENCKEYS 参照ファイル内の暗号化鍵の論理名を指定します。鍵名は ENCKEYS ファイル内の実際の鍵を参照するのに使用されます。 Oracle 以外のデータベースに AES 暗号化を使用するには、プロセスを開始する前に、次のように、環境変数として、Oracle GoldenGate インストール・ディレクトリの lib サブディレクトリのパスを指定する必要があります。 ◆ UNIX: LD_LIBRARY_PATH または SHLIB_PATH 変数への入力として、パスを指定します。次に例を示します。 `setenv LD_LIBRARY_PATH ./lib:$LD_LIBRARY_PATH` ◆ Windows: PATH 変数にパスを追加します。 SETENV パラメータを使用して、プロセスのセッション変数として設定できます。 データ暗号化の使用の詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。
KEYNAME <keyname>	ENCKEYS ファイル内の鍵名。Oracle GoldenGate は、鍵を使用してデータを復号化します。ターゲット・システムの ENCKEYS ファイルに一致する鍵名が存在しない場合、Oracle GoldenGate は異常終了します。ENCKEYS ファイルの作成の詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。

引数	説明
`MGRPORT <port>`	Manager が実行されているリモート・システム上のポート番号。MGRPORT または PORT の指定が必要です。
`PORT <port>`	静的 Collector プロセスのポート番号。Manager ポート (動的 Collector を使用している場合) または静的 Collector ポート番号の指定が必要です。静的 Collector の詳細は、第 4 章を参照してください。
`PARAMS <collector parameters>`	NonStop ターゲット・システム上の Collector パラメータを指定します。注意 : Manager が Collector を動的に起動する場合は、Collector ポート (-p 引数) を指定しないでください。 NonStop プラットフォームの Collector パラメータの詳細は、『Oracle GoldenGate *HP NonStop リファレンス・ガイド*』を参照してください。
`STREAMING | NOSTREAMING`	TCP/IP ストリーミングを制御します。 ◆ STREAMING はデフォルトで、非同期インターネット・ストリーミング・プロトコルを有効化します。STREAMING モードでは、通常、送信者 (プライマリ Extract またはデータ・ポンプ) がチェックポイントを行うか、書込み位置を決定する必要があるとき、レスポンスをリクエストするフラグがパケットに含まれていない場合は、受信者 (Collector) は送信者にデータ・パケットの確認を送信しません。この方法では確認が省略されるため、ネットワークが中断した場合、送信者または受信者のプロセスは終了します。そのため、STREAMING を使用する場合、Manager パラメータ・ファイルで AUTORESTART パラメータを使用して、Extract および Collector が終了した場合に再起動します。 ◆ NOSTREAMING は、同期インターネット・プロトコルを有効化します。NOSTREAMING モードでは、送信者は、パケットを送信し、受信者が確認するまで待機してから次のパケットを送信します。この方法は、ネットワークが中断した場合、送信者または受信者のプロセスをリカバリできるため、より信頼性があります。 受信者のプロセスのホスト・システムがストリーミングを使用するように構成されていない場合、Extract は自動的に同期プロトコルに戻ります。 ストリーミングによって、特に送信の遅延がすでに問題になっているネットワークでは遅延が減少するため、無効化するようリクエストしないかぎり、デフォルトの STREAMING を保持してください。ストリーミングは、Extract が Replicat と直接通信する初期ロード・タスクではサポートされていません。
`TCPBUFSIZE <bytes>`	Extract が保持を試みる TCP ソケット・バッファのサイズ (バイト) を制御します。バッファ・サイズを増やすと、ターゲット・システムにより大きなパケットを送信できます。 実際のバッファ・サイズは、TCP スタックの実装およびネットワークによって決定されます。デフォルトは 30,000 バイトですが、最新のネットワーク構成では、通常これより大きい値がサポートされています。有効な値は、1000 ～ 200000000(2 億) バイトです。ネットワーク管理者とともに最適な値を決定してください。314 ページの「最適なバッファ・サイズの決定」も参照してください。

引数	説明
	ターゲット・システムの Oracle GoldenGate インストールが 8.0.4 より前のリリースの場合は、TCPBUFSIZE の指定にかかわらず、30,000 バイトのバッファのみが使用されます。以前のリリースの Collector プロセスでは、これより大きなパケットはサポートされていません。 初期ロード Extract では、このパラメータを使用しないでください。これは、オンライン Extract グループにのみ有効です。 ターゲット・システムが NonStop の場合は、このパラメータを使用しないでください。
`TCPFLUSHBYTES <bytes>`	ネットワークを介して送信されるデータを収集するバッファのサイズ (バイト) を制御します。この値または FLUSHSECS パラメータの値に到達したときに、データがターゲットにフラッシュされます。 デフォルトは 30,000 バイトです。有効な値は 1000 ～ 200000000(2 億) バイトですが、少なくとも TCPBUFSIZE の値を設定する必要があります。 初期ロード Extract では、このパラメータを使用しないでください。これは、オンライン Extract グループにのみ有効です。 ターゲット・システムが NonStop の場合は、このパラメータを使用しないでください。
`TIMEOUT <seconds>`	Collector が Extract からの接続を待機する時間、および接続を終了する前に Collector が Extract からのハートビート・シグナルを待機する時間を指定します。有効な値は 1 秒から 1800 秒 (30 分) までです。デフォルト値は 300 秒 (5 分) です。本番環境では、タイムアウト設定を非常に低くすることをお薦めしません。エラー・ログに、TCP/IP エラー 10054 (既存の接続がリモート・ホストによって強制的にクローズされる) が発生したことを示す警告が記録されている場合、TIMEOUT の値を増やす必要がある可能性があります。このエラーは、通常、TIMEOUT の値を超えたときに Collector 自体が終了した場合に発生します。このパラメータは、静的な Collector には影響しません。

例 1 `RMTHOST 20.20.20.17, MGRPORT 7809, ENCRYPT AES192, KEYNAME newyork`

例 2 `RMTHOST newyork, MGRPORT 7809, COMPRESS, COMPRESSTHRESHOLD 750, NOSTREAMING`

例 3 `RMTHOST newyork, MGRPORT 7809, TCPBUFSIZE 100000, TCPFLUSHBYTES 300000`

RMTHOSTOPTIONS

適用対象 パッシブ Extract

RMTHOSTOPTIONS パラメータでは、信頼性の低いソース上で**パッシブ**・モードで実行されている Extract グループと、より安全性の高いネットワーク・ゾーン内のターゲット・システム間の TCP/IP 接続属性を制御します。このパラメータでは、圧縮、データ暗号化、バッファ属性、ストリーミングおよび接続リクエストの待機時間を制御します。Collector パラメータの設定にも使用できます。

このパラメータは、リモート接続の確立に必要なホスト情報を提供しないため、RMTHOST パラメータとは異なります。Extract が**パッシブ**・モードで実行されている場合、ソースおよびターゲット間のすべ

ての接続はターゲット上の別名 Extract グループによって確立されます。ゾーン化されたネットワークでの Oracle GoldenGate の使用の詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。

すべてのパラメータ・オプションは、1 つの RMTHOSTOPTIONS 文内に指定する必要があります。複数の RMTHOSTOPTIONS 文が使用されている場合は、パラメータ・ファイルで最後に指定されている文が使用され、他の文は無視されます。RMTHOSTOPTIONS は、ファイル内のすべての RMTHOST 文よりも優先されます。

サポートされている IP プロトコルの詳細は、「RMTHOST」を参照してください。

デフォルト なし

構文

```
RMTHOSTOPTIONS
[, COMPRESS]
[, COMPRESSTHRESHOLD]
[, ENCRYPT <algorithm> KEYNAME <keyname>]
[, PARAMS <collector parameters>]
[, STREAMING | NOSTREAMING]
[, TCPBUFSIZE <bytes>]
[, TCPFLUSHBYTES <bytes>]
[, TIMEOUT <seconds>]
```

引数	説明
COMPRESS	送信レコードのブロックを圧縮して、必要な帯域幅を低減します。Oracle GoldenGate は、トレイルに書き込む前にデータを解凍します。COMPRESS での通常の圧縮率は少なくとも 4:1 で、それより高くなる場合もあります。ただし、データを圧縮すると CPU リソースを消費します。
COMPRESSTHRESHOLD	圧縮を行うための最小ブロック・サイズを設定します。有効な値は、0 から 28000 までです。デフォルトは 1,000 バイトです。
ENCRYPT <algorithm> KEYNAME <keyname>	ターゲット・システムに TCP/IP を介して送信されるデータ・ストリームを暗号化します。 ◆ <algorithm> は、使用する暗号化アルゴリズムを次の中から指定します。 ◆ AES128 は、キー・サイズが 128 ビットの AES-128 暗号を使用します。 ◆ AES192 は、キー・サイズが 192 ビットの AES-192 暗号を使用します。 ◆ AES256 は、キー・サイズが 256 ビットの AES-256 暗号を使用します。 ◆ BLOWFISH は、ブロック・サイズが 64 ビットで可変長のキー・サイズが 32 ビットから 128 ビットの Blowfish 暗号化を使用します。BLOWFISH は、以前の Oracle GoldenGate バージョンとの下位互換性を維持するためにのみ使用します。これは AES より安全ではありません。 アルゴリズムを指定しない場合、デフォルトは AES128 です。 ◆ KEYNAME <keyname> は、ENCKEYS 参照ファイル内の暗号化鍵の論理名を指定します。鍵名は ENCKEYS ファイル内の実際の鍵を参照するのに使用されます。データの暗号化の詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。

引数	説明
	Oracle 以外のデータベースに AES 暗号化を使用するには、プロセスを開始する前に、次のように、環境変数として、Oracle GoldenGate インストール・ディレクトリの lib サブディレクトリのパスを指定する必要があります。 ◆ UNIX: LD_LIBRARY_PATH または SHLIB_PATH 変数への入力として、パスを指定します。次に例を示します。 `setenv LD_LIBRARY_PATH ./lib:$LD_LIBRARY_PATH` ◆ Windows: PATH 変数にパスを追加します。 SETENV パラメータを使用して、プロセスのセッション変数として設定できます。
`PARAMS <collector parameters>`	NonStop ターゲット・システム上の Collector パラメータを指定します。注意 : Manager が Collector を動的に起動する場合は、Collector ポート (-p 引数) を指定しないでください。 NonStop プラットフォームの Collector パラメータの詳細は、『Oracle GoldenGate *HP NonStop リファレンス・ガイド*』を参照してください。
`STREAMING | NOSTREAMING`	TCP/IP ストリーミングを制御します。 ◆ STREAMING はデフォルトで、非同期インターネット・ストリーミング・プロトコルを有効化します。STREAMING モードでは、通常、送信者 (プライマリ Extract またはデータ・ポンプ) がチェックポイントを行うか、書込み位置を決定する必要があるとき、レスポンスをリクエストするフラグがパケットに含まれていない場合は、受信者 (Collector) は送信者にデータ・パケットの確認を送信しません。この方法では確認が省略されるため、ネットワークが中断した場合、送信者または受信者のプロセスは終了します。そのため、STREAMING を使用する場合、Manager パラメータ・ファイルで AUTORESTART パラメータを使用して、Extract および Collector が終了した場合に再起動します。 ◆ NOSTREAMING は、同期インターネット・プロトコルを有効化します。NOSTREAMING モードでは、送信者は、パケットを送信し、受信者が確認するまで待機してから次のパケットを送信します。この方法は、ネットワークが中断した場合、送信者または受信者のプロセスをリカバリできるため、より信頼性があります。 受信者のプロセスのホスト・システムがストリーミングを使用するように構成されていない場合、Extract は自動的に同期プロトコルに戻ります。 ストリーミングによって、特に送信の遅延がすでに問題になっているネットワークでは遅延が減少するため、無効化するようリクエストしないかぎり、デフォルトの STREAMING を保持してください。ストリーミングは、Extract が Replicat と直接通信する初期ロード・タスクではサポートされていません。

引数	説明
TCPFLUSHBYTES <bytes>	ネットワークを介して送信されるデータを収集するバッファのサイズ (バイト) を制御します。この値または FLUSHSECS パラメータの値に到達したときに、データがターゲットにフラッシュされます。 デフォルトは 30,000 バイトです。有効な値は 1000 ～ 200000000(2 億) バイトですが、少なくとも TCPBUFSIZE の値を設定する必要があります。 初期ロード Extract では、このパラメータを使用しないでください。これは、オンライン Extract グループにのみ有効です。 ターゲット・システムが NonStop の場合は、このパラメータを使用しないでください。
TIMEOUT <seconds>	**パッシブ・**モードで実行されている Extract が Collector からの接続を待機する時間、および接続を終了する前に Extract が Collector からのハートビート・シグナルを待機する時間を指定します。有効な値は 1 秒から 1800 秒 (30 分) までです。デフォルト値は 300 秒 (5 分) です。本番環境では、タイムアウト設定を非常に低くすることをお薦めしません。エラー・ログに、TCP/IP エラー 10054 (既存の接続がリモート・ホストによって強制的にクローズされる) が発生したことを示す警告が記録されている場合、TIMEOUT の値を増やす必要がある可能性があります。このエラーは、通常、TIMEOUT の値を超えたときに Extract 自体が終了した場合に発生します。

例 `RMTHOSTOPTIONS ENCRYPT AES192, KEYNAME newyork, COMPRESS, COMPRESSTHRESHOLD 750, TCPBUFSIZE 100000, TCPFLUSHBYTES 300000, NOSTREAMING`

RMTTASK

適用対象 Extract

初期ロード Extract 用の RMTTASK パラメータでは、Oracle GoldenGate ダイレクト・ロードまたは SQL*Loader へのダイレクト・バルク・ロード中に、Replicat 処理タスクを開始します。RMTTASK では、Extract に TCP/IP を介して Replicat と直接通信させ、Collector プロセスまたはディスク・ストレージの使用をバイパスします。また RMTTASK では、Extract から Manager に、Replicat を自動起動させ、実行の終了時に Replicat を停止するようにリクエストさせます。タスクはチェックポイントを作成しません。

次に依存関係にあるパラメータを示します。

- 初期ロード Extract パラメータ・ファイルの各 RMTTASK 文の後に、RMTHOST 文を続ける必要があります。
- 初期ロード Replicat パラメータ・ファイルで、EXTRACT を使用する必要があります。
- 初期ロード Replicat パラメータ・ファイルで、REPLICAT を使用する必要があります。
- ADD EXTRACT コマンドで、SOURCEISTABLE を使用する必要があります。
- ADD REPLICAT コマンドで、SPECIALRUN を使用する必要があります。

RMTTASK は、どのような種類の暗号化もサポートしていません。暗号化を使用するには、ファイルにデータを書き込み、Replicat がこのデータを読み込んでロードする初期ロード方法を使用する必要があります。

RMTTASK は、LOB、LONG、ユーザー定義型 (UDT)、またはサイズが 4k より大きいその他の大きいデータ型を含む列がある表をサポートしていません。

RMTTASK を使用するときは、Replicat を START REPLICAT コマンドで起動しないでください。Replicat は、このタスク中に自動的に起動されます。

初期データ・ロードの実行方法の詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。

デフォルト なし

構文

```
RMTTASK REPLICAT, GROUP <group name>
[FORMAT RELEASE <major>.<minor>]
```

引数	説明
`GROUP <group name>`	ターゲット・システム上の初期ロード Replicat グループ名。
`FORMAT RELEASE <major>.<minor>`	Extract から、トレイル、ファイル、または (リモート・タスクの場合) 別のプロセスに送信されるデータのメタデータ・フォーマットを指定します。リーダー・プロセスは、メタデータに基づいて、データ・レコードが自身がサポートしているバージョンかどうかを把握します。メタデータのフォーマットは、Oracle GoldenGate プロセスのリリースによって異なります。古い Oracle GoldenGate リリースには、新しいリリースとは異なるメタデータが含まれます。 ◆ FORMAT は必須のキーワードです。 ◆ RELEASE は Oracle GoldenGate のリリース・バージョンを指定します。<major> はメジャー・バージョン番号で、<minor> はマイナー・バージョン番号です。有効な値は、9.0 から現在の Oracle GoldenGate リリース番号です。(9.0 以前の Oracle GoldenGate リリースを使用している場合は、9.0 または 9.5 を指定してください。) リリース・バージョンは、プログラムによって適切なトレイル・フォーマット互換性レベルにマッピングされます。デフォルトは、このトレイルに書き込むプロセスの現在のバージョンです。 FORMAT と RECOVERYOPTIONS パラメータの間には、依存関係があります。RECOVERYOPTIONS を APPENDMODE に設定する場合、FORMAT は RELEASE 10.0 以上に設定する必要があります。RECOVERYOPTIONS を OVERWRITEMODE に設定する場合、FORMAT は RELEASE 9.5 以下に設定する必要があります。 Oracle GoldenGate トレイル・ファイルのバージョニングおよびリカバリ・オプションの詳細は、297 ページの付録 3 を参照してください。

例

```
RMTTASK REPLICAT, GROUP initrep, FORMAT RELEASE 10.0
```

RMTTRAIL

適用対象 Extract

RMTTRAIL パラメータでは、GGSCI の ADD RMTTRAIL コマンドで作成されているリモート・トレイルを指定します。RMTTRAIL で指定するトレイルは、関連する TABLE 文よりも先に配置する必要があります。異なるリモート・トレイルを定義するために、複数の RMTTRAIL 文を使用できます。RMTTRAIL より先に RMTHOST パラメータを配置する必要があります。

ENCRYPTTRAIL パラメータ (192 ページ) を使用して、このトレイル内のデータを暗号化できます。

デフォルト なし

構文

```
RMTTRAIL <trail name>
[, FORMAT RELEASE <major>.<minor>]
```

引数	説明
<name>	トレイルの相対パス名または完全修飾パス名。名前には 2 文字を使用します。エージングされたトレイル・ファイルは、この名前に 6 桁の順序番号が追加されます (例：/ggs/dirdat/rt000001)。
FORMAT RELEASE <major>.<minor>	Extract から、トレイル、ファイル、または (リモート・タスクの場合) 別のプロセスに送信されるデータのメタデータ・フォーマットを指定します。リーダー・プロセスは、メタデータに基づいて、データ・レコードが自身がサポートしているバージョンかどうかを把握します。メタデータのフォーマットは、Oracle GoldenGate プロセスのリリースによって異なります。古い Oracle GoldenGate リリースには、新しいリリースとは異なるメタデータが含まれます。 ◆ FORMAT は必須のキーワードです。 ◆ RELEASE は Oracle GoldenGate のリリース・バージョンを指定します。<major> はメジャー・バージョン番号で、<minor> はマイナー・バージョン番号です。有効な値は、9.0 から現在の Oracle GoldenGate リリース番号です。(9.0 以前の Oracle GoldenGate リリースを使用している場合は、9.0 または 9.5 を指定してください。) リリース・バージョンは、プログラムによって適切なトレイル・フォーマット互換性レベルにマッピングされます。デフォルトは、このトレイルに書き込むプロセスの現在のバージョンです。 FORMAT と RECOVERYOPTIONS パラメータの間には、依存関係があります。RECOVERYOPTIONS を APPENDMODE に設定する場合、FORMAT は RELEASE 10.0 以上に設定する必要があります。RECOVERYOPTIONS を OVERWRITEMODE に設定する場合、FORMAT は RELEASE 9.5 以下に設定する必要があります。 Oracle GoldenGate トレイル・ファイルのバージョニングおよびリカバリ・モードの詳細は、297 ページの付録 3 を参照してください。

例 1

```
RMTTRAIL dirdat/ny
```

例 2

```
RMTTRAIL /ggs/dirdat/ny, FORMAT RELEASE 10.4
```

ROLLOVER

適用対象 Extract

ROLLOVER パラメータでは、トレイル・ファイルをいつエージングし、新しいファイルを作成するかを指定します。ROLLOVER はグローバル・パラメータで、パラメータ・ファイルの RMTTRAIL または RMTFILE 文で定義されているすべてのトレイルに適用されます。

ROLLOVER を使用して、特定の期間を表す (例 : 各日) トレイル・ファイルを作成します。これにより、継続処理を促進し、出力を体系化する手段を提供します。また、1 つのファイルを非アクティブ化し、次の実行用に新しいファイルを作成することにより、バッチ実行を体系化する手段を提供します。

ファイルは、トランザクションの途中でなく、トランザクションとトランザクションの間でロールオーバーされるので、データの整合性が確保されます。ファイルがロールオーバーされるときにチェックポイントが作成されるため、処理の際に以前のファイルは不要になります。

ロールオーバーは、実行中にロールオーバーの条件が満たされた場合にのみ行われます。たとえば、ROLLOVER ON TUESDAY が指定され、データ抽出が火曜日に開始される場合、(より詳しい ROLLOVER ルールが指定されていないかぎり) ロールオーバーは次の火曜日まで行われません。最大 30 のロールオーバー・ルールを指定できます。

AT または ON のいずれかのオプションを指定する必要があります。両方のオプションを一緒に、任意の順番で使用できます。AT のみで ON を指定しない場合、指定した時刻に毎日新しいトレイル・ファイルが作成されます。

トレイルの順序番号は 000001 から 999999 まで増やすことが可能で、順序番号は 000000 から再開始します。

デフォルト デフォルトのファイル・サイズに到達したときか、ADD RMTTRAIL または ADD EXTTRAIL コマンドの MEGABYTES オプションで指定されているサイズに到達したときにロールオーバーする。

構文

```
ROLLOVER {AT <hh:mi> | ON <day> | AT <hh:mi> ON <day>} [REPORT]
```

引数	説明
`AT <hh:mi>`	ファイルをエージングする時刻。 有効な値 : ◆ hh は 24 時間表記の時間で、1 ～ 23 までの値が有効です。 ◆ mi は 00 ～ 59 までの値を受け付けます。

`ON <day>`	ファイルをエージングする曜日。
	有効な値 :
	`SUNDAY`
	`MONDAY`
	`TUESDAY`
	`WEDNESDAY`
	`THURSDAY`
	`FRIDAY`
	`SATURDAY`
	大 / 小文字は区別されません。

`REPORT`	最後のレポート生成以降に各表から抽出されたレコード数のレポートを生成します。このレポートは、REPORT パラメータによって他のレポートが生成されないかぎり、対応するトレイルにレコード出力数を示します。

例 1 次に、トレイルを毎日午後 3:00 にエージングする例を示します。

```
ROLLOVER AT 15:00
```

例 2 次に、トレイルを毎週日曜日午前 8:00 にエージングする例を示します。

```
ROLLOVER AT 08:00 ON SUNDAY
```

SEQUENCE

適用対象 Extract

SEQUENCE パラメータでは、Oracle GoldenGate トレイルへの伝播および別のデータベースへの送信のために、順序値をトランザクション・ログから抽出します。現在 Oracle GoldenGate は、Oracle データベースで順序をサポートしています。

> **注意** **順序の DDL サポート (CREATE、ALTER、DROP、RENAME) は、順序値のレプリケーションと共存できますが、順序値のレプリケートには必要ありません。順序値のみをレプリケートするときは、Oracle GoldenGate DDL サポート環境のインストールは*不要です*。SEQUENCE パラメータのみを使用して実行できます。**

Oracle GoldenGate は、ターゲットの順序値が次のようになることを保証します。

- 増分間隔が正の数の場合、ソース値より高い
- 増分間隔が負の数の場合、ソース値より低い

Replicat は挿入を実行するときに、増分方向に応じて、次のいずれかの公式をテストとして適用します。

```
source_highwater_value + (source_cache_size * source_increment_size) <= target_highwater_value
```

または

```
source_highwater_value + (source_cache_size * source_increment_size) >= target_highwater_value
```

この公式が FALSE と評価される場合、ターゲットの順序は (順序が増分されるときは) ソース値よりも高い値に更新され、(順序が減分されるときは) ソース値よりも低い値に更新されます。ターゲットは、常にこの公式のカッコ内の式以上の値である必要があります。たとえば、ソースのウォーターマーク値が 40、CACHE が 20 である場合、ターゲットのウォーターマーク値は少なくとも 60 になる必要があります。

```
40 + (20*1) <60
```

ターゲットのウォーターマーク値が 80 未満の場合、Oracle GoldenGate は順序を更新してウォーターマーク値を増分し、常にソースより高いターゲットの値を維持します。現在のウォーターマーク値を取得するには、次の問合せを実行します。

```
SELECT last_number FROM all_sequences WHERE sequence_owner=upper('SEQUENCEOWNER')
AND sequence_name=upper('SEQUENCENAME');
```

サポートされている処理モード

- SOURCEISTABLE が、Extract パラメータとして、または ADD EXTRACT 内に含まれる Oracle GoldenGate 初期ロード方法では、順序値のレプリケーションはサポートされていません。
- Oracle GoldenGate は、アクティブアクティブの双方向構成での順序値のレプリケーションをサポートしていません。
- Oracle GoldenGate は、高可用性構成で順序値をサポートしています。この構成には、両方のサーバー (プライマリ・サーバとターゲット・フェイルオーバー・サーバー) にプライマリ Extract、データ・ポンプ、および Replicat が含まれます。この構成では、フェイルオーバー・サーバー上の Extract プロセスは非アクティブである必要があり、取得対象の順序は含まれません。高可用性構成での Oracle GoldenGate の構成および操作を行う方法の詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。

- データ・ポンプに書き込むプライマリ Extract に対して SEQUENCE を使用する場合は、データ・ポンプが PASSTHRU モードか NOPASSTHRU モードであるかにかかわらず、データ・ポンプに対して同一の SEQUENCE パラメータを使用する必要があります。ただし、DDL パラメータを使用し、同じデータ・ポンプを介して (順序またはその他のオブジェクトの) DDL 操作を伝播する場合は、データ・ポンプは PASSTHRU モードで動作する*必要があります*。

SEQUENCE 使用のガイドライン

- ソースおよびターゲットの順序のキャッシュ・サイズおよび増分間隔は、同一である必要があります。
- キャッシュ・サイズは、0 (NOCACHE) を含む任意の値に設定できます。
- 順序は、循環を有効または無効に設定できますが、ソース・データベースとターゲット・データベースの設定は同一にする必要があります。
- DDL サポートが有効化されている構成に SEQUENCE を追加するには、INITIALSETUP モードで Oracle GoldenGate DDL オブジェクトを再インストールする必要があります。

エラー処理

- Extract は、順序名を解決できない場合、操作を無視します。
- Replicat による順序のエラー処理を有効化するには、REPERROR パラメータを使用します。このパラメータは、MAP パラメータのオプションとしても、単独のパラメータとしても使用できます。REPERROR は、ターゲットで順序が削除されたかどうかを検出できるので、このパラメータを使用して順序が再作成されるまで順序操作を再試行できます。
- REPERROR は、起動時に行方不明のオブジェクトを処理しません。DDLERROR と IGNOREMISSINGTABLES を使用してください。

その他の重要情報

- データベースによる順序の維持手法にギャップは固有で、発生が予想されるため、Oracle GoldenGate がレプリケートする順序値にはギャップが存在する可能性があります。ただし、ターゲット値は常にソース値より大きくなります。
- Extract が RAC システム上でシングルスレッド・モードで実行中で、ラグがあるノード上で順序が更新された場合、順序の取得に時間がかかる場合があります。これは正常な動作です。
- フェイルオーバーでは、トランザクション・ログまたは Oracle GoldenGate トレイル・ファイルのデータの損失または破損を引き起こす問題が発生すると、レプリケートされた順序の更新値が失われます。
- SEND EXTRACT および SEND REPLICAT を REPORT オプションとともに使用するときに表示される統計には、順序操作は UPDATE として表示されます。

デフォルト なし

構文

```
SEQUENCE <owner>.<sequence>;
```

引数	説明
SEQUENCE <owner>.<sequence>	ソース順序の所有者 (スキーマ) および名前を指定します。サポートされている文字およびグローバリゼーション・サポートについては、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。
;	SEQUENCE パラメータ文を終了します。セミコロンはオプションです。

例 SEQUENCE hr.employees_seq;

SESSIONCHARSET

適用対象 GLOBALS

SESSIONCHARSET パラメータでは、ローカル Oracle GoldenGate インスタンスの Oracle GoldenGate プロセスによって開始されるすべてのデータベース接続に、データベース・セッションの文字セットを設定します。データベースにログインするプロセスには、GGSCI、DEFGEN、Extract および Replicat があります。

このパラメータは、Sybase、Teradata および MySQL をサポートしています。他のデータベースのデータベースの文字セットは、プログラムによって取得されます、

DBLOGIN コマンドの SESSIONCHARSET オプションを使用して、同じ GGSCI セッションで発行されたどのコマンドに対するこの設定よりも優先させることができます。SOURCEDB および TARGETDB パラメータの SESSIONCHARSET を使用して、個別のプロセス・ログインに対するこの設定よりも優先させることができます。

デフォルト オペレーティング・システムの文字セット

構文 SESSIONCHARSET <character set>

引数	説明
<character set>	データベース・セッションの文字セット。

例 SESSIONCHARSET ISO-8859-11

SETENV

適用対象 Extract および Replicat

SETENV パラメータでは、任意の環境変数の値を設定します。Extract または Replicat は、起動時に、環境で設定された値ではなく、このパラメータで指定された値を使用します。

設定する変数ごとに 1 つの SETENV 文を使用します。SETENV 文で設定されたすべての変数は、オペレーティング・システム・レベルで設定されているすべての既存の変数よりも優先されます。

デフォルト なし

構文

```
SETENV (
{<environment_variable> |
    GGS_CacheRetryCount |
    GGS_CacheRetryDelay}
= "<value>"
)
```

オプション	説明
<environment_variable>	設定する環境変数名。
"<value>"	指定する変数の値。この値は引用符で囲む必要があります。
GGS_CacheRetryCount	(SQL Server) システム・アクティビティが多いために、Extract によるソース・トランザクション・ログ・ファイルの読取りがブロックされる場合に、この読取りの再試行回数を制御する Oracle GoldenGate 環境変数です。デフォルトの再試行回数は 10 回です。Extract は、指定された回数の試行後、次のようなエラーとともに異常終了します。 `GGS ERROR 600 [CFileInfo::Read] Timeout expired after 10 retries with 1000 ms delay waiting to read transaction log or backup files.` レポート・ファイルやエラー・ログにタイムアウト・メッセージが記録され続ける場合は、このパラメータで再試行回数を増やしてください。
GGS_CacheRetryDelay	(SQL Server) 前回の試行が失敗したときに、Extract のトランザクション・ログ読取りの再試行までの待機時間 (ミリ秒) を制御する Oracle GoldenGate 環境パラメータです。デフォルトの待機時間は 1000 ミリ秒です。

例 1

独立した SETENV 文を使用すると、環境設定を変更せずに、単一の Oracle GoldenGate インスタンスを複数の Oracle データベース・インスタンスに接続させることができます。次のパラメータ文では、ORACLE_HOME および ORACLE_SID の値を設定します。

```
SETENV (ORACLE_HOME = "/home/oracle/ora9/product")
SETENV (ORACLE_SID = "ora9")
```

例 2

次のパラメータ文では、Extract が異常終了する前に実行するトランザクション・ログ読取り再試行回数を最大 20 回、再試行間隔を 3000 ミリ秒として、SQL Server 環境の Oracle GoldenGate の値を設定します。

```
SETENV (GGS_CacheRetryCount = 20)
SETENV (GGS_CacheRetryDelay = 3000)
```

SHOWSYNTAX

適用対象 Replicat

SHOWSYNTAX パラメータでは、適用する前に各 Replicat SQL 文を表示できるインタラクティブ・セッションを開始します。失敗した SQL 文の構文を表示することにより、問題の原因を診断できます。たとえば、WHERE 句で非索引列を使用していることが判明することがあります。

SHOWSYNTAX 使用の要件

- SHOWSYNTAX を初めて使用するときは、Oracle Support アナリストにサポートをリクエストしてください。これはデバッグ・パラメータで、適切に使用しないと望ましくない結果が生じる場合があります。手動操作が要求され自動処理が中断されるため、処理速度が低下し、バックログや遅延が発生することがあります。
- SHOWSYNTAX を使用するためには、オペレーティング・システムのコマンド・シェルから Replicat を起動する必要があります。Replicat を GGSCI から起動している場合は、SHOWSYNTAX を使用しないでください。
- SHOWSYNTAX はテスト環境で使用してください。本番環境に影響がないように、Replicat グループとターゲット表の複製を作成してください。

SHOWSYNTAX の使用

1. Replicat パラメータ・ファイルに、パラメータごとに 1 行を使用して次に示す順序で次のパラメータを含めます。
   - NOBINARYCHARS
   - NODYNSQL
   - SHOWSYNTAX

   注意 NOBINARYCHARS は、Oracle GoldenGate にバイナリ・データを NULL で終わる文字列として処理させる、ドキュメントに記載されていないパラメータです。このパラメータを使用する前に、Oracle サポートに連絡してください。NODYNSQL を使用すると、Replicat は動的 SQL とバインド変数を使用するかわりに、リテラル SQL 文を使用します。サポートの詳細は、http://support.oracle.com を参照してください。
2. Oracle GoldenGate ホーム・ディレクトリで、オペレーティング・システムのコマンド・シェルから、次の構文を使用して Replicat を起動します。reportfile オプションは使用しないでください。画面に出力する必要があります。

   ```
   replicat paramfile dirprm/<Replicat_name>.prm
   ```
3. 最初の SQL 文がプロンプトとともに表示されます。
   - 現在の文を実行し、次の文を表示するには、Keep Displaying (デフォルト) を選択します。
   - 通常処理を再開して画面への SQL 文の出力を停止するには、Stop Display を選択します。
4. 構文の表示が終了したら、パラメータ・ファイルから SHOWSYNTAX、NOBINARYCHARS および NODYNSQL を削除します。

デフォルト なし

構文 SHOWSYNTAX

SOURCEDB

適用対象 Manager、Extract、DEFGEN

SOURCEDB は、接続情報の一部としてデータ・ソース名または識別子を必要とするデータベースまたはデータ・セット用のパラメータです。SOURCEDB に続く TABLE 文で指定された表は、指定したデータ・ソースのものであるとみなされます。

データ・ソースで要求される認証方法に応じて、SOURCEDB とともに USERID パラメータ (417 ページを参照してください) の使用が必要になることがあります。

- データベース・ログイン情報が要求される場合は、同じパラメータ文内で (必要な場合) SOURCEDB を USERID パラメータとともに使用する必要があります。
- SQL/MX データベースの場合は、SOURCEDB ではカタログを指定し、USERID ではスキーマを指定します。スキーマが省略されると、SOURCEDB はグループに関連付けられているスキーマをデフォルトで使用します。
- オペレーティング・システム・レベルでの認証を許可するデータベースの場合は、USERID なしで SOURCEDB を指定できます。

Manager に対しては、PURGEOLDEXTRACTS など、Manager とソース・データベースとのやり取りを伴う Oracle GoldenGate パラメータを使用するときにのみ、SOURCEDB を使用します。

DB2 LUW の場合は、SOURCEDB 文は別名ではなく実際の名前でデータベースを参照する必要があります。

デフォルト なし

構文

```
SOURCEDB <data source>[, SESSIONCHARSET <character set>]
```

引数	説明
`<data source>`	データ・ソース名。 MySQL データベースの場合、SOURCEDB <database_name>@<host_name> の形式を使用して、ローカル・ホスト・ファイルで localhost の構成が正しくないことによって発生する接続の問題を回避できます。
`SESSIONCHARSET <character set>`	Sybase、Teradata および MySQL をサポートします。プロセス・ログイン・セッションに対するデータベース・セッションの文字セットを設定します。このパラメータは、GLOBALS ファイルで指定されるすべての SESSIONCHARSET より優先されます。

例 1 次に、USERID パラメータがある場合とない場合の SOURCEDB の例を示します。

```
SOURCEDB mydb

SOURCEDB mydb, USERID ggs, &
    PASSWORD AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC &
    AES128, ENCRYPTKEY securekey1
```

例 2 次に、ユーザー・セッションの文字セットを設定する例を示します。

```
SOURCEDB mydb, USERID ggs, &
    PASSWORD AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC &
    AES128, ENCRYPTKEY securekey1, SESSIONCHARSET ISO-8859-11
```

SOURCEDEFS

適用対象 Extract データ・ポンプおよび Replicat

SOURCEDEFS パラメータでは、ソース表またはファイルの定義が含まれるファイル名を指定します。ソース定義は、異種のソースとターゲット間でデータをレプリケートするために Oracle GoldenGate を使用する場合に必要です。お使いの Oracle GoldenGate 構成に応じて、次の 1 つまたは複数のプロセスで SOURCEDEFS を使用します。

- ターゲット・システム上の Replicat プロセス
- ソースまたは仲介システム上のデータ・ポンプ

ソース定義ファイルを生成するには、DEFGEN ユーティリティを使用します。データ・ポンプまたは Replicat を起動する前に、仲介システムまたはターゲット・システムにこのファイルを送信してください。

たとえば、各 SOURCEDEFS ファイルに別々のアプリケーションの定義が保持されている場合など、複数のソース定義ファイルを使用するときは、パラメータ・ファイルで複数の SOURCEDEFS 文を使用できます。

127 ページの「ASSUMETARGETDEFS」も参照してください。

Oracle GoldenGate がメタデータを利用する方法の詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。

デフォルト なし

構文

```
SOURCEDEFS <file name>
```

引数	説明
`<file name>`	ソース・データ定義を含むファイルの相対名または完全修飾名。

例 1

```
SOURCEDEFS dirdef\tcust.def
```

例 2

```
SOURCEDEFS /ggs/dirdef/source_defs
```

SOURCEISTABLE

適用対象 Extract

SOURCEISTABLE パラメータでは、別の表またはファイルへのロードの準備のために、ソース表から直接完全なレコードを抽出します。SOURCEISTABLE は、TABLE 文内に指定されているすべての列データを抽出します。

このパラメータは、次の初期ロード方法に適用されます。

- ファイルから Replicat へのデータのロード
- ファイルからデータベース・ユーティリティへのデータのロード

このパラメータは、次の初期ロード方法では使用*しないでください*。

- Extract がファイルを使用せずに Replicat プロセスにロード・データを直接送信する Oracle GoldenGate ダイレクト・ロード
- SQL*Loader への Oracle GoldenGate ダイレクト・バルク・ロード

これらの処理の場合、SOURCEISTABLE は、パラメータ・ファイルでは使用せずに、ADD EXTRACT の引数として指定します。初期データ・ロード方法の詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。

SOURCEISTABLE を使用する場合は、Extract パラメータ・ファイルの最初のパラメータ文にする必要があります。

SOURCEISTABLE を使用するには、Extract および Replicat パラメータ・ファイルから DDL パラメータを削除し、DDL 抽出およびレプリケーションを無効にします。詳細は、164 ページを参照してください。

デフォルト なし

構文 `SOURCEISTABLE`

SPACESTONULL | NOSPACESTONULL

適用対象 Replicat

SPACESTONULL および NOSPACESTONULL パラメータでは、空白のみを含むソース列をターゲット表で NULL に変換するかどうかを制御します。ターゲット列が NULL 値を受け付ける場合、SPACESTONULL は空白を NULL に変換します。 NOSPACESTONULL は、空白をターゲット列で単一の空白文字に変換します。

これらのパラメータは表に固有です。一方のパラメータは、もう一方のパラメータが見つかるまで、それ以降のすべての MAP 文に有効です。このパラメータは、Oracle のみサポートしています。

デフォルト `NOSPACESTONULL`

構文 `SPACESTONULL | NOSPACESTONULL`

SPECIALRUN

適用対象 Replicat

Replicat パラメータ・ファイルのワンタイム処理実行用の SPECIALRUN パラメータは、Replicat にチェックポイントを作成しないように指示します。ワンタイム実行には開始と終了があるため、チェックポイントは不要です。SPECIALRUN は、特定の初期データ・ロード方法に対して使用します。詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。

Replicat が SPECIALRUN モードの場合は、GGSCI の START REPLICAT コマンドで Replicat を起動しないでください。初期ロード中に自動的に起動されます。

SPECIALRUN を指定するときは、END パラメータを使用する必要があります。Replicat パラメータ・ファイルには、REPLICAT (308 ページを参照してください) または SPECIALRUN のいずれかを指定する必要があります。REPLICAT では、オンライン処理を指定します。

デフォルト なし

構文 `SPECIALRUN`

SQLDUPERR

適用対象 Replicat

SQLDUPERR パラメータでは、重複行を検出したときにデータベースによって返される数値エラー・コードを指定します。重複レコード・エラーは、データベース内の既存レコードと一致する主キーを使用した挿入操作が試みられたときに発生します。

SQLDUPERR は、OVERRIDEDUPS パラメータを使用して特別な重複レコード処理を指定するときに使用する必要があります。複数のデータベース・タイプにレプリケートする場合は、複数の SQLDUPERR インスタンスを使用します。

デフォルト なし

構文 `SQLDUPERR <error number>`

引数	説明
`<error number>`	重複レコードが検出されたときに返される数値エラー・コード。

例 次の文は、Microsoft Access および SQL Server の重複レコード・エラー・コードを示しています。

```
SQLDUPERR -1605
SQLDUPERR -2601
```

SQLEXEC

適用対象 Extract および Replicat

SQLEXEC パラメータは、次のように使用します。

- パラメータ・ファイルのルート・レベルで、SQL ストアド・プロシージャまたは問合せを実行する単独文として使用します。単独文としての SQLEXEC は、Oracle GoldenGate 処理中に TABLE または MAP 文から独立して実行されます。
- データベース・コマンドを実行する単独文として使用します。

> **注意** SQLEXEC は、TABLE (Extract 用) または MAP (Replicat 用) 文の一部として、SQL ストアド・プロシージャまたは問合せを実行するためにも使用できます。この使用方法の詳細は、この章のアルファベット順リファレンスの TABLE および MAP の項を参照してください。

SQLEXEC により、Oracle GoldenGate はデータベースと通信し、データベースによってサポートされているファンクションを実行できます。データベース・ファンクションは、データ抽出およびレプリケーション・プロセスと統合することも、これらのプロセスから独立して使用することもできます。

SQLEXEC でサポートされているデータ型

次に、SQLEXEC によってサポートされているデータベースと、入力および出力パラメータとしてサポートされているデータ型を示します。

- 数値データ型
- 日付データ型
- 文字データ型

単独 SQLEXEC パラメータ使用のガイドライン

- 単独 SQLEXEC 文は、他のパラメータも考慮して、パラメータ・ファイルにリストされている順序で実行されます。
- SQLEXEC プロシージャまたは問合せには、すべての例外処理を含める必要があります。
- 問合せまたはプロシージャは、SQLEXEC 文の実行時に、データベースの有効な SQL 構文を使用して正しく構築されている必要があり、そうでない場合、Replicat は、設定されているエラー処理ルールにかかわらず異常終了します。許可されている SQL 構文の詳細は、データベース・ベンダーによって提供されている SQL リファレンス・ガイドを参照してください。
- SQLEXEC 句よりも先に、Oracle GoldenGate ユーザーのデータベース・ログイン情報を指定する必要があります。Extract の場合は、データベースに適切な SOURCEDB および USERID パラメータを使用します。Replicat の場合は、適切な TARGETDB および USERID パラメータを使用します。
- Oracle GoldenGate プロセスを実行しているユーザーが、SQL を実行するユーザーです。このユーザーは、コマンドおよびストアド・プロシージャの実行、およびデータベース提供のプロシージャのコール権限を持っている必要があります。
- 単独 SQLEXEC 文は、レコードから入力パラメータを受け付けたり、出力パラメータを渡したりするためには使用できません。パラメータの受渡しにストアド・プロシージャおよび問合せを使用するには、TABLE または MAP 文内で SQLEXEC 文を使用します。
- 単独 SQLEXEC 文によって影響を受けるすべてのオブジェクトは、Oracle GoldenGate プロセスの起動前に存在している必要があります。そのため、これらのオブジェクトに対する DDL サポートを無効にする必要があり、無効にしない場合には、SQLEXEC プロシージャまたは問合せがこれらのオブジェクト上で実行される前に、DDL 操作によってオブジェクト構造の変更やオブジェクトの削除が行われることがあります。

Oracle GoldenGate でのストアド・プロシージャおよび問合せの使用の詳細は、『Oracle GoldenGate*Windows and UNIX 管理者ガイド*』を参照してください。

注意 Oracle GoldenGate は、z/OS 上の DB2 に対して、ODBC SQLExecDirect ファンクションを使用して SQL 文を動的に実行します。つまり、接続先のデータベース・サーバーは、SQL 文を動的に準備できる必要があります。ODBC は、実行のたびに (リクエストされる間隔で)SQL 文を準備します。通常は、このことが Oracle GoldenGate ユーザーの問題になることはありません。詳細は、DB2 のマニュアルを参照してください。

構文

```
Procedures:

SQLEXEC "call <procedure name>()"
[EVERY <n> {SECONDS | MINUTES | HOURS | DAYS}]
[ONEXIT]
```

構文

```
Queries:

SQLEXEC "<sql query>"
[EVERY <n> {SECONDS | MINUTES | HOURS | DAYS}]
[ONEXIT]
```

構文

```
Database commands:

SQLEXEC "<database command>"
[EVERY <n> {SECONDS | MINUTES | HOURS | DAYS}]
[ONEXIT]
```

コンポーネント	説明
"call <procedure name> ()"	実行するストアド・プロシージャ名を指定します。この文は、二重引用符で囲む必要があります。call キーワードは必須です。 例 : SQLEXEC "call prc_job_count ()"
"<sql query>"	実行する問合せ名を指定します。問合せは引用符で囲む必要があります。SQLEXEC 問合せ内に引用符で囲まれたオブジェクト名を使用するには、SQL 問合せを二重引用符ではなく一重引用符で囲む必要があり、GLOBALS ファイルで USEANSISQLQUOTES パラメータを使用して、オブジェクトおよびリテラル識別子に SQL-92 ルールを施行する必要があります。次に、問合せ内に引用符で囲まれたオブジェクト名を使用する例を示します。 SQLEXEC 'SELECT "col1" from "schema"."table"' 複数の行にわたる問合せでは、各行で引用符を使用します。最良の結果を得るために、各開始引用符の後ろおよび各終了引用符の前に (または少なくとも各終了引用符の前に) 空白を入力してください。 例 : SQLEXEC " select x from dual "
EVERY <n> {SECONDS | MINUTES | HOURS | DAYS}	単独ストアド・プロシージャまたは問合せを、次の例のように、定義済間隔で実行します。 SQLEXEC "call prc_job_count ()" EVERY 30 SECONDS 間隔は、正の整数である必要があります。
ONEXIT	Extract または Replicat プロセスが正常に停止したときに SQL を実行します。
"<database command>"	データベース・コマンドを実行します。

例 1 SQLEXEC "call prc_job_count ()"

例 2 SQLEXEC " select x from dual "

例 3 SQLEXEC "call prc_job_count ()" EVERY 30 SECONDS

例 4 SQLEXEC "call prc_job_count ()" ONEXIT

例 5 SQLEXEC "SET TRIGGERS OFF"

STARTUPVALIDATIONDELAY[CSECS]

適用対象 Manager

STARTUPVALIDATIONDELAY または STARTUPVALIDATIONDELAYCSECS パラメータでは、START EXTRACT または START REPLICAT コマンドで起動されたプロセスのステータスを Manager が検証するまでの遅延時間を設定します。指定した遅延時間の経過後にプロセスが実行されていない場合、GGSCI プロンプトにエラー・メッセージが表示されます。

これらのパラメータによって、プロセス起動に必要なメモリー不足などが原因で、エラー・メッセージまたはレポートを生成できずに失敗するプロセスが明らかになります。起動時に検証を実施することで、Oracle GoldenGate ユーザーはこのような失敗を把握できます。

デフォルト 0 秒 (起動ステータスを検証しない)

構文

```
STARTUPVALIDATIONDELAY <seconds> | STARTUPVALIDATIONDELAYCSECS <centiseconds>
```

引数	説明
`<seconds> | <centiseconds>`	プロセスのステータスを確認するまでの遅延時間 (秒またはセンチ秒)。

例 次に、Manager が START コマンドの発行後 10 センチ秒待機してからプロセスのステータスを確認する例を示します。

```
STARTUPVALIDATIONDELAYCSECS 10
```

STATOPTIONS

適用対象 Extract および Replicat

STATOPTIONS パラメータでは、STATS EXTRACT または STATS REPLICAT コマンドによって生成される統計表示に含める情報を指定します。このパラメータのオプションは、必要に応じてこれらのコマンドの引数としても有効化できます。

デフォルト 各オプションを参照してください。

構文

```
STATOPTIONS
[, REPORTDETAIL | NOREPORTDETAIL]
[, REPORTFETCH | NOREPORTFETCH]
[, RESETREPORTSTATS | NORESETREPORTSTATS]
```

引数	説明
`REPORTFETCH | NOREPORTFETCH`	Extract に有効です。REPORTFETCH は、FETCHCOLS 句 (355 ページを参照してください) によってトリガーされるフェッチや、トランザクション・レコードに十分な情報が含まれていないときに実行する必要があるフェッチなどの行フェッチの統計を返します。NOREPORTFETCH は、フェッチに関する統計のレポートを無効にします。デフォルトは NOREPORTFETCH です。
`REPORTDETAIL | NOREPORTDETAIL`	Replicat に有効です。REPORTDETAIL は、衝突エラーが原因でレプリケートされなかった操作の統計を返します。出力を有効にすると、このような操作が通常の統計 (実行された INSERT、UPDATE および DELETE 操作)、および詳細な表示の統計でレポートされます。たとえば、10 レコードが INSERT 操作で、キーの重複が原因でこれらすべてのレコードが無視された場合、レポートには INSERT 操作数 10、衝突が原因で破棄された操作数 10 と出力されます。 NOREPORTDETAIL は、衝突に関する統計のレポートを無効にします。デフォルトは REPORTDETAIL です。

引数	説明
RESETREPORTSTATS | NORESETREPORTSTATS	Extract および Replicat に有効です。新しいプロセス・レポートが作成されるときに、REPORT パラメータによって生成される統計をリセットするかどうかを制御します。デフォルトの NORESETREPORTSTATS では、REPORTROLLOVER パラメータに基づいてレポートがロールオーバーされるときに、レポートから次のレポートに統計を繰り越します。統計をリセットするには、RESETREPORTSTATS を使用します。

SYSLOG

適用対象 GLOBALS、Manager

SYSLOG パラメータでは、Oracle GoldenGate が Windows または UNIX システムのシステム・ログに送信するメッセージのタイプを制御します。次のことを実行できます。

- すべての Oracle GoldenGate メッセージを含める
- すべての Oracle GoldenGate メッセージを抑止する
- フィタリングを実行して、情報、警告、またはエラー・メッセージを含めるか、これらのタイプを組み合せて含める

SYSLOG は、GLOBALS パラメータ、Manager パラメータ、またはその両方のパラメータとして使用できます。GLOBALS パラメータ・ファイルで使用する場合は、システム上のすべての Oracle GoldenGate プロセスのメッセージ・フィルタリングを制御します。Manager パラメータ・ファイルで使用する場合は、Manager プロセスのメッセージのみのフィルタリングを制御します。GLOBALS および Manager の両方のパラメータ・ファイルで使用する場合は、Manager プロセスについては Manager 設定が GLOBALS 設定よりも優先されます。したがって、Manager と他のすべての Oracle GoldenGate プロセスに別の設定を使用できます。

デフォルト すべての Oracle GoldenGate メッセージをシステム・ログに書き込む。

構文 `SYSLOG {[ALL | NONE] | [, INFO] [, WARN] [, ERROR]}`

引数	説明
ALL	すべての INFO (情報)、WARN (警告) および ERROR (エラー) メッセージをシステム・ログに送信します。これはデフォルトで、次の設定と同じです。 `SYSLOG INFO, WARN, ERROR` 他のオプションと組み合せることはできません。
NONE	Oracle GoldenGate メッセージをシステム・ログに書き込みません。他のオプションと組み合せることはできません。
INFO	INFO としてレポートされるメッセージをシステム・ログに送信します。WARN および ERROR と組み合せることができます (順序は任意)。
WARN	WARN としてレポートされるメッセージをシステム・ログに送信します。INFO および ERROR と組み合せることができます (順序は任意)。
ERROR	ERROR としてレポートされるメッセージをシステム・ログに送信します。INFO および WARN と組み合せることができます (順序は任意)。

例 次の例はともに、警告およびエラー・メッセージをシステム・ログに送信しますが、情報メッセージは送信しません。

```
SYSLOG WARN, ERROR
```

または

```
SYSLOG ERROR, WARN
```

DEFGEN 用 TABLE

DEFGEN パラメータ・ファイルの TABLE パラメータでは、このユーティリティを実行するソース表を指定します。各 TABLE 文は、セミコロンで終了する必要があります。

デフォルト なし

構文

```
TABLE <[owner.]table>[, DEF <definitions template>];
```

引数	説明
`<[owner.]table>`	表の所有者 (オプション) および名前。Oracle 表の場合を除き、所有者はオプションです。このパラメータは、表についてのみワイルドカード (*) 引数を受け付けます。Oracle GoldenGate は、自動的に内部記憶域を拡大し、最大で 100,000 のワイルドカード・エントリを追跡します。
`DEF <definitions template>`	この表の定義を基盤にする定義テンプレートを指定します。同じ定義を持つ新しい表は、DEFGEN を実行せずに、かつ Oracle GoldenGate プロセスの停止と起動を伴わずに後で追加できます。テンプレートは、TABLE または MAP 文の DEF または TARGETDEF オプションで指定されます。このオプションは、初期ロードではサポートされていません。

例 1

```
TABLE fin.account;
```

例 2

```
TABLE fin.acc*;
```

例 3

```
TABLE fin.acct1, DEF acctdefs;
```

Extract 用 TABLE

適用対象 Extract

Extract パラメータ・ファイルの TABLE パラメータでは、Oracle GoldenGate によって抽出されるオブジェクトを指定します。TABLE は、次に対して有効です。

- データ変更の取得をサポートするオンライン Extract。
- ソース表からの完全なデータ・レコードの抽出をサポートする初期ロード Extract。

注意 TABLE で取得するソース・オブジェクトをレプリケーションのためにターゲット・オブジェクトにマップするには、Replicat パラメータ・ファイルの MAP パラメータ文でソースおよびターゲット表を指定します。

TABLE では、次のオブジェクトを指定できます。

- 索引
- トリガー
- マテリアライズド・ビュー
- 表

注意 取得する順序を指定するには、SEQUENCE パラメータを使用します。

サポートの制限

表に対しては、すべての TABLE オプションを使用できます。次のことを実行できますが、これらに限定されません。

- 表の行を選択およびフィルタする
- 表の列をマップする
- データを変換する
- キー列およびビフォア値を指定する
- ストアド・プロシージャおよび問合せを実行する
- ユーザー・トークンを定義する
- 末尾を切り捨てる
- ユーザー・イグジットにパラメータを渡す

表以外のオブジェクトに対しては、TABLE では取得するオブジェクトの指定のみを行います。

注意 Oracle GoldenGate は、Oracle マテリアライズド・ビューの実際のデータ値のレプリケーションをサポートしています。Oracle GoldenGate は、Oracle と Teradata の索引およびトリガーの DDL レプリケーションをサポートしていますが、これらのオブジェクトのコンテンツはサポートしていません。DDL サポートの詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。

デフォルト なし

構文

```
TABLE <table spec> [, TARGET <table spec>]
[, DEF <definitions template>]
[, TARGETDEF <definitions template>]
[, COLMAP (<column mapping expression>)]
[, {COLS | COLSEXCEPT} (<column specification>)]
[, EVENTACTIONS <action>]
[, EXITPARAM "<parameter string>"]
[, {FETCHCOLS | FETCHCOLSEXCEPT} (column specification)]
[, {FETCHMODCOLS | FETCHMODCOLSEXCEPT} (<column spec>)]
[, FETCHBEFOREFILTER]
[, FILTER (<filter specification>)]
[, GETBEFORECOLS(
    {ON UPDATE | ON DELETE}
    {ALL |
    KEY |
    KEYINCLUDING (<col>[,...]) |
    ALLEXCLUDING (<col>[,...]) }
    [,...]
    )]
[, KEYCOLS (<column specification>)]
[, SQLEXEC (<SQL specification>)]
[, SQLPREDICATE "WHERE <where clause>"]
[, TOKENS (<token specification>)]
[, TRIMSPACES | NOTRIMSPACES]
[, TRIMVARSPACES | NOTRIMVARSPACES]
[, WHERE (<where clause>)]
;
```

表 40　TABLE 構文コンポーネントの概要

コンポーネント	説明
TABLE <table spec>	ソース表を指定します。サポートされている文字およびワイルドカードについては、『Oracle GoldenGate *管理ガイド*』を参照してください。
TARGET <table spec>	ソース表をマップするターゲット表を指定します。オブジェクト名指定のガイドラインは、『Oracle GoldenGate *管理ガイド*』を参照してください。
DEF <definitions template>	ソース定義テンプレートを指定します。
TARGETDEF <definitions template>	ターゲット定義テンプレートを指定します。
COLMAP	異なるソース列およびターゲット列間でレコードをマップします。
COLS | COLSEXCEPT	処理する列を選択または除外します。
EVENTACTIONS (<action>)	指定されているフィルタ・ルールを満たすレコードに基づいて、アクションをトリガーします。

表 40　TABLE 構文コンポーネントの概要 (続き)

コンポーネント	説明
EXITPARAM	リテラル文字列形式でパラメータをユーザー・イグジットに渡します。
FETCHCOLS FETCHCOLSEXCEPT	トランザクション・レコードに値が存在しない場合に、ソース・データベースからの列値のフェッチを有効にします。
FETCHBEFOREFILTER	フィルタの実行前に、FETCHCOLS または FETCHCOLSEXCEPT アクションを実行します。
FETCHMODCOLS | FETCHMODCOLSEXCEPT	トランザクション・ログに列が存在する場合でも、データベースから列値をフェッチさせます。
FILTER	数値に基づいてレコードを選択します。FILTER は、WHERE よりも柔軟に使用できます。
GETBEFORECOLS	列のビフォア・イメージを取得させ、トレイルに書き込ませます。
KEYCOLS	行を一意に特定する列を指定します。
SQLEXEC	ストアド・プロシージャおよび問合せを実行します。
SQLPREDICATE	初期ロードのために WHERE 句を使用して行を選択します。
TOKENS	ユーザー・トークンを定義します。
TRIMSPACES | NOTRIMSPACES	CHAR 列を VARCHAR 列にマッピングする際に、末尾の空白を切り捨てるかどうかを制御します。
TRIMVARSPACES | NOTRIMVARSPACES	VARCHAR 列を CHAR または VARCHAR 列にマッピングする際に、末尾の空白を切り捨てるかどうかを制御します。
WHERE	条件演算子に基づいてレコードを選択します。

TABLE 句でのオブジェクト名の指定

TABLE 句にオブジェクト名を指定するには、『Oracle GoldenGate *管理ガイド*』の「Oracle GoldenGate プロセス・インタフェースのスタート・ガイド」を参照してください。

TARGET の使用

Extract が変換を実行するためにターゲット定義ファイル (TARGETDEFS パラメータで指定) を参照する必要がある場合、または COLMAP オプションを使用する場合は、TABLE 文に TARGET を指定する必要があります。それ以外の場合は省略できます。TARGET を使用することによって、定義ファイルまたは列マップに指定されているレコードの構造を反映して、ソース構造ではなくターゲット構造に基づいて抽出データを特定します。

ソース・システムへのオーバーヘッドの追加を防止するために、列マッピングおよび変換をターゲット・システム上で実行できます。ただし、Windows または UNIX システムから NonStop システムへのレプリケーションでは、これらのファンクションをソースで実行する必要があります。

ソースが複数でターゲットが 1 つのときは、ソースでマッピングと変換を実行するほうが適切な場合もあります。このようなケースで、特に新しいファイルの生成が必要になる頻繁なアプリケーション変更が発生するときは、ターゲットに転送する必要がある各ソース・データベースの複数のソース定義を管理するよりも、各ソースに転送する 1 つのターゲット・ファイルを管理するほうが容易です。

COLMAP の使用

COLMAP では、ソース列を異なる名前のターゲット列に明示的にマップするか、ソース名とターゲット名が同一の場合にデフォルトの列マッピングを指定します。COLMAP は、列データの選択、変換、および移動方法を提供します。データ・ポンプ Extract グループにパススルー・モードで処理される表には、このオプションを使用しないでください。

注意　後続の TABLE 文のすべての表に適用する列マッピングの *グローバル・ルール*を作成するには、COLMATCH パラメータを使用します。

列名での大 / 小文字の区別および特殊文字のサポート

デフォルトでは、Oracle GoldenGate は二重引用符で囲まれた文字列をリテラルとして処理します。大 / 小文字が区別されるか、特殊文字が含まれる列名をサポートするには、USEANSISQLQUOTES パラメータを使用できます。USEANSISQLQUOTES によって、Oracle GoldenGate は、識別子およびリテラル文字列を区切るための引用符を使用する SQL-92 ルールに従うことができます。USEANSISQLQUOTES を有効にすると、Oracle GoldenGate は、二重引用符で囲まれた文字列を列名として、一重引用符で囲まれた文字列をリテラルとして処理します。このサポートは、Oracle GoldenGate インスタンス内のすべてのプロセスにローバルに適用されます。使用方法および制約事項の詳細は、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』の USEANSISQLQUOTES に関する項を参照してください。

データ定義の生成

構造が同一でないソース表とターゲット表に対して COLMAP を使用する場合は、ソース表のデータ定義を生成し、ターゲットに転送し、SOURCEDEFS パラメータを使用して定義ファイルを特定する必要があります。

ソースおよびターゲットが同一の構造とみなされるためには、同一の列名 (該当する場合は大 / 小文字区別を含む) とデータ型が含まれ、かつ各表の列の順序が同一である必要があります。また、両方の表は、文字用の列の列長セマンテック (バイトまたは文字) が同一である必要があります。

両方の表が同一の構造を持ち、変換などのその他の機能のために COLMAP を使用する場合は、ソース定義ファイルは必要ありません。かわりに ASSUMETARGETDEFS パラメータを使用できます。

詳細は、次を参照してください。

- SOURCEDEFS は 331 ページ
- ASSUMETARGETDEFS は 127 ページ
- 『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』のデータ定義ファイルの作成に関する項

デフォルトの列マッピングの使用

ソース列と対応するターゲット列の名前が同一の場合は、明示的なマッピング文ではなく、デフォルト・マッピングを使用できます。デフォルト・マッピングでは、Oracle GoldenGate は名前が同じ列を自動的にマップします。適切な場合は、データ変換も自動的に行います。

デフォルト・マッピングを使用するには、USEDEFAULTS オプションを使用します。デフォルト・マッピングは、明示的なマッピング文でマップされていない列にのみ有効です。

デフォルトでは、SQL Server および Sybase の列は、大 / 小文字区別を考慮して比較が行われます。他のすべてのデータベースでは、列名は名前比較のために大文字に変更されます。これらのデータベースで大 / 小文字が区別される列名をサポートするには、GLOBALS ファイルの USEANSISQLQUOTES パラメータを使用します。これにより SQL-92 ルールが適用され、このルールでは、列名を二重引用符で囲み、リテラルを一重引用符で囲む必要があります。

大 / 小文字が認識される場合、USEDEFAULTS は次の方法で大 / 小文字区別をサポートします。

- 名前および大 / 小文字がターゲット列と正確に一致するソース列が見つかった場合、2 つの列はマップされます。
- 大 / 小文字が一致する列が見つからない場合、大 / 小文字の一致にかかわらずターゲット列と名前が一致する最初の候補のソース列を使用してマップが作成されます。

たとえば、次は大 / 小文字を区別する列を含むソース表とターゲット表です。

ソース表 USER1.SM01:

```
id
owner
created
changed
creator
modifiedBy
comment
COMMENT
```

ターゲット表 USER3.SM01:

```
ID
owner
id
Creator
comment
ModifiedBy
creationDate
alterationDate
Comment
COMMENT
```

次に示すこれらの表に対する列マップには、明示的およびデフォルトの列マッピングが含まれています。

```
TABLE USER1.SM01, TARGET USER3.SM01,
COLMAP (USEDEFAULTS,
    ID = id,
    creationDate = created,
    alterationDate = changed,
    );
```

このマップの結果を次に示します。デフォルト・マッピングでは、該当する場合には大 / 小文字の区別が識別され、それ以外の場合には名前のみが照合されます。2 つのターゲット列は、明示的にマップされておらず、デフォルト・マッピングも確立できなかったため、マップされません。

マッピングのタイプ	マッピングの結果

明示的マッピング :	`ID = id,`
	`creationDate = created,`
	`alterationDate = changed`

デフォルト・マッピング :	`owner = owner,`
	`comment = comment,`
	`COMMENT = COMMENT,`
	`Creator = creator,`
	`ModifiedBy = modifiedby`

マッピングされないターゲット列 :	`id, Comment`

列マッピングの詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。

構文

```
TABLE <table spec>, TARGET <table spec>,
COLMAP (
[USEDEFAULTS, ]
<target column> = <source expression>
[, ...]
);
```

コンポーネント	説明
`<table spec>`	ソースまたはターゲット表。
`<target column> = <source expression>`	ソースおよびターゲット列間のマップを明示的に定義します。列名でサポートされている文字の詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。 <target column> はターゲット列名です。 <source expression> には、次のいずれかを指定できます。 ◆ ソース列名 (例 : ORD_DATE) ◆ 数値定数 (例 : 123) ◆ 引用符で囲まれた文字列定数 (例 : "ABCD") ◆ Oracle GoldenGate 列変換ファンクションを使用する式 (例 : @STREXT (COL1, 1, 3))。列変換ファンクションの詳細は、第 5 章を参照してください。

コンポーネント	説明
USEDEFAULTS	明示的な列マップで指定されていない場合に、同一の名前を持つソース列とターゲット列を自動的にマップします。1 つの列セットには、明示的マップまたは USEDEFAULTS を使用し、両方を使用しないでください。詳細は、343 ページの「デフォルトの列マッピングの使用」を参照してください。USEDEFAULTS は、明示的な列マップの前に指定します。

例 1 `TABLE ggs.tran, TARGET ggs.tran2, COLMAP (loc2 = loc, type2 = type);`

例 2 `TABLE ggs.tran, TARGET ggs.tran2, COLMAP COLMAP ( EUROVAL = "\u20ac0" );`

例 3 `TABLE ggs.tran, TARGET ggs.tran2, COLMAP (SECTION = @STRCAT("\u00a7", SECTION ));`

COLS および COLSEXCEPT の使用

COLS および COLSEXCEPT では、列選択を制御します。

- COLS では、同期するデータが含まれる列を指定します。他のすべての列は Oracle GoldenGate に無視されます。
- COLSEXCEPT では、同期から除外する列を指定します。他のすべての列は Oracle GoldenGate に処理されます。多数の列が含まれる表では、COLS を使用して各列を指定するよりも、COLSEXCEPT を使用するほうが効率的な場合があります。

警告 **キー列は除外*しないでください*。また、Oracle GoldenGate によってサポートされていないデータ型を含む列を除外するために、COLSEXCEPT を使用*しないでください*。COLSEXCEPT は、サポートされていないデータ型を除外しません。**

次に、COLS を使用するための条件を示します。

- 表に 1 つ以上のキー列が含まれているか、TABLE の KEYCOLS オプションで代替キーが定義されている必要があります。
- キー列または KEYCOLS で指定されている列が COLS で指定されている列リストに含まれている必要があります。含まれていない場合、これらの列は取得されず、処理中にエラーが生成されます。(注意 : COLS を指定しない場合、キー列は自動的に取得されます。)

主キーまたは一意キーがない、またはその両方が存在しない場合に TABLE 文の KEYCOLS 句がない場合、Oracle GoldenGate は表のすべての列を使用するため、COLS の指定は不要になります。

データ・ポンプ Extract グループにパススルー・モードで処理される表には、このオプションを使用しないでください。

構文 `TABLE <table spec>, {COLS | COLSEXCEPT} (<column> [, ...]) ;`

コンポーネント	説明

<column>	列名。複数の列を指定するには、次の例のようにコンマ区切りリストを作成します。
	次の例では、列 1 および 3 のみを処理します。
	`TABLE hq.acct, COLS (col1, col3);`
	次の例では、列 4 を除きすべての列を処理します。
	`TABLE hq.acct, COLSEXCEPT (col4);`

注意 **データベースが (変更されないかぎり列値が記録されない) 圧縮更新を使用している場合、COLS で抽出するように指定した列を使用できないことがあります。このような列値を使用するには、TABLE 文の FETCHCOLS オプションを使用するか、列のサプリメンタル・ロギングを有効にします。**

DEF の使用

DEF では、ソース定義テンプレートを指定します。定義テンプレートは、特定のソース表に対して DEFGEN が実行されるときに、このオブジェクトの定義に基づいて作成されます。テンプレートが作成されると、この表と同一の定義を持つ新しいソース表は、DEFGEN を実行せずに、かつ Extract の停止と起動を伴わずに追加できます。DEF で指定されたテンプレートの定義は、定義の参照に使用されます。DEFGEN の詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。

構文

```
TABLE <table spec>, DEF <definitions template>;
```

引数	説明
<definitions template>	DEFGEN パラメータ・ファイルの TABLE の DEF オプションで指定されている定義テンプレート名。テンプレートに含まれる定義は、この TABLE 文の表の定義と同一である必要があります。

例

```
TABLE acct.cust*, DEF custdef;
```

EVENTACTIONS の使用

EVENTACTIONS では、特定のフィルタリング・ルールに適合する、*イベント・レコード*と呼ばれるトランザクション・ログ内のレコードに基づいて、Extract に定義済のアクションを実行させます。このシステムを使用して、データベース・イベントに基づいて Oracle GoldenGate 処理をカスタマイズできます。

警告 **EVENTACTIONS は、ソース・データベースが Teradata で、Extract が最大パフォーマンス・モードで構成されている場合にはサポートされません。**

このシステムは、プロセスの起動、一時停止または停止、変換の実行、または統計のレポートなどに使用されます。イベント・マーカー・システムは、次の目的でも使用できます。

- SQLEXEC またはユーザー・イグジット・ファンクションを実行できる同期ポイントを確立する
- データ検証スクリプトを実行するか電子メールを送信するシェル・コマンドを実行する

- 特定のアカウント番号が検出されたときにトレースを有効化する
- ラグ履歴を取得する
- 1 日の終わりにレポートまたはバッチ・プロセスを実行するプロセスを停止または一時停止する

イベント・マーカー機能は、データ変更のレプリケーションではサポートされていますが、初期ロードではサポートされていません。

イベント・マーカー・システムの使用

このシステムは、次の入力コンポーネントを必要とします。

1. アクションをトリガーする*イベント・レコード*を指定します。イベント・レコードを指定するには、次のいずれかのパラメータ文に、FILTER か WHERE 句、または SQLEXEC 問合せかプロシージャを含めます。
   - Extract パラメータ・ファイルの TABLE 文
   - Replicat パラメータ・ファイルの MAP 文
   - ソース表とターゲット表のマッピングを行わずに EVENTACTIONS アクションを実行できるようにする、Replicat パラメータ・ファイルの特別な TABLE 文
2. イベント・レコードを指定した同じ TABLE または MAP 文に、EVENTACTIONS パラメータと、プロセスが実行するアクションを指定する適切なオプションを含めます。

   注意 **すべてではありませんが、多くの EVENTACTIONS オプションは (Extract 用)TABLE および (Replicat 用)MAP 両方に適用されるため、ここでは両方のプロセスのすべてのオプションを説明します。一方のみに適用されるオプションには、その旨記載しています。**

複数のアクションの組合せ

- すべてではありませんが、多くの EVENTACTIONS オプションは組み合せることができます。目的の達成のために、複数のアクションを組み合せる必要がある場合があります。
- 最初に EVENTACTIONS 文全体が解析されてから、優先度に従って、指定したオプションが実行されます。次のリストでは、**レコードの処理**の前にリストされているアクションは、レコードがトレイルに書き込まれる前またはターゲットに適用される前に発生します (プロセスによって異なります)。**レコードの処理**の後ろにリストされているアクションは、レコードが処理された後に実行されます。
  - TRACE
  - LOG
  - CHECKPOINT BEFORE
  - IGNORE
  - DISCARD
  - SHELL
  - ROLLOVER
  - (レコードの処理)
  - REPORT
  - SUSPEND
  - ABORT
  - CHECKPOINT AFTER
  - FORCESTOP
  - STOP

イベント・レコード自体の処理の制御

標準の方法でイベント・レコード自体の処理を防ぐには、IGNORE または DISCARD オプションを使用します。IGNORE および DISCARD は、レコードより先に評価されるため、これらを使用してイベント・レコードの処理を防ぐことができます。これらのオプションが指定されていない場合、Extract はトレイルにレコードを書き込み、Replicat はレコードに含まれている操作をターゲット・データベースに適用します。

1 つのトランザクションに、イベント・アクションをトリガーするレコードが複数含まれている可能性があることに注意してください。そのようなケースでは、指定されている特定の EVENTACTIONS が複数回実行されることがあります。たとえば、2 回の連続する ROLLOVER アクションをトリガーする 2 つのレコードを検出すると、Extract はトレイルを 2 回ロールオーバーするため、実質上 2 つのトレイル・ファイルの 1 つは空になります。

構文

```
EVENTACTIONS (
[STOP | SUSPEND | ABORT | FORCESTOP]
[IGNORE [RECORD | TRANSACTION [INCLUDEVENT]]
[DISCARD]
[LOG [INFO | WARNING]]
[REPORT]
[ROLLOVER]
[SHELL "<command>" |
    SHELL ("<command>", VAR <variable> = {<column name> | <expression>}
    [, ...][, ...]) ]
[TRACE[2] <trace file> [TRANSACTION] [DDL[INCLUDE] | DDLONLY]
    [PURGE | APPEND]]
[CHECKPOINT [BEFORE | AFTER | BOTH]]
[, ...]
)
```

アクション	説明
STOP	指定のイベント・レコードが検出されたときにプロセスを正常に停止させます。プロセスは、オープンしているトランザクションが完了するまで待機してから停止します。Replicat によってグループ化されたトランザクション、またはバッチ・トランザクションの場合、プロセスは、現在のトランザクション・グループが適用されてから正常に停止します。イベント・レコードがトランザクションの終了点であることも示している場合、プロセスはイベント・レコードの次のレコードで再開します。 プロセスは、トランザクションがまだオープンしているために即座に停止できない場合、メッセージを記録します。ただし、長時間オープンしているトランザクション内でイベント・レコードを検出しても、トランザクションがコミットされていないことを通知する警告メッセージは出しません。したがって、STOP イベントが発生しても、プロセスが長時間実行中になる場合があります。 STOP は、ABORT および FORCESTOP を除く、他の EVENTACTIONS オプションと組み合せることができます。

アクション	説明
SUSPEND	現在の実行のアクティブ・コンテキストを保持し、GGSCI で発行される SEND コマンドに応答できるように、プロセスを一時停止します。プロセスが一時停止すると、INFO コマンドによって RUNNING と示され、RBA フィールドに最後のチェックポイントの位置が表示されます。 処理を再開するには、SEND <group> コマンドを RESUME オプションとともに発行します。 CHECKPOINT BEFORE オプションを SUSPEND とあわせて使用するには、イベント・レコードを、SUSPEND が発生するトランザクションの開始にする必要があります。これにより、一時停止状態の間にプロセスが中断した場合、SUSPEND アクションのイベント・レコードは、再起動時に再処理する最初のレコードになります。CHECKPOINT BERORE および SUSPEND の両方を指定しましたが、イベント・レコードがトランザクションの開始ではない場合、SUSPEND が発生する前にプロセスは異常終了します。 CHECKPOINT AFTER オプションを SUSPEND とあわせて使用するには、チェックポイントが発生する前に RESUME コマンドを発行する必要があり、イベント・レコードを COMMIT レコードにする必要があります。SUSPEND 状態の間にプロセスが中断した場合、プロセスは再起動時に最後にチェックポイントが指定された位置からトランザクションを再処理します。 SUSPEND は、ABORT と組み合せることはできませんが、他のすべてのオプションと組み合せることができます。
ABORT	オープンしているトランザクションの有無に関わらず、指定のイベント・レコードが検出されたときに即座にプロセスを終了させます。イベント・レコードは処理されません。ログに致命的なエラーが書き込まれ、DISCARD も指定している場合には破棄ファイルにイベント・レコードが書き込まれます。プロセスの起動時にリカバリが行われます。 ABORT は、CHECKPOINT BEFORE、DISCARD、SHELL および REPORT とのみ組み合せることができます。
FORCESTOP	指定のイベントが検出され、イベント・レコードがトランザクションの最後の操作またはトランザクションの唯一のレコードの場合にのみ、プロセスを正常に停止させます。レコードは正常に書き込まれます。 長時間オープンしているトランザクション内でイベント・レコードが検出された場合、プロセスはログに警告メッセージを記録し、ABORT と同様に即座に終了します。このケースでは、起動時にリカバリが必要です。長時間におよぶトランザクションの途中で FORCESTOP アクションがトリガーされると、プロセスは警告メッセージを出さずに終了します。 FORCESTOP は、ABORT、STOP、CHECKPOINT AFTER および CHECKPOINT BOTH を除く、他の EVENTACTIONS オプションと組み合せることができます。ROLLOVER とともに使用する場合、ロールオーバーはプロセスが正常に停止したときにのみ行われます。

アクション	説明
IGNORE [RECORD | TRANSACTION [INCLUDEVENT]]	選択したアクションに応じて、トランザクションの一部またはすべてを無視します。 ◆ RECORD はデフォルトです。残りのトランザクションではなく、プロセスで指定されたイベント・レコードのみが無視されます。警告またはメッセージはログに書き込まれませんが、Oracle GoldenGate 統計が更新され、レコードが無視されたことが統計に反映されます。 ◆ TRANSACTION では、イベントをトリガーしたレコードを含むトランザクション全体を無視します。TRANSACTION を使用する場合は、イベント・レコードをトランザクション内の最初のレコードにする必要があります。トランザクションを無視する場合、デフォルトではイベント・レコードも無視されます。TRANSACTION は、TRANS に短縮できます。 ◆ INCLUDEEVENT を TRANSACTION とともに指定すると、イベント・レコードはトレイルまたはターゲットに伝播されますが、関連トランザクションの残りの部分は無視されます。 IGNORE は、ABORT および DISCARD を除く、他のすべての EVENTACTIONS オプションと組み合せることができます。 このアクションは、DDL レコードには無効です。DDL 操作は自律的であるため、レコードを無視することはトランザクション全体を無視することと同等です。
DISCARD	プロセスに次のことを実行させます。 ◆ 指定のイベント・レコードを破棄ファイルに書き込む。 ◆ Oracle GoldenGate 統計を更新し、レコードが破棄されたことを統計に反映させる。 プロセスは、トレイルの次のレコードから処理を再開します。このオプションを使用する場合は、DISCARDFILE パラメータを使用して破棄ファイルの名前を指定します。デフォルトでは、破棄ファイルは作成されません。 DISCARD は、IGNORE を除く、他のすべての EVENTACTIONS オプションと組み合せることができます。
LOG [INFO | WARNING]	指定のイベント・レコードが検出されたときに、プロセスにこのイベントを記録させます。レポート・ファイル、Oracle GoldenGate エラー・ログ、およびシステム・イベント・ログにメッセージが書き込まれます。 次のオプションを使用して、メッセージの重大度を指定します。 ◆ INFO では、重大度の低い情報メッセージを指定します。これはデフォルトです。 ◆ WARNING では、重大度の高い警告メッセージを指定します。 LOG は、ABORT を除く、他のすべての EVENTACTIONS オプションと組み合せることができます。ABORT を使用する場合は、プロセスが終了する前に ABORT によって致命的なエラーが記録されるため、LOG は不要です。

アクション	説明
REPORT	指定のイベント・レコードが検出されたときにプロセスにレポート・ファイルを生成させます。GGSCI で SEND コマンドと REPORT オプションを使用する場合と同じ結果になります。 (DISCARD、IGNORE または ABORT を使用している場合を除き) REPORT メッセージは、イベント・レコードが処理された後に発生するので、レポート・データにはイベント・レコードが含まれます。 REPORT は、他のすべての EVENTACTIONS オプションと組み合せることができます。
ROLLOVER	Extract にのみ有効です。指定のイベント・レコードが検出されたときに、Extract にトレイルの新しいファイルにロールオーバーさせます。ROLLOVER アクションは、Extract がトレイル・ファイルにイベント・レコードを書き込む前に発生するので、DISCARD、IGNORE または ABORT も使用されている場合を除き、イベント・レコードが新しいファイルの最初のレコードになります。 ROLLOVER は、ABORT を除く、他のすべての EVENTACTIONS オプションと組み合せることができます。 注意 : ROLLOVER は、次の理由から ABORT とは組み合せることができません。 ◆ ROLLOVER を使用すると、プロセスがチェックポイントの書込みを行わない。 ◆ ROLLOVER は ABORT よりも先に発生する。 ROLLOVER のチェックポイントがない場合に ABORT を使用すると、Extract は再起動時に前回のトレイル・ファイル内の前回のチェックポイントから再開します。そのため、実質上ロールオーバーがキャンセルされます。
SHELL "<command>"	イベント・レコードが検出されたときに、指定したシェル・コマンドをプロセスに実行させます。SHELL "<command>" は、基本的なシェル・コマンドを実行します。コマンド文字列は、リテラル値で取得され、その方法でシステムに送信されます。コマンドは大 / 小文字が区別され、二重引用符で囲む必要があります。たとえば、次のようになります。 `EVENTACTIONS (SHELL "echo hello world! > output.txt")` シェル・コマンドが成功した場合、プロセスはレポート・ファイルおよびイベント・ログに情報メッセージを書き込みます。コマンドの成功は、UNIX シェル言語に従って、コマンドの終了ステータスで判断します。UNIX シェル言語では、0 が成功を表します。 システム・コールが成功しなかった場合、プロセスは致命的なエラーとともに異常終了します。UNIX シェル言語では、0 以外は失敗を表します。エラー・メッセージは、SHELL コマンド自体の実行にのみ関連し、従属するコマンドの終了ステータスには関連しないことに注意してください。たとえば、SHELL はスクリプトを正常に実行できますが、そのスクリプト内のコマンドは失敗することがあります。 SHELL は、他のすべての EVENTACTIONS オプションと組み合せることができます。

アクション	説明
SHELL ("<command>", VAR <variable> = {<column name> | <expression>} [, ...][, ...])	イベント・レコードが検出されたときに、指定したシェル・コマンドをプロセスに実行させ、パラメータの受渡しをサポートします。コマンドおよびパラメータは、大 / 小文字が区別されます。 デフォルトでは、二重引用符の内側の入力値は、リテラル・テキストとして処理されます。ただし、USEANSISQLQUOTES パラメータを GLOBALS ファイル内で使用して、SQL-92 ルールを適用できます。ここでは、一重引用符で囲まれたテキストはリテラルとして処理され、二重引用符で囲まれたテキストは名前として処理されます。「USEANSISQLQUOTES」も参照してください。 SHELL を引数とともに使用する場合、コマンドおよび引数の文字列全体をカッコで囲む必要があります。たとえば、次のようになります。 `EVENTACTIONS (SHELL ("Current timestamp: $1  SQLEXEC result is $2 ",VAR $1 = @GETENV("JULIANTIMESTAMP"),VAR $2 = mytest.description));` 入力内容は次のようになります。 <command> はコマンドで、システムにリテラルに渡されます。 VAR は、パラメータ入力を開始する必須のキーワードです。 <variable> は、ランタイム変数の値が置き換えられる、プレースホルダ変数のユーザー定義名です。コマンドで使用されない余分な変数は無視されます。VAR 変数名に一致する SHELL コマンド内のリテラルは、置換済の VAR の値によって置き換えられることに注意してください。これは、予期しない結果になる可能性があるため、本番環境に移行する前にコードのテストを実行してください。 <column name > は、列値の前または後 (現在) のイメージになります。 <expression> は、列データや DDL の処理の有無に応じて、次のようになります。 **列データの有効な式は、次のとおりです。** ◆ TABLE 文の TOKENS 句からの値。 ◆ 任意の Oracle GoldenGate 列変換ファンクションからの戻り値。 ◆ SQLEXEC 問合せまたはプロシージャからの戻り値。

アクション	説明
	DDL の有効な式は、次のとおりです。 ◆ @TOKEN ファンクションからの戻り値 (Replicat のみ)。 ◆ @GETENV ファンクションからの戻り値。 ◆ 列データを参照しないその他のファンクション (@DATENOW など) からの戻り値。 ◆ @DDL ファンクションからの戻り値。

`TRACE[2] <trace file>`
`[TRANSACTION]`
`[DDL[INCLUDE] | DDLONLY]`
`[PURGE | APPEND]`
	指定のイベント・レコードが検出されたときに、トレース情報をトレース・ファイルに書き込ませます。TRACE は、ステップバイステップの処理情報を提供します。TRACE2 では、プロセスがほとんどの時間を費やしているコード・セグメントを特定します。
	デフォルト (オプションなし) では、トランザクション境界を考慮しない標準的な DML トレースは、プロセスが終了するまで有効になります。
	◆ <trace file> にはトレース・ファイル名を指定し、TRACE キーワードの直後に配置します。独自のトレース・ファイルを指定することも、単独の TRACE または TRACE2 パラメータを使用して指定しているデフォルトのトレース・ファイルを使用することもできます。
	EVENTACTIONS TRACE が使用されている様々な TABLE または MAP 文で同一のトレース・ファイルを使用できます。複数の TABLE または MAP 文で同一のトレース・ファイル名が指定されていて、TRACE オプションの指定方法が一貫していない場合は、このトレース・ファイルを含む最後に解決された TABLE または MAP のオプションが優先されます。
	◆ TRANSACTION では、プロセスの終了時までではなく、現在のトランザクションの終了時までのトレースのみ有効にします。Replicat のトランザクションの境界は、Replicat によってグループ化されたトランザクションまたはバッチのターゲット・トランザクションではなく、ソース・トランザクションに基づきます。TRANSACTION は、TRANS に短縮できます。このオプションは、DML 操作にのみ有効です。
	◆ DDL[INCLUDE] では、DDL のトレースに加え、DML トランザクション・データ処理もトレースします。DDL または DDLINCLUDE のいずれかが有効です。
	◆ DDLONLY では、DDL をトレースし、DML トランザクション・データはトレースしません。
	これらのオプションは Replicat にのみ有効です。デフォルトでは、DDL トレースは無効化されます。
	◆ PURGE では、追加のトレース・レコードを書き込む前にトレース・ファイルを切り捨て、APPEND では、既存のレコードの末尾に新しいトレース・レコードを書き込みます。デフォルトは APPEND です。

アクション	説明
	TRACE は、ABORT を除く、他のすべての EVENTACTIONS オプションと組み合せることができます。 指定したトレース・ファイルへのトレースを無効にするには、GGSCI の SEND <process> コマンドと TRACE OFF <filename> オプションを発行します。
CHECKPOINT [BEFORE | AFTER | BOTH]	指定のイベント・レコードが検出されたときに、プロセスにチェックポイントを書き込ませます。チェックポイント・アクションは、TABLE または MAP 文に定義されている処理の前後のコンテキストを提供します。このコンテキストは開始点と終了点を持つため、SQLEXEC およびユーザー・イグジットで実行されるファンクションをマッピングするための同期点を提供します。 ◆ BEFORE Extract プロセス用の BEFORE では、Extract がトレイルにイベント・レコードを書き込む前にチェックポイントを書き込みます。 Replicat プロセス用の BEFORE では、Replicat がレコードに含まれている SQL 操作をターゲットに適用する前にチェックポイントを書き込みます。 BEFORE を使用する場合は、イベント・レコードをトランザクションの最初のレコードにする必要があります。イベント・レコードが最初のレコードでない場合、プロセスは異常終了します。BEFORE を使用することで、イベント・レコードから始まるトランザクション以前のすべてのトランザクションを確実にコミットできます。 DDL レコードに EVENTACTIONS を使用する場合、各 DDL レコードは自律的であるため、DDL レコードはトランザクションの開始であることが保証されることに注意してください。そのため、CHECKPOINT BEFORE イベント・アクションは DDL レコードに対して暗黙的です。 CHECKPOINT BEFORE は、すべての EVENTACTIONS オプションと組み合せることができます。 ◆ AFTER Extract 用の AFTER では、Extract がトレイルにイベント・レコードを書き込んだ後にチェックポイントを書き込みます。 Replicat 用の AFTER では、レコードに含まれている SQL 操作を Replicat がターゲットに適用した後にチェックポイントを書き込みます。 AFTER は、チェックポイント・リクエストに注意としてフラグを立て、プロセスが次の可能な機会にのみチェックポイントを発行することを示します。たとえば、イベント・レコードがマルチレコード・トランザクションの 1 つである場合、チェックポイントは、Oracle GoldenGate データ整合性モデルに従って、次のトランザクションの境界で発生します。 DDL レコードに EVENTACTIONS を使用する場合、各 DDL レコードは自律的であるため、DDL レコードはトランザクションの終了 (境界) であることが保証されることに注意してください。そのため、CHECKPOINT AFTER イベント・アクションは DDL レコードに対して暗黙的です。 CHECKPOINT AFTER は、ABORT を除く、すべての EVENTACTIONS オプションと組み合せることができます。

アクション	説明
	◆ BOTH BOTH は、BEFORE および AFTER の組合せです。Extract または Replicat プロセスは、イベント・レコードの処理前および処理後にチェックポイントを書き込みます。 CHECKPOINT BOTH は、ABORT を除く、すべての EVENTACTIONS オプションと組み合せることができます。 CHECKPOINT は、CP に短縮できます。

例 1 次に、特定の受注番号の挿入操作を含むトランザクションのトレースを有効にする例を示します。

```
TABLE source.order, FILTER (@GETENV ("GGHEADER", "OPTYPE") = "INSERT" AND order_no = 1),
EVENTACTIONS (TRACE order_1.trc TRANSACTION);
```

例 2 次に、定義された処理期間の終了時に、トレイルを順序の次のファイルにロールオーバーする例を示します。一連のトレイル・ファイルは、処理期間に基づいてユニットとして結合できます。その動作を説明します。

1. Extract が FILTER 句を満たすレコードを検出すると、ROLLOVER イベント・アクションがソース・データベースに記録されます。
2. トランザクション・ログでこのレコード (*イベント・レコード*) を検出すると、Extract は現在のトレイル・ファイルを閉じ、新しいトレイル・ファイルを開きます。
3. ROLLOVER イベント・アクションと IGNORE アクションを組み合せて使用することで、イベント・レコード自体のトレイル・ファイルへの書込みを防ぎます。

   ```
   TABLE source.event_table, FILTER (@GETENV ("GGHEADER", "OPTYPE") = "INSERT" AND
   order_no = 10,000), EVENTACTIONS (ROLLOVER, IGNORE);
   ```

イベント・マーカー・システムの他の使用例および詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。

EXITPARAM の使用

EXITPARAM では、TABLE 文からのレコードを検出するたびにユーザー・イグジット・ルーチンにパラメータを渡します。データ・ポンプ Extract グループにパススルー・モードで処理される表には、このオプションを使用しないでください。ユーザー・イグジットの詳細は、第 6 章を参照してください。

構文

```
TABLE <table spec>, EXITPARAM "<parameter string>";
```

コンポーネント	説明
`"<parameter string>"`	リテラル文字列のパラメータ。パラメータは二重引用符で囲みます。パラメータ文字列では、最大で 100 文字まで指定できます。

FETCHCOLS および FETCHCOLSEXCEPT の使用

FETCHCOLS および FETCHCOLSEXCEPT では、トランザクション・ログ・レコードに値が存在しない場合に、データベースから列値をフェッチします。このオプションは、データベースが (変更されないかぎり列値は記録されない) 圧縮更新を使用しているものの、FILTER 操作に必要な他の列値を使用できるようにする必要がある場合に使用します。

- FETCHCOLS では、指定した列をフェッチします。
- FETCHCOLSEXCEPT では、指定した列を除くすべての列をフェッチします。多数の列が含まれる表では、FETCHCOLS を使用して各列を指定するよりも、FETCHCOLSEXCEPT を使用するほうが効率的な場合があります。

FETCHCOLS および FETCHCOLSEXCEPT は、Oracle GoldenGate にサポートされているすべてのデータベースに有効です。

Oracle データベースの場合、Oracle GoldenGate は、Oracle の Flashback Query メカニズムを使用して、UNDO 表領域から値をフェッチします。フラッシュバック問合せは、特定の時刻または SCN 時点の列の読取り一貫性イメージを提供します。Oracle GoldenGate の Flashback Query の使用方法の詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。

FETCHCOLS または FETCHCOLSEXCEPT を使用するかわりに、必要な列のサプリメンタル・ロギングを有効にするほうが効率的なこともあります。

Sybase の場合は、Oracle GoldenGate によって Sybase 暗号化データがサポートされていないため、暗号化列データはこれらのパラメータにサポートされていません。

フェッチの制御、およびフェッチ対象に指定した列が見つからない場合のレスポンスを指定するには、FETCHOPTIONS パラメータを使用します。STATS EXTRACT コマンドによって生成される統計表示にフェッチ結果を含めるには、STATOPTIONS パラメータを使用します。

FETCHCOLS または FETCHCOLSEXCEPT で指定した値がトランザクション・ログに存在する場合、データベースのフェッチは実行されません。これにより、データベースのオーバーヘッドが軽減されます。

データ・ポンプ Extract グループにパススルー・モードで処理される表には、このオプションを使用しないでください。

構文

```
TABLE <table spec>, {FETCHCOLS | FETCHCOLSEXCEPT} (<column> [, ...]) ;
```

コンポーネント	説明
`<column>`	次のいずれかを指定できます。 ◆ 列名、または `(COL1, COL2)` のような列名のコンマ区切りリスト。 ◆ `(*)` のようなアスタリスク・ワイルドカード。

FETCHMODCOLS および FETCHMODCOLSEXCEPT の使用

FETCHMODCOLS および FETCHMODCOLSEXCEPT では、列がトランザクション・ログに存在する場合でも、列値をデータベースからフェッチさせます。データベースのタイプに応じて、ログ・レコードに表のすべての列が含まれている場合と、特定のトランザクション操作で変更された列のみが含まれている場合があります。

- FETCHMODCOLS では、指定した列をフェッチします。
- FETCHMODCOLSEXCEPT では、指定した列を除き、トランザクション・ログに存在するすべての列をフェッチします。多数の列が含まれる表では、FETCHMODCOLS を使用して各列を指定するよりも、FETCHMODCOLSEXCEPT を使用するほうが効率的な場合があります。

このオプションは、Oracle GoldenGate によってサポートされているすべてのデータベースに有効です。

使用の制限

- キー列には FETCHMODCOLS および FETCHMODCOLSEXCEPT を使用しないでください。
- データ・ポンプ Extract グループにパススルー・モードで処理される表 (パラメータ・ファイルの PASSTHRU パラメータ) には、FETCHMODCOLS および FETCHMODCOLSEXCEPT を使用しないでください。この処理モードでは、データベース・ログインはサポートされていません。
- (Sybase) 暗号化された列データには FETCHMODCOLS および FETCHMODCOLSEXCEPT を使用しないでください。Oracle GoldenGate では、Sybase 暗号化データはサポートされません。

デフォルト なし

構文
```
TABLE <table spec>, {FETCHMODCOLS | FETCHMODCOLSEXCEPT} (<column spec>);
```

引数	説明
(<column spec>)	次のいずれかを指定できます。 ◆ 列名、または (COL1, COL2) のような列名のコンマ区切りリスト。 ◆ (*) のようなアスタリスク・ワイルドカード。

FETCHBEFOREFILTER の使用

FETCHBEFOREFILTER では、FILTER が実行される前に、FETCHCOLS または FETCHCOLSEXCEPT で指定された列をフェッチします。事前にフェッチすることにより、フィルタに必要な値を使用できます。FETCHBEFOREFILTER を指定しない場合、FETCHCOLS または FETCHCOLSEXCEPT で指定したフェッチは、フィルタが実行されるまで実行されません。

データ・ポンプ Extract グループにパススルー・モードで処理される表には、このオプションを使用しないでください。

構文
```
TABLE <table spec>, FETCHCOLS (<column> [, ...]),
FETCHBEFOREFILTER,
FILTER <filter clause>
;
```

FILTER の使用

FILTER では、数値に基づいてレコードを選択または除外します。フィルタ式では、条件演算子、Oracle GoldenGate 列変換ファンクション、またはその両方を使用できます。

> **注意** **文字列に基づいてフィルタするには、Oracle GoldenGate 文字列ファンクションの 1 つを使用するか (第 5 章を参照してください)、WHERE オプションを使用します。**

FILTER コンポーネントはすべてコンマで区切ります。FILTER 句には、次を含めることができます。

- 数字
- 数字を含む列
- 数字を返すファンクション
- 算術演算子 :

  +(加算)
  -(減算)
  *(乗算)
  /(除算)
  \(余り)

- 比較演算子 :

  >(より大きい)
  >=(以上)
  <(より少ない)
  <=(以下)
  =(等しい)
  <>(等しくない)

  比較から導出した結果はゼロ (FALSE を示す) またはゼロ以外 (TRUE を示す) になります。
- カッコ (式の結果をグループ化)
- 結合演算子 : AND、OR

データ・ポンプ Extract グループにパススルー・モードで処理される表には、このオプションを使用しないでください。

構文

```
TABLE <table spec>
, FILTER (
[, ON INSERT | ON UPDATE| ON DELETE]
[, IGNORE INSERT | IGNORE UPDATE | IGNORE DELETE]
, <filter clause>
);
```

コンポーネント	説明

`<filter clause>`	次のように、式に基づいてレコードを選択します。
	`FILTER ((PRODUCT_PRICE*PRODUCT_AMOUNT)>10000))`
	フィルタ句では、次の例のように、Oracle GoldenGate の列変換ファンクションを使用できます。
	`FILTER (@COMPUTE (PRODUCT_PRICE*PRODUCT_AMOUNT)>10000)`
	デフォルトでは、Oracle GoldenGate は、`FILTER (@STRFIND(NAME, "JOE") > 0)` のように、二重引用符で囲まれた入力文字列をリテラルとして処理します。識別子およびリテラルに SQL-92 ルールを使用するには、GLOBALS ファイルで USEANSISQLQUOTES パラメータを使用します。
	Oracle GoldenGate では、マルチバイト文字セットまたはローカル・オペレーティング・システムの文字セットと互換性のない文字セットを含む列に対して、FILTER をサポートしていません。
	フィルタ句のファイルの最大サイズは 5,000 バイトです。

`ON INSERT | ON UPDATE| ON DELETE`	レコードのフィルタリングを、指定した操作に限定します。次のように、操作はコンマで区切ります。 `FILTER (ON UPDATE, ON DELETE, @COMPUTE (PRODUCT_PRICE*PRODUCT_AMOUNT)>10000)` この例では、更新および削除に対してフィルタを実行しますが、挿入に対しては実行しません。

コンポーネント	説明
IGNORE INSERT | IGNORE UPDATE | IGNORE DELETE	指定した操作にフィルタを適用しません。次のように、操作はコンマで区切ります。 FILTER (IGNORE INSERT, @COMPUTE (PRODUCT_PRICE*PRODUCT_AMOUNT)>10000) この例では、更新および削除に対してフィルタを実行しますが、挿入は無視します。

GETBEFORECOLS の使用

GETBEFORECOLS では、更新または削除操作中にビフォア・イメージを取得し、トレイルに書き込む列を指定します。GETBEFORECOLS は、Oracle GoldenGate が双方向構成またはマルチマスター構成で競合の検出および解決 (CDR) 機能を使用しているときに使用します。更新の場合、特定の列が変更されたかどうかにかかわらず、指定した列のビフォア・イメージがトレイルに含まれます。このパラメータを使用するには、デフォルトでビフォア値を記録しないデータベースに対して、サプリメンタル・ロギングを有効化する必要があります。

同じパラメータ・ファイルで使用されている場合、GETBEFORECOLS は、COMPRESSUPDATES および COMPRESSDELETES よりも優先されます。

このパラメータは、DB2 を除くすべてのデータベースに有効です。Oracle GoldenGate でサポートされるすべてのプラットフォームの DB2 には、GETBEFORECOLS のかわりに GETUPDATEBEFORES パラメータを使用します。

構文

```
TABLE <table spec> , GETBEFORECOLS(
{ON UPDATE | ON DELETE}
{ALL |
KEY |
KEYINCLUDING (<col>[,...]) |
ALLEXCLUDING (<col>[,...]) }
[,...]
)
```

引数	説明
{ON UPDATE | ON DELETE}	指定した列のビフォア・イメージを、更新または削除のために取得するかどうかを指定します。ON UPDATE のみ、または ON DELETE のみ、または両方を使用できます。両方使用する場合、同じ GETBEFORECOLS 句の中に指定します。両方使用する方法は、例を参照してください。

引数	説明
{ALL | KEY | KEYINCLUDING (<col>[,...]) | ALLEXCLUDING (<col>[,...])}	ビフォア・イメージを取得する列を指定します。 ALL: 主キーを含むすべての列のビフォア・イメージを取得します。これは Extract に最も高い処理負荷がかかりますが、すべての列を使用して競合検出の比較を実行し、正確性を最大にすることができます。 KEY: 主キーのビフォア・イメージのみを取得します。このオプションは最速ですが、キーは一致していても非キー列は異なる可能性があるため、最も正確な競合検出は不可能です。KEY はデフォルトです。 KEYINCLUDING: 主キーおよび指定した列のビフォア・イメージを取得します。これは、速度と検出の正確性との間での妥当な方法です。 ALLEXCLUDING: 指定した列を除くすべての列のビフォア・イメージを取得します。多数の列が含まれる表では、KEYINCLUDING よりも ALLEXCLUDING のほうが効率的な場合があります。キー列は除外*しないでください*。

例

次の例では、更新および削除操作の際、キー列に加え、name、address および salary のビフォア・イメージが常にトレイル・ファイルに書き込まれます。

```
TABLE src,
GETBEFORECOLS (
ON UPDATE KEYINCLUDING (name, address,  salary),
ON DELETE KEYINCLUDING (name, address, salary));
```

KEYCOLS の使用

KEYCOLS では、ターゲット表の 1 つ以上の列を一意列として定義します。主に KEYCOLS は、ターゲット表で主キーまたは一意索引が使用できないときに、代替主キーを定義するために使用します。

ソースおよびターゲットのキー列または一意索引列は、データベースで定義されている場合も、KEYCOLS によって代替キーが指定されている場合でも、一致する必要があります。ソース表には、少なくともターゲット表と同じ数のキー列または索引列が含まれている必要があります。そうでなければ、ソースのキー列または索引列を更新する際に、Replicat は余剰なターゲット列のビフォア・イメージを取得できません。

キーを定義する際は、次のガイドラインに従ってください。

- ソース表とターゲット表両方にキーまたは一意索引が含まれていない場合は、TABLE および MAP 文両方で KEYCOLS を使用し、一致する列セットを指定します。
- いずれか一方の表にキーまたは一意索引が含まれていない場合は、その表に対して KEYCOLS を使用し、もう一方の表の実際のキーまたは索引列に一致する列を指定します。一致する列セットを定義できない場合は、TABLE および MAP 文両方で KEYCOLS を使用し、一意の値が含まれる一致する列セットを指定します。KEYCOLS の指定は、既存のキーまたは索引よりも優先されます。
- ターゲット表にソース表よりも大きなキー (または多くの一意索引列) が含まれている場合は、TABLE 文で KEYCOLS を使用し、実際のソースのキーまたは索引列に加え、余剰なターゲット列と一致するソース列を指定する必要があります。表に主キーまたは一意索引が含まれている場合、KEYCOLS の指定はこれらよりも優先されるため、余剰な列のみを指定しないでください。このように KEYCOLS を使用すると、キーまたは索引列は更新のときにビフォア・イメージを利用できます。

KEYCOLS を使用するときは、指定した列をトランザクション・ログに記録し、Replicat がトレイルで使用できるようにしてください。この設定は、データベース・インタフェースを使用するか、ADD TRANDATA コマンドの COLS オプションを使用して行えます (Oracle のみ)。

ターゲット表では、KEYCOLS で定義したキー列に一意索引を作成します。索引によって、Oracle GoldenGate が処理する必要があるターゲット行をより高速に特定できます。

データ・ポンプ Extract グループにパススルー・モードで処理される表には、KEYCOLS を使用しないでください。

構文

```
TABLE <table spec>, KEYCOLS (<column> [, ... ]);
```

コンポーネント	説明
(<column>)	代替主キーとして使用する列を定義します。複数の列を指定するには、次のようにコンマ区切りリストを作成します。 `KEYCOLS (id, name)` 主キーまたは一意キーが存在する場合は、これらの列を KEYCOLS 指定に含める必要があります。次の列タイプは KEYCOLS でサポートされていません。 **KEYCOLS でサポートされていない Oracle 列タイプ** : 仮想列、UDT、ファンクションベース列、および Oracle GoldenGate 構成から明示的に除外されているすべての列 **KEYCOLS でサポートされていない SQL Server、DB2 LUW、DB2 z/OS、MySQL、SQL/MX、Teradata、TimesTen の列タイプ** : タイプスタンプまたは非マテリアライズド計算結果列を含む列、および Oracle GoldenGate 構成から除外されているすべての列。SQL Server の場合、Oracle GoldenGate は、主キーがないターゲット表の行のデータの長さの合計を、強制的に 8000 バイトより小さくします。 **KEYCOLS でサポートされていない Sybase 列タイプ** : 計算結果列、ファンクションベース列、および GoldenGate 構成から明示的に除外されているすべての列

SQLEXEC の使用

SQLEXEC では、Oracle GoldenGate 処理中に TABLE 文内から SQL ストアド・プロシージャまたは問合せを実行します。SQLEXEC により、Oracle GoldenGate はデータベースと直接通信し、データベースによってサポートされているファンクションを実行できます。データベース・ファンクションは、列変換のための値取得などの同期プロセスの一部として使用することも、データの抽出またはレプリケーションと関係なく使用することもできます。

TABLE 文内で使用する場合、実行されるプロシージャまたは問合せは、ソースまたはターゲット行から入力パラメータを受け付け、出力パラメータを渡すことができます。

Oracle GoldenGate でのストアド・プロシージャおよび問合せの使用の詳細は、『Oracle GoldenGate*Windows and UNIX 管理者ガイド*』を参照してください。

SQLEXEC の依存関係および制約事項

- SQL は、Oracle GoldenGate プロセスを実行しているユーザーによって実行されます。このユーザーは、ストアド・プロシージャの実行、およびデータベース提供のプロシージャのコール権限を持っている必要があります。
- 問合せまたはプロシージャは、SQLEXEC 文の実行時に、データベースの有効な構文を使用して正しく構築されている必要があり、そうでない場合、設定されているエラー処理ルールにかかわらず、Oracle GoldenGate は異常終了します。許可されている SQL 構文の詳細は、データベース・ベンダーによって提供されている SQL リファレンス・ガイドを参照してください。
- SQLEXEC は、主キー列の値を変更するために使用しないでください。主キーの値は、Extract から Replicat に渡されます。主キーの値がない場合、Replicat 操作は完了できません。主キーの値を SQLEXEC で変更する必要がある場合は、元のキーの値を別の列にマッピングした後、KEYCOLS オプションでこの列を代替キーとして定義することにより、エラーを回避できます。360 ページの「KEYCOLS の使用」を参照してください。
- Oracle GoldenGate は、z/OS 上の DB2 に対して、ODBC SQLExecDirect ファンクションを使用して SQL 文を動的に実行します。つまり、接続先のデータベース・サーバーは、SQL 文を動的に準備できる必要があります。ODBC は、実行のたびに (リクエストされる間隔で)SQL 文を準備します。通常は、このことが Oracle GoldenGate ユーザーの問題になることはありません。詳細は、z/OS 上の DB2 のマニュアルを参照してください。
- データ・ポンプ Extract グループにパススルー・モードで処理される表には、SQLEXEC を使用しないでください。
- Oracle GoldenGate DDL サポートを使用する場合は、SQL の実行前に、ストアド・プロシージャまたは問合せに影響を受けるすべてのオブジェクトが、正しい構造で存在している必要があります。したがって、これらのオブジェクトの構造に影響する DDL (CREATE や ALTER など) は、SQLEXEC の実行前に実行される必要があります。

サポートされるデータ型

入力パラメータおよび出力パラメータとして SQLEXEC でサポートされるデータ型は次のとおりです。

- 数値データ型
- 日付データ型
- 文字データ型

Oracle GoldenGate でのストアド・プロシージャおよび問合せの使用の詳細は、『Oracle GoldenGate*Windows and UNIX 管理者ガイド*』を参照してください。

SQLEXEC とストアド・プロシージャの使用

TABLE 文内からストアド・プロシージャを実行するには、SPNAME 句を使用します。

構文

```
SQLEXEC (
SPNAME <sp name>
[, ID <logical name>]
{, PARAMS <param spec> | NOPARAMS}
[, <option>] [, ...]
)
```

コンポーネント	説明
`<sp name>`	ストアド・プロシージャ名を指定します。
`ID <logical name>`	プロシージャの論理名を定義します。このオプションは、TABLE 文内でプロシージャを複数回実行するときに使用します。1 つの TABLE 文で、最大 20 のストアド・プロシージャを実行できます。プロシージャの実行が 1 回の場合、ID は必要ありません。
`PARAMS <param spec> | NOPARAMS`	プロシージャがパラメータを受け付けるかどうかを定義します。PARAMS <param spec> または NOPARAMS のいずれかを使用する必要があります。<param spec> では、入力パラメータおよび入力のソースを定義します。

`<option>`	ストアド・プロシージャの影響を制御するために、単独または他のオプションと組み合せて使用できる、次のいずれかのオプションを指定します。
	`AFTERFILTER | BEFOREFILTER`
	`ALLPARAMS`
	`DBOP`
	`ERROR`
	`EXEC`
	`MAXVARCHARLEN`
	`PARAMBUFSIZE`
	`TRACE`

SQLEXEC コンポーネントの説明は、アルファベット順に 366 ページから記載されています。

SQLEXEC と問合せの使用

TABLE 文内から問合せを実行するには、ID および QUERY 句を使用します。

構文

```
SQLEXEC (
ID <logical name>
, QUERY "<sql query>"
{, PARAMS <param spec>| NOPARAMS}
[, <option>] [, ...]
)
```

コンポーネント	説明
`ID <logical name>`	問合せの論理名を定義します。問合せの結果から値を抽出するには、論理名が必要です。ID <logical name> は、問合せから返された列値を参照します。

コンポーネント	説明
`QUERY "<sql query>"`	データベースに対して実行する SQL 問合せの構文を指定します。問合せは、問合せを実行するデータベースの有効な標準問合せ言語を使用している必要があります。 問合せは、SELECT 文の結果を返すか、INSERT、UPDATE、または DELETE 文を実行できます。SELECT 文の出力を生成する問合せの場合は、SELECT によって最初に返される行のみが処理されます。SELECT 文には "INTO ..." 句を指定しないでください。 問合せはすべてを 1 行に収め、引用符で囲む必要があります。SQLEXEC 問合せ内に引用符で囲まれたオブジェクト名を使用するには、SQL 問合せを二重引用符ではなく一重引用符で囲む必要があり、GLOBALS ファイルで USEANSISQLQUOTES パラメータを使用して、オブジェクトおよびリテラル識別子に SQL-92 ルールを施行する必要があります。次に、問合せ内に引用符で囲まれたオブジェクト名を使用する例を示します。 `SQLEXEC 'SELECT "col1" from "schema"."table"'` 最良の結果を得るために、各開始引用符の後ろおよび各終了引用符の前に空白を入力してください。
`PARAMS <param spec>| NOPARAMS`	問合せがパラメータを受け付けるかどうかを定義します。これらのオプションのいずれかを使用する必要があります。<param spec> では、入力パラメータおよび入力ソースを定義します。

`<option>`	問合せの影響を制御するために、単独または他のオプションと組み合せて使用できる、次のいずれかのオプションを指定します。
	`AFTERFILTER | BEFOREFILTER`
	`ALLPARAMS`
	`DBOP`
	`ERROR`
	`EXEC`
	`MAXVARCHARLEN`
	`PARAMBUFSIZE`
	`TRACE`

SQLEXEC コンポーネントの説明は、アルファベット順に 366 ページから記載されています。

入力パラメータ用のプレースホルダの使用

ほとんどの問合せは、入力パラメータ用のプレースホルダを必要とします。問合せ内でのパラメータの指定方法は、データベースのタイプによって異なります。

- Oracle の場合は、次の例のように、入力パラメータはコロン (:) を使用して指定し、その後にパラメータ名を続けます。
  `"SELECT NAME FROM ACCOUNT WHERE SSN = :SSN AND ACCOUNT = :ACCT"`
- 他のデータベースの場合は、次の例のように、入力パラメータは疑問符を使用して指定します。
  `"SELECT NAME FROM ACCOUNT WHERE SSN = ? AND ACCOUNT = ?"`

どのデータベースでも、パラメータ名を引用符で囲む必要はありません。

パラメータの値渡し

Oracle GoldenGate は、入力値および出力値をプロシージャまたは問合せとやり取りするためのオプションを提供しています。

- ストアド・プロシージャまたは問合せ内で入力パラメータにデータ値を渡すには、SQLEXEC の PARAMs オプションを使用します (371 ページを参照してください)。
- ストアド・プロシージャまたは問合せから入力として値を FILTER または COLMAP 句に渡すには、次の構文を使用します。

  ```
  {<procedure name> | <logical name>}.<parameter>
  ```

  条件 :

  - <procedure name> は、ストアド・プロシージャの実際の名前で、SQLEXEC 文の SPNAME に指定している値と一致する必要があります。この引数は、Oracle GoldenGate 実行中にプロシージャを 1 回実行する場合にのみ指定します。
  - <logical name> は、SQLEXEC 文の ID オプションで指定した論理名です。この引数は、TABLE 文内でプロシージャを複数回実行する場合に、問合せまたはストアド・プロシージャのインスタンスから値を渡すときに使用します。
  - <parameter> は、参照表の列などのパラメータ名か、返された値を抽出する場合の RETURN_VALUE のいずれかです。

上記の構文のかわりとして、@GETVAL ファンクションを使用できます。詳細は、457 ページを参照してください。

入力パラメータのネーミングには、次のような異なる構成があります。

- Oracle は、次の例のように、入力パラメータに論理名を付けることを許可します。

  ```
  SQLEXEC (ID appphone, QUERY " select per_type from ps_personal_data "
      " where emplid = :vemplid "
      " and per_status = 'N' and per_type = 'A' ",
      PARAMS (vemplid = emplid)),
  TOKENS (applid = @GETVAL(appphone.per_type));
  ```

- 他のデータベースでは、入力パラメータに、P1、P2 のような入力パラメータごとに番号を増分させた名前を付ける必要があります。データベースによっては "p" を大文字または小文字にする必要があることに注意します。このタイプの入力パラメータの例を次に示します。

  ```
  SQLEXEC (ID appphone, QUERY " select per_type from ps_personal_data "
      " where emplid = ? "
      " and per_status = 'N' and per_type = 'A' ",
      PARAMS (p1 = emplid)),
  TOKENS (applid = @GETVAL(appphone.per_type));
  ```

次に、Oracle のソース表、ターゲット表、および参照表と、これらの表のパラメータをストアド・プロシージャの単一のインスタンス、およびストアド・プロシージャの複数のインスタンスに渡す方法の例を示します。

ソース表 "cust":

```
custid                      Number
current_residence_state     Char(2)
birth_state                 Char(2)
```

ターゲット表 "cust_extended":

```
custid                          Number
current_residence_state_long    Varchar(30)
birth_state_long                Varchar(30)
```

参照表 "state_lookup":

```
abbreviation      Char(2)
long_name         Varchar(30)
```

例 1 次に、参照表から値を取得するために 1 回実行されるストアド・プロシージャの使用例を示します。この値は、COLMAP 文のターゲット列にマッピングされます。

```
TABLE sales.cust, TARGET sales.cust_extended, &
SQLEXEC (SPNAME lookup, &
PARAMS (long_name = birth_state)), &
COLMAP (custid = custid, &
birth_state_long = lookup.long_name);
```

例 2 次に、参照表から値を取得するストアド・プロシージャを複数回実行する例を示します。値は、ターゲット列にマッピングされます。

```
TABLE sales.cust, TARGET sales.cust_extended, &
SQLEXEC (SPNAME lookup, ID lookup1, &
PARAMS (long_name = current_residence_state)), &
SQLEXEC (SPNAME lookup, ID lookup2, &
PARAMS (long_name = birth_state)), &
COLMAP (custid = custid, &
current_residence_state_long = lookup1.long_name, &
birth_state_long = lookup2.long_name);
```

AFTERFILTER および BEFOREFILTER の使用

AFTERFILTER および BEFOREFILTER では、TABLE 文の FILTER 句との関連でストアド・プロシージャまたは問合せをいつ実行するかを指定します。

構文 `AFTERFILTER | BEFOREFILTER`

ルール	説明
AFTERFILTER	FILTER 文の後に SQL 文を実行します。これにより、フィルタが成功しない場合に SQL 実行のオーバーヘッドを回避できます。これはデフォルトです。
BEFOREFILTER	FILTER 文の前に SQL 文を実行します。したがって、プロシージャまたは問合せの結果をフィルタで使用できます。

例 SQLEXEC (SPNAME check, NOPARAMS, BEFOREFILTER)

ALLPARAMS の使用

ALLPARAMS は、ストアド・プロシージャまたは問合せの実行のために、指定されたすべてのパラメータが存在している必要があるかどうかを決定するグローバル・ルールとして使用します。ALLPARAMS で設定するグローバル・ルールよりも、PARAMS 句内で設定する個別のパラメータのルールのほうが優先されます。

構文 ALLPARAMS {OPTIONAL | REQUIRED}

ルール	説明
OPTIONAL	すべてのパラメータが存在しているかどうかにかかわらず、SQL の実行を許可します。これはデフォルトです。
REQUIRED	SQL を実行するために、すべてのパラメータが存在している必要があります。

例
```
SQLEXEC (SPNAME lookup,
PARAMS (long_name = birth_state, short_name = state),
ALLPARAMS OPTIONAL)
```

DBOP の使用

DBOP では、ストアド・プロシージャまたは問合せ内で実行された INSERT、UPDATE、DELETE、および SELECT 文をコミットします。そうしないと、これらの文はロールバックされる可能性があります。Oracle GoldenGate は、ソース・トランザクションと同じトランザクション境界内でコミットを発行します。

警告 **データベース、特に本番環境のデータベースに対して SQLEXEC プロシージャを実行する場合は、注意して使用してください。プロシージャによってコミットされた変更は、既存のデータを上書きすることがあります。**

構文 DBOP

例 SQLEXEC (SPNAME check, NOPARAMS, DBOP)

ERROR の使用

ERROR では、ストアド・プロシージャまたは問合せに関連するエラーに対するレスポンスを定義します。明示的なエラー処理の指定がない場合、Oracle GoldenGate プロセスはエラーを検出すると異常終了します。プロシージャからプロセスにエラーを返させ、ERROR でレスポンスを指定するようにしてください。

構文 ERROR <action>

アクション	説明
IGNORE	Oracle GoldenGate に、ストアド・プロシージャまたは問合せに関連するすべてのエラーを無視させ、処理を継続させます。このパラメータ抽出結果は、" 列行方不明 " 状態になります。これはデフォルトです。
REPORT	ストアド・プロシージャまたは問合せに、関連するすべてのエラーを破棄ファイルにレポートさせます。DISCARDFILE パラメータで、破棄ファイルを指定しておく必要があります。このレポートは、エラーの原因の追跡に役立ちます。ここには、エラーの説明と、プロシージャまたは問合せとやり取りしたパラメータの値両方が含まれます。Oracle GoldenGate で、エラーのレポート後、処理が続けられます。
RAISE	REPERROR パラメータで設定されたルールに従ってエラーを処理します。Oracle GoldenGate は、エラーを処理する前に、現在の TABLE 文に関連する他のストアド・プロシージャまたは問合せの処理を継続します。
FINAL	RAISE と似ていますが、プロシージャまたは問合せに関連するエラーを検出したときに、残りのストアド・プロシージャおよび問合せがバイパスされます。エラー処理は、エラーの直後に起動されます。
FATAL	プロシージャまたは問合せに関連するエラーを検出したときに、即座に Oracle GoldenGate を異常終了させます。

例 `SQLEXEC (SPNAME check, NOPARAMS, ERROR REPORT)`

EXEC の使用

EXEC では、TABLE 文のストアド・プロシージャまたは問合せを実行する頻度、および出力パラメータを抽出する場合に結果を有効とみなす期間を制御します。

構文 `EXEC <frequency>`

頻度	説明
MAP	プロシージャまたは問合せが指定されている各ソース - ターゲットのマップで、プロシージャまたは問合せを 1 回実行します。MAP を使用する場合、同一のソース表を持つそれ以降のマップでは結果は無効になります。たとえば、ソース表が複数のターゲット表と同期を取る場合、結果は最初のソース - ターゲットのマップでのみ有効です。MAP はデフォルトです。
ONCE	Oracle GoldenGate の実行中、関連する TABLE 文の最初の呼び出し時にプロシージャまたは問合せを 1 回実行します。結果は、プロセスが実行しているかぎり有効です。
TRANSACTION	プロシージャまたは問合せをソース・トランザクションで 1 回実行します。結果は、トランザクションのすべての操作に有効です。

頻度	説明
SOURCEROW	プロシージャまたは問合せをソース行操作で 1 回実行します。このオプションは、ソース表を複数のターゲット表と同期し、プロシージャまたは問合せの結果がソース - ターゲット・マッピングのたびに呼び出される場合に使用します。

例 1 次に、ONCE の使用例を示します。

```
TABLE sales.cust, TARGET sales.cust_extended, &
SQLEXEC (SPNAME lookup, &
PARAMS (long_name = birth_state), &
EXEC ONCE), &
COLMAP (custid = custid,
birth_state_long = lookup.long_name);
```

例 2 次に、TRANSACTION の使用例を示します。

```
TABLE sales.cust, TARGET sales.cust_extended, &
SQLEXEC (SPNAME lookup, PARAMS (long_name = birth_state), EXEC TRANSACTION), &
COLMAP (custid = custid, &
birth_state_long = lookup.long_name);
```

例 3 次に、デフォルト (MAP) の誤った使用例を示します。2 つの TABLE 文で、同一のソース表と 2 つの異なるターゲット表を同期します。しかし、プロシージャ lookup の結果は、2 番目の MAP の実行までに無効になってしまい、2 番目の MAP は " 列行方不明 " の状態になります。この機能を正しく実装するには、SOURCEROW を使用する必要があります。

```
TABLE sales.srctab, TARGET sales.targtab, &
SQLEXEC (SPNAME lookup, PARAMS (param1 = srccol)), &
COLMAP (targcol = lookup.param2); &
TABLE sales.srctab, TARGET sales.targtab2, &
COLMAP (targcol2 = lookup.param2);
```

例 4 次に、SOURCEROW の使用例を示します。プロシージャが各ソース行操作で実行されるため、2 番目の MAP は有効な値を返します。

```
TABLE sales.srctab, TARGET sales.targtab, &
SQLEXEC (SPNAME lookup, &
PARAMS (param1 = srccol), EXEC SOURCEROW ), &
COLMAP (targcol = lookup.param2);

TABLE sales.srctab, TARGET sales.targtab2, &
COLMAP (targcol2 = lookup.param2);
```

ID の使用

ID は、TABLE 文内の問合せおよびストアド・プロシージャに対して、次のように使用します。

- 問合せでは、ID <logical name> を使用し、Oracle GoldenGate がこの名前を使用して問合せから返される列値を参照できるようにします。
- ストアド・プロシージャでは、ID <logical name> を使用して、たとえば 2 つの異なる列のマップなどのために、1 つの TABLE 文内でプロシージャを複数回呼び出せるようにします。それ以外の場合、ID は必要ありません。1 つの TABLE 文で、最大 20 のストアド・プロシージャを実行できます。これらは、パラメータ・ファイルにリストされた順に実行されます。

構文　　`ID <logical name>`

コンポーネント	説明
`<logical name>`	ストアド・プロシージャまたは問合せの論理名。たとえば、"lookup" という名前のプロシージャの論理名を "lookup1"、"lookup2" などのようにできます。

例 1　次に、ID <logical name> の使用例を示します。各列マップが、lookup1 および lookup2 を使用して lookup という名前のストアド・プロシージャを個別にコールし、それぞれの結果を参照します。

```
TABLE sales.srctab, TARGET sales.targtab, &
SQLEXEC (SPNAME lookup, ID lookup1, PARAMS (param1 = srccol)), &
COLMAP (targcol1 = lookup1.param2), &
SQLEXEC (SPNAME lookup, ID lookup2, PARAMS (param1 = srccol)), &
COLMAP, (targcol2 = lookup2.param2);
```

例 2　次に、lookup という名前のストアド・プロシージャを 1 回実行する例を示します。このケースでは、実際のプロシージャ名を使用します。論理名は必要ありません。

```
TABLE sales.tab1, TARGET sales.tab2, &
SQLEXEC (SPNAME lookup), PARAMS (param1 = srccol)), &
COLMAP (targcol = lookup.param1);
```

例 3　次に、Oracle および SQL Server の問合せにそれぞれ ID <logical name> を使用する例を示します。この例では、このドキュメントのスペースの制約によって、SQLEXEC 文が複数の行にまたがっています。実際の SQLEXEC 文は、1 行内に含める必要があります。

```
TABLE sales.account, TARGET sales.newacct,
SQLEXEC (ID lookup,
QUERY "select desc_col into desc_param from lookup_table
" where code_col = :code_param ",
PARAMS (code_param = account_code)),
COLMAP (newacct_id = account_id,
newacct_val = lookup.desc_param);

TABLE sales.account, TARGET sales.newacct,
SQLEXEC (ID lookup,
QUERY "select desc_col into desc_param from lookup_table
where code_col = ?",
PARAMS (p1 = account_code)),
COLMAP (newacct_id = account_id,
newacct_val = lookup.desc_param);
```

MAXVARCHARLEN の使用

MAXVARCHARLEN では、プロシージャまたは問合せの出力パラメータに割り当てる最大長を指定します。これを超える出力値は切り捨てられます。

構文　　`MAXVARCHARLEN <num bytes>`

コンポーネント	説明
(<num bytes>)	出力パラメータに許可する最大バイト数を定義します。明示的に MAXVARCHARLEN 句を使用しない場合のデフォルトは、255 バイトです。

例 MAXVARCHARLEN 100

NOPARAMS の使用

NOPARAMS は、ストアド・プロシージャまたは問合せがパラメータを必要としない場合に、PARAMS のかわりに使用します。PARAMS 句または NOPARAMS のいずれかを指定する必要があります。

構文 NOPARAMS

例 SQLEXEC (SPNAME check, NOPARAMS)

PARAMBUFSIZE の使用

PARAMBUFSIZE では、入力および出力パラメータを含むパラメータ情報を保持するメモリー・バッファの最大値を指定します。Oracle GoldenGate は、パラメータに割り当てられたメモリーと最大値との差が 500 バイト以内に達するたびに警告を発行します。

構文 PARAMBUFSIZE <num bytes>

コンポーネント	説明
<num bytes>	メモリー・バッファに許可する最大バイト数を定義します。明示的に PARAMBUFSIZE 句を使用しない場合のデフォルトは、10,000 バイトです。

例 PARAMBUFSIZE 15000

PARAMS の使用

PARAMS では、入力を受け付けるストアド・プロシージャまたは問合せのパラメータ名と、入力を提供するソース列名または Oracle GoldenGate 列変換ファンクション名を指定します。PARAMS 句または NOPARAMS のいずれかを指定する必要があります。

デフォルトでは、Oracle GoldenGate は二重引用符で囲まれた文字列をリテラルとして処理します。列名に二重引用符、およびリテラルに一重引用符 (SQL-92 ルール) を使用するには、GLOBALS パラメータ・ファイルで USEANSISQLQUOTES パラメータを使用します。

次に、SQLEXEC によってサポートされているデータベースと、入力および出力パラメータとしてサポートされているデータ型を示します。

- 数値データ型
- 日付データ型
- 文字データ型

デフォルトでは、出力パラメータは、パラメータ当たり 255 バイトで切り捨てられます。プロシージャがこれより長いパラメータを必要とする場合は、MAXVARCHARLEN オプションを使用します。

構文

```
PARAMS (
[OPTIONAL | REQUIRED]
<param name> = {<source column> | <GG function>}
[, ...]
)
```

コンポーネント	説明
`OPTIONAL | REQUIRED`	パラメータ値が見つからないときに、プロシージャまたは問合せを実行するかどうかを決定します。 ◆ OPTIONAL は、SQL の実行にパラメータ値が必要ないことを示します。必要なソース列がデータベース操作で見つからない場合、またはソース列が見つからないために列変換ファンクションを完了できない場合でも、SQL は実行されます。 OPTIONAL は、Oracle 以外のすべてのデータベースのデフォルトです。Oracle では、ストアド・プロシージャ定義を取得するときに、パラメータがオプションかどうかが自動的に判別されます。 ◆ REQUIRED は、パラメータ値が存在している必要があることを示します。パラメータ値が存在しない場合、SQL は実行されません。
`<param name> = { <source column> | <GG function }`	パラメータ名を、入力を提供する列またはファンクションにマップします。 条件 : ◆ <param name> は、次のいずれかです。 ストアド・プロシージャの場合、ストアド・プロシージャ内の入力を受け付ける任意のパラメータ名 (参照表の列など) です。 Oracle 問合せの場合、先行するコロンを*除いた*、問合せの任意の入力パラメータ名です。たとえば、:param1 の場合は、PARAMS 句では `param1` と指定します。 Oracle 以外の問合せの場合は *Pn* (*n* は 1 から始まるパラメータの番号) です。たとえば、パラメータが 2 個の問合せでは、<param name> エントリは P1 および P2 です。 ◆ <source column> は、ソース列名です。デフォルトでは、指定した列が (圧縮更新であるために) ログに存在しない場合、このパラメータはプロシージャまたは問合せでこのパラメータに指定されているデフォルト値を使用します。 ◆ <GG function> は、Oracle GoldenGate 列変換ファンクション名です。列変換ファンクションの詳細は、第 5 章を参照してください。

例

次に、account 表からターゲット表 newacct にデータをマップする例を示します。account 表からのレコードを処理するとき、Oracle GoldenGate は、列マップを実行する前に lookup ストアド・プロシージャを実行します。プロシージャの code_param パラメータは、account_code ソース列から入力を受け付けます。

```
TABLE sales.account, TARGET sales.newacct, &
SQLEXEC (SPNAME lookup, PARAMS (code_param = account_code)), &
COLMAP (newacct_id = account_id, &
newacct_val = lookup.desc_param);
```

TRACE の使用

TRACE では、入力および出力パラメータをレポート・ファイルに記録します。

次に、SQLEXEC トレースを有効化している場合の破棄ファイルの例を示します。

```
Input parameter values...

LMS_TABLE: INTERACTION_ATTR_VALUES
   KEY1: 2818249
  KEY2: 1
Report File:

From Table MASTER.INTERACTION_ATTR_VALUES to MASTER.INTERACTION_ATTR_VALUES:
       #  inserts:      0
       #       updates:         0
       #       deletes:         0
       #  discards:     1

  Stored procedure GGS_INTERACTION_ATTR_VALUES:
       attempts:        2
       successful:      0
```

構文

```
TRACE {ALL | ERROR}
```

アクション	説明
ALL	呼び出された各プロシージャまたは問合せの入力および出力パラメータをレポート・ファイルに書き込みます。これはデフォルトです。
ERROR	SQL エラーの発生後にのみ、呼び出された各プロシージャまたは問合せの入力および出力パラメータをレポート・ファイルに書き込みます。

例

```
SQLEXEC (SPNAME lookup, PARAMS (long_name = birth_state, short_name = state), TRACE
ERROR)
```

SQLPREDICATE の使用

SQLPREDICATE では、Extract が初期ロードの準備で表からデータを選択するときに使用する SELECT 文に、通常の SQL WHERE 句を含めます。SQLPREDICATE では、選択条件で返されるレコードがキー値によって順序付けされます。

SQLPREDICATE は、WHERE または FILTER オプションよりも初期ロードに適した選択方法です。これは SQL 文を直接適用し、他のオプションのように Oracle GoldenGate がフィルタリング前にすべてのレコードをフェッチする必要がないため、はるかに処理が高速です。

Oracle 表の場合、SQLPREDICATE を使用すると UNDO セグメントに保持されるデータ量が削減され、スナップショットが古すぎるために発生するエラーが低減します。非常に大きな表をロードするときに有益です。

SQLPREDICATE 句を使用することにより、複数の Extract プロセス間で大きな表の行をパーティション化できます。この構成では、並列配信ロード処理も活用できます。

SQLPREDICATE では、タイムスタンプやその他の基準に基づいてデータを選択し、抽出してターゲット表にロードする行を制限するためにも使用できます。また SQLPREDICATE では、ORDER BY 句やその他のタイプの選択句も使用できます。

最適なパフォーマンスを得るためには、WHERE 句の一部として指定する列がキーまたは索引の一部である必要があります。そうでない場合は、完全な表スキャンが必要になるため、SELECT 文の効率が低下します。

このパラメータは、Oracle、DB2 LUW および z/OS、SQL Server、Teradata データベースに有効です。表から直接レコードを選択する SELECT 文が使用されるとみなされるため、変更データ同期ではなく、初期ロード・プロセスでのみこれを使用してください。データ・ポンプ Extract グループにパススルー・モードで処理される表には、このオプションを使用しないでください。

構文

```
TABLE <table spec>, SQLPREDICATE "WHERE <where clause>"
```

コンポーネント	説明
WHERE	必須のキーワード。
<where clause>	有効な SQL WHERE 句。

例

```
SQLPREDICATE "where state = 'CO' and city = 'DENVER'"
```

TARGETDEF の使用

TARGETDEF では、ターゲット定義テンプレートを指定します。定義テンプレートは、特定のターゲット表に対して DEFGEN が実行されるときに、このオブジェクトの定義に基づいて作成されます。テンプレートが作成されると、この表と同一の定義を持つ新しいターゲット表は、DEFGEN を実行せずに、かつ Extract の停止と起動を伴わずに追加できます。TARGETDEF で指定されたテンプレートの定義は、定義の参照に使用されます。DEFGEN の詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。

構文

```
TABLE <table spec>, TARGETDEF <definitions template>;
```

引数	説明
<definitions template>	DEFGEN パラメータ・ファイルの TABLE の DEF オプションで指定されている定義テンプレート名。テンプレートに含まれる定義は、この TABLE 文の表の定義と同一である必要があります。

例

```
TABLE acct.cust*, TARGET acc.cust*, DEF custdef, TARGETDEF tcustdef;
```

TOKENS の使用

TOKENS では、ユーザー・トークンを定義し、データと関連付けます。トークンを使用すると、データを抽出し、トレイル・レコード・ヘッダーのユーザー・トークン・エリア内に保持できます。トークン・データを様々な方法で取得および使用することにより、Oracle GoldenGate のデータ提供方法をカスタマイズできます。たとえば、列マップ、SQLEXEC で呼び出されるストアド・プロシージャ、またはマクロでトークン・データを使用できます。

ターゲット表で定義済トークン・データを使用するには、Replicat 用 MAP 文の COLMAP 句で、@TOKEN 列変換ファンクションを使用します。@TOKEN ファンクションは、トークン名をターゲット列にマップします。

データ・ポンプ Extract グループにパススルー・モードで処理される表には、このオプションを使用しないでください。

トークン・データの文字セットは変換されません。トークンは、Extract 用ソース・データベースの文字セットおよび Replicat 用ターゲット・データベースの文字セットである必要があります。

ターゲット表が EBCDIC ではない場合、z/OS システム上の EBCDIC としてエンコードされているソース表に、このオプションを使用しないでください。

トークンの使用の詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。

構文

```
TABLE <table spec>, TOKENS (<token name> = <token data> [, ...]) ;
```

コンポーネント	説明
<token name>	トークンに付ける名前。任意の数の有効な文字を使用できます (大 / 小文字区別なし)。マルチバイトの名前はサポートされていません。
<token data>	2000 バイトまでの有効な文字列。データは、二重引用符 (USEANSISQLQUOTES を使用している場合は一重引用符) で囲まれたリテラル、または Oracle GoldenGate 列変換ファンクションの結果のいずれかを指定できます。 USEANSISQLQUOTES の詳細は、416 ページを参照してください。

例

次に、TK-OSUSER、TK-GROUP および TK-HOST という名前のトークンを作成し、これらのトークンを @GETENV ファンクションで取得したトークン・データにマップする例を示します。

```
TABLE ora.oratest, TOKENS (
TK-OSUSER = @GETENV ("GGENVIRONMENT" , "OSUSERNAME"),
TK-GROUP = @GETENV ("GGENVIRONMENT" , "GROUPNAME")
TK-HOST =  @GETENV ("GGENVIRONMENT" , "HOSTNAME"));
```

TRIMSPACES および NOTRIMSPACES の使用

TRIMSPACES および NOTRIMSPACES では、ソースの CHAR 列の末尾の空白を、ターゲットの CHAR または VARCHAR 列に適用するときに切り捨てるかどうかを制御します。デフォルトは TRIMSPACES です。

注意 Sybase は、すべての CHAR 型を VARCHAR 型として処理し、このため TRIMSPACES は無効になります。Sybase では、TRIMVARSPACES パラメータを使用します。

TRIMSPACES は、シングルバイトの空白 (U+0020) にのみ適用されます。表意空白 (U+3000) はサポートされていません。

TRIMSPACES および NOTRIMSPACES は、異なる TABLE 文または文グループに対する切捨て機能を有効化または無効化するために、パラメータ・ファイルのルート・レベルでも使用できます。

Extract では、Extract が (TARGET 文を使用して)TABLE 文内でマッピングを実行している場合、TRIMSPACES のみが有効になります。

構文

```
TABLE <table spec>, {TRIMSPACES | NOTRIMSPACES};
```

例

次に、最初の 2 つの表に対してはデフォルトの末尾空白の切捨てを行い、最後の 2 つでは Extract が末尾空白の切捨てを行わない例を示します。

```
TABLE fin.src1;
TABLE fin.src2;
TABLE fin.src3, NOTRIMSPACES;
TABLE fin.src4, NOTRIMSPACES;
```

TRIMVARSPACES および NOTRIMVARSPACES の使用

TRIMVARSPACES および NOTRIMVARSPACES では、ソースの VARCHAR 列の末尾の空白を、ターゲットの CHAR または VARCHAR 列に適用するときに切り捨てるかどうかを制御します。

VARCHAR 列の空白は実際はデータの一部であるため、デフォルトは NOTRIMVARSPACES です。TRIMVARSPACES を使用する前に、末尾の空白がターゲット・データにとって不可欠ではないことを確認してください。

TRIMSPACES は、シングルバイトの空白 (U+0020) にのみ適用されます。表意空白 (U+3000) はサポートされていません。

TRIMVARSPACES および NOTRIMVARSPACES は、異なる TABLE 文または文グループに対する切捨て機能を有効化または無効化するために、パラメータ・ファイルのルート・レベルでも使用できます。

Extractでは、Extractが(TARGET文を使用して)TABLE文内でマッピングを実行している場合、TRIMVARSPACESのみが有効になります。

構文

```
TABLE <table spec>, {TRIMVARSPACES | NOTRIMVARSPACES};
```

例

次に、最初の 2 つの表に対してはデフォルトの末尾空白の切捨てを行わず、最後の 2 つでは末尾空白の切捨てを行う例を示します。

```
TABLE fin.src1;
TABLE fin.src2;
TABLE fin.src3, TRIMVARSPACES;
TABLE fin.src4, TRIMVARSPACES;
```

WHERE の使用

WHERE では、条件文に基づいてレコードを選択します。データ・ポンプ Extract グループにパススルー・モードで処理される表には、このオプションを使用しないでください。

Oracle GoldenGate では、マルチバイト文字セットまたはローカル・オペレーティング・システムの文字セットと互換性のない文字セットを含む列に対して、WHERE をサポートしていません。

GLOBALS ファイルで USEANSISQLQUOTES パラメータが使用されていない場合、WHERE 内で使用するリテラル文字列は二重引用符で囲む必要があります。このパラメータは、識別子およびリテラルに SQL-92 ルールを施行します。

WHERE 句の使用および使用可能な列データの詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。

構文

```
TABLE <table spec>, WHERE (<where clause>);
```

コンポーネント	説明
<where clause>	次の例のように、条件に基づいてレコードを選択します。 WHERE (branch = "NY") 次の表に、許可されている WHERE 演算子を示します。 WHERE は、主キー更新操作の一部として条件文の主キー列のビフォア・イメージの評価をサポートしていません。

表 41　許可されている WHERE 演算子

演算子	例：
列名	PRODUCT_AMT
数値	-123, 5500.123
引用符で囲まれたリテラル文字列	"AUTO"、"Ca"
列テスト	@NULL、@PRESENT、@ABSENT(レコードの列が NULL、存在、不在かのテスト)。これらのテストは、Oracle GoldenGate に組み込まれています。
比較演算子	=, <>, >, <, >=, <=
結合演算子	AND、OR
グループ化用カッコ	複数の要素を論理的にグループ化するには、開きおよび閉じカッコを使用します。

例

次の WHERE の例では、AMOUNT 列が 10,000 を超えるとすべてのレコードを返し、AMOUNT が存在しないとレコードは破棄されません。

```
WHERE (amount = @PRESENT AND amount > 10000)
```

Replicat 用 TABLE

Replicat パラメータ・ファイルの TABLE パラメータでは、EVENTACTIONS で指定されているイベント・アクションの対象にするデータ・レコードをトレイルから限定するフィルタリング・ルールを指定します。

警告　**EVENTACTIONS は、ソース・データベースが Teradata で、Extract が最大パフォーマンス・モードで構成されている場合にはサポートされません。**

Replicat 用 TABLE 文は、Replicat 用 MAP 文に似ていますが、この文では TARGET 句によってデータ・レコードのソース表からターゲット表へのマッピングを行いません。Replicat 用の TABLE は、イベント・

レコードを検出したときに Replicat が実行するデータ操作以外のアクションをトリガーする手段としてのみ使用します。

ターゲット表が指定されないため、次のような制限があります。

- Replicat が表名または列をターゲット表にマップするためのオプションや、Replicat がデータを操作するためのオプションは使用できません。
- ASSUMETARGETDEFS を使用するには、Replicat が表定義を問い合せるターゲット表名が必要になるため、Replicat 用 TABLE 文が含まれていると同じパラメータ・ファイルでは ASSUMETARGETDEFS パラメータを使用できません。Replicat にソース表の定義を提供するために、ソース定義ファイルを作成する必要があります。このファイルをターゲット・システムに送信し、Replciat パラメータ・ファイルで SOURCEDEFS パラメータを使用してファイルのパス名を指定します。
- イベント・レコード自体は、Replicat によってターゲット・データベースに適用されません。EVENTACTIONS オプションの IGNORE または DISCARD を指定する必要があります。

TABLE 文は、セミコロンを使用して終了します。

次の内容については、Replicat 用 MAP を参照してください。

- 表名でサポートされている文字
- EVENTACTIONS 構文オプションの説明

構文

```
TABLE <table spec>,
[, SQLEXEC (<SQL specification>), BEFOREFILTER]
[, FILTER (<filter specification>)]
[, WHERE (<where clause>)]
{, EVENTACTIONS ({IGNORE | DISCARD} [<action>])}
;
```

例

次に、特定の受注番号 (order_no = 1) に対する挿入操作が含まれる受注トランザクションを Replicat にトレースさせる例を示します。トレース情報は、order_1.trc トレース・ファイルに書き込まれます。MAP パラメータでは、ソース表とターゲット表のマッピングを指定します。

```
MAP sales.order, TARGET rpt.order;

TABLE sales.order,
FILTER (@GETENV ("GGHEADER", "OPTYPE") = "INSERT" AND order_no = 1), &
EVENTACTIONS (TRACE order_1.trc TRANSACTION);
```

イベント・マーカー・システムの他の使用例および詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。

TABLEEXCLUDE

適用対象 Extract

TABLEEXCLUDE パラメータは、TABLE および SEQUENCE パラメータとともに使用して、ワイルドカード指定から明示的に表と順序を除外します。TABLEEXCLUDE は、除外するオブジェクトを含むすべての TABLE および SEQUENCE 文よりも先に指定する必要があります。

デフォルト なし

構文 TABLEEXCLUDE <exclude specification>

引数	説明
<exclude specification>	除外するオブジェクトの名前またはワイルドカード指定。TABLE または SEQUENCE 文でオブジェクトをワイルドカード指定するときと同じワイルドカード・ルールが TABLEEXCLUDE に適用されます。

例 次の例では、TABLE 文は TEST という名前の表を除くすべての表を取得します。

```
TABLEEXCLUDE fin.TEST
TABLE fin.*;
```

TARGETDB

適用対象 Replicat

TARGETDB は、接続情報の一部としてデータ・ソース名を必要とするデータベース用のパラメータです。TARGETDB エントリに続く MAP 文に指定されているターゲット表は、指定したデータベースのものであるとみなされます。

このパラメータは、データベースへのログインで要求される認証方法に応じて、次に示すように USERID パラメータとともに使用する必要がある場合があります。

- データベース・ログイン情報が要求されるデータベースの場合は、同じパラメータ文内で (必要な場合) TARGETDB を USERID とともに使用する必要があります。
- オペレーティングシステム・レベルでの認証を許可するデータベースの場合は、USERID なしで TARGETDB を指定できます。

デフォルト なし

構文 TARGETDB <data source>[, SESSIONCHARSET <character set>]

引数	説明
<data source>	データ・ソース名。
SESSIONCHARSET <character set>	Sybase、Teradata および MySQL をサポートします。プロセス・ログイン・セッションに対するデータベース・セッションの文字セットを設定します。このパラメータは、GLOBALS ファイルで指定されるすべての SESSIONCHARSET より優先されます。

例 1 次に、USERID パラメータがある場合とない場合の TARGETDB の例を示します。

```
TARGETDB mydb

TARGETDB mydb, USERID ggs, &
    PASSWORD AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC &
    AES128, ENCRYPTKEY securekey1
```

例 2 次に、ユーザー・セッションの文字セットを設定する例を示します。

```
TARGETDB mydb, USERID ggs, &
    PASSWORD AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC &
    AES128, ENCRYPTKEY securekey1, SESSIONCHARSET ISO-8859-11
```

TARGETDEFS

適用対象 Extract(プライマリおよびデータ・ポンプ)

TARGETDEFS パラメータは、ターゲットが Enscribe ファイルの場合に使用します。TARGETDEFS には、ターゲット・システム上に存在する表およびファイルのデータ定義を含む、ソース・システムまたは仲介システム上のファイル名を指定します。ターゲットとして Enscribe ファイルが指定されている TABLE 文の前に、少なくとも 1 つの TARGETDEFS エントリを指定します。

ターゲット定義ファイルを生成するには、DEFGEN ユーティリティを使用します。Extract を起動する前に、このファイルをソース・システムまたは仲介システムに送信してください。

たとえば、各 TARGETDEFS ファイルに別々のアプリケーションの定義が保持されている場合など、異なる定義を使用するために複数のターゲット定義ファイルが必要なときは、パラメータ・ファイルで複数の TARGETDEFS 文を使用できます。

デフォルト なし

構文

```
TARGETDEFS <file name>
```

引数	説明
<file name>	データ定義を取得するファイルの相対パス名または完全修飾パス名。

例 1

```
TARGETDEFS C:\repodbc\sales.def
```

例 2

```
TARGETDEFS /ggs/dirdef/ODBC/tandem_defs
```

TCPSOURCETIMER | NOTCPSOURCETIMER

適用対象 Extract

TCPSOURCETIMER および NOTCPSOURCETIMER パラメータでは、レポートでの使用を目的として、Oracle GoldenGate 環境内にレプリケートされた操作のタイムスタンプを管理します。

デフォルトは TCPSOURCETIMER です。これにより、データ・レコードが他のシステムに送信されるときにタイムスタンプが調整されるため、同期ラグの理解が容易になります。

NOTCPSOURCETIMER では、元のタイムスタンプ値を保持します。NOTCPSOURCETIMER は、双方向構成でタイムスタンプベースでの競合解決を使用するときに使用します。NOTCPSOURCETIMER は、@GETENV 列変換ファンクションの "GGHEADER"、"COMMITTIMESTAMP" を参照するユーザー・トークンを使用するときに使用します。

TCPSOURCETIMER および NOTCPSOURCETIMER はグローバル・パラメータで、Extract パラメータ・ファイル内のすべての TABLE 文に適用されます。

デフォルト TCPSOURCETIMER

構文

```
TCPSOURCETIMER | NOTCPSOURCETIMER
```

THREADOPTIONS

適用対象 Extract

THREADOPTIONS パラメータでは、スレッド Extract の動作方法を制御します。

パフォーマンス・オプションの使用

Oracle GoldenGate は、トレイルに送信する前に、データをメモリーのキューに入れます。THREADOPTIONS パラメータの INQUEUESIZE および OUTQUEUESIZE オプションでは、キューに入れるデータ量を決定します。この値が大きいほど、大量のデータを処理するときのパフォーマンスが向上します。この値を小さくすると、アクティビティが非常に少ない環境でデータがターゲットにより迅速に送信されます。デフォルト値から使用を開始してください。一般的な値は、100 から 1500 までです。ほとんどの環境では、各オプションとも 1000 で十分なパフォーマンスが得られるはずです。

この 2 つのパラメータに加え、AIX ユーザーは、環境変数 AIXTHREAD_SCOPE を S (システム・スコープ) に設定して、プロセスの同時実行をサポートする複数 CPU の使用を指定することにより、パフォーマンスを改善できることがあります。システム範囲を使用するには、Manager プロセスを起動するユーザーの .profile ファイルに次を追加するか、GGSCI を起動する前に、この変数を手動でエクスポートします。

```
AIXTHREAD_SCOPE=S
export AIXTHREAD_SCOPE
```

GGSCI、Manager、および Extract を停止および再起動して、変更を有効にします。

デフォルト なし

構文

```
THREADOPTIONS
[EOFDELAYMS <milliseconds>]
[IOLATENCY <milliseconds>]
[INQUEUESIZE <n>]
[MAXCOMMITPROPAGATIONDELAY <milliseconds>]
[OUTQUEUESIZE <n>]
```

引数	説明
`EOFDELAYMS <milliseconds>`	REDO ログの論理的な末尾に到達した後に、Extract プロセスが処理する追加データを探すまでの遅延時間を指定します。デフォルトは 250 ミリ秒です。この値を増やすと、ソース表に変更が行われた時間とターゲット表に変更が適用される時間のラグが増える可能性があります。
`INQUEUESIZE <n>`	Oracle RAC クラスタ内の各プロデューサ Extract スレッドの入力キューのキュー・エントリ数を指定します。有効な値は 16 から 65535 までです。デフォルト値は 128 です。ほとんどの場合、デフォルトで十分です。

引数	説明
IOLATENCY <milliseconds>	データベースで構成されている最大コミット伝播遅延と、Oracle GoldenGate によって使用される内部値との時間差を指定します。有効な値は、0 ～ 180000 ミリ秒 (3 分) です。内部 I/O 遅延を表すデフォルト値は、1500(1.5 秒) です。 Extract がアーカイブ・ログ専用 (ALO) モードの場合は無効です。
MAXCOMMITPROPAGATIONDELAY <milliseconds>	このオプションは、Oracle 11.2 より前の Oracle リリースに有効です。対応する Oracle MAX_COMMIT_PROPAGATION_DELAY パラメータは、Oracle 11.2 で廃止されました。 MAX_COMMIT_PROPAGATION_DELAY は、トランザクションがコミットされた時間から、アイドルの REDO ログを読み取る時間までの遅延時間を指定します。この遅延を指定することにより、Oracle が REDO ログにデータを書き込む時点から、Oracle GoldenGate が REDO ログからデータを読み取るまでの時間差を考慮できます。共有 RAC データベース・ドライブへのアクセスに競合がある場合、この時間差は大きくなることがあります。キャッシングおよびシリアライズ機能を持ち、このような遅延を最小化または排除する SAN および NFS デバイス上にデータベースが存在する場合は、低い値が適切です。 この値は、0 より大きく、90000 ミリ秒 (90 秒) 未満に設定する必要があります。デフォルトは 3 秒です。この値は、Oracle の同じ名前のパラメータの最小値である 2000 ミリ秒よりも常に大きい値に設定し、また Oracle 値よりも絶対に低い値に設定しないでください。Oracle の値を確認するには、DBA 権限を持つユーザーとして接続し、SQL*Plus で次のコマンドを発行します。 `show parameter max_commit` Extract がアーカイブ・ログ専用 (ALO) モードの場合は無効です。
OUTQUEUESIZE <n>	Extract プロデューサ・スレッドの出力キューのキュー・エントリ数を指定します。有効な値は 8 から 65535 までです。デフォルト値は 2048 です。ほとんどの場合、デフォルトで十分です。

TRACE | TRACE2

適用対象 Extract および Replicat

TRACE および TRACE2 パラメータでは、処理のボトルネックの把握に役立つ、Extract または Replicat の処理情報を取得します。

- TRACE は、ステップバイステップの処理情報を提供します。
- TRACE2 では、Extract または Replicat がほどんどの時間を費やしているコード・セグメントを特定します。

両パラメータとも、DML および DDL のトレースをサポートしています。

またトレースは、GGSCI の SEND EXTRACT または SEND REPLICAT コマンドを使用して、有効化と無効化を切り替えることもできます。GGSCI コマンドの詳細は、第 1 章を参照してください。

トレースによって重大な処理のボトルネックが明らかになった場合は、Oracle サポートに連絡してください。詳細は、http://support.oracle.com を参照してください。

デフォルト トレースしない

構文

```
TRACE | TRACE2
[, DDL[INCLUDE] | DDLONLY]
[, [FILE] <file name>]
```

引数	説明
`DDL[INCLUDE] | DDLONLY`	(Replicat のみ) DDL トレースを有効化し、トレース・レポートに DDL トレースを取得する方法を指定します。 ◆ DDL[INCLUDE] では、DDL のトレースに加え、トランザクション・データ処理もトレースします。DDL または DDLINCLUDE のいずれかが有効です。 ◆ DDLONLY では、DDL をトレースし、トランザクション・データはトレースしません。

`[FILE] <file name>`	Oracle GoldenGate のトレース情報の記録先ファイルの相対名または完全修飾名。FILE キーワードはオプションですが、次の例のように、ファイル名の後に他のパラメータ・オプションを指定する場合は使用する必要があります。
	`TRACE FILE <file name> DDLINCLUDE`
	次の例のように、ファイル名の後に他のオプションを指定しない場合は、FILE キーワードを省略できます。
	`TRACE DDLINCLUDE <file name>`

例

```
TRACE /home/ggs/dirrpt/trace.trc
```

TRACETABLE | NOTRACETABLE

適用対象 Extract および Replicat

TRACETABLE および NOTRACETABLE パラメータは、Oracle データベースとともに使用し、ADD TRACETABLE コマンドで作成されたトレース表を特定します。TRACETABLE は、トレース表がデフォルトの GGS_TRACE 以外の名前で作成されている場合にのみ必要です。GGS_TRACE という名前のトレース表がデータベースに存在する場合は、トレース表機能は自動的に有効になり、TRACETABLE の指定は必要ありません。

トレース表は、双方向同期で Extract に Replicat トランザクションを識別させるために使用します。

TRACETABLE を使用する場合、このパラメータは Extract および Replicat 両方のパラメータ・ファイルに指定する必要があります。

- Replicat パラメータ・ファイルの TRACETABLE は、各トランザクションの開始時に、Replicat に操作をトレース表に書き込ませます。
- Extract パラメータ・ファイルの TRACETABLE では、Extract に、トレース表に含まれる操作で開始されるすべてのトランザクションを Replicat トランザクションとして識別させます。

NOTRACETABLE を指定すると、Replicat はトレース表に操作を書き込まないため、Extract は Replicat トランザクションを認識できません。

Replicat トランザクションを Extract に抽出させるか無視させるかを制御するには、GETREPLICATES および IGNOREREPLICATES パラメータを使用します。213 ページを参照してください。

双方向同期の構成方法は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。

デフォルト GGS_TRACE

構文 TRACETABLE [<owner>.]<table name> | NOTRACETABLE

引数	説明
[<owner>.]<table name>	トレース表の所有者 (オプション) および名前。所有者を指定しない場合は、USERID パラメータで指定されているユーザーのスキーマに所有されているとみなされます。

例 TRACETABLE ggs.excl_trans

TRAILCHARSET

適用対象 Replicat

TRAILCHARSET パラメータは、次の用途で使用します。

- トレイル・ヘッダーに記録されているソース文字セットより優先させるため。これは、ソース文字セットがトレイル・ヘッダーに格納される、バージョン 11.2.1.0.0 以上の Extract によって書き込まれるトレイルに適用されます。
- トレイルが、11.2.1.0.0 より前のバージョンの Extract によって書き込まれる場合に、ソース・データの文字セットを指定するため。以前のバージョンでは、ソース文字セットはトレイルに格納されていません。

TRAILCHARSET が使用されている場合、文字型の列をターゲットの文字セットに変換する際、Replicat は、指定された文字セットをソース文字セットとして使用します。TRAILCHARSET の文字セットを使用する際、Replicat は警告メッセージを表示します。

デフォルトでは、Replicat は文字セットの変換を行います。この機能は、CHARSETCONVERSION パラメータ (デフォルト) および NOCHARSETCONVERSION パラメータで制御されます。TRAILCHARSET を使用する場合、NOCHARSETCONVERSION は使用できません。

デフォルト オペレーティング・システムの文字セット

構文 `TRAILCHARSET <source_charset> [, REPLACEBADCHAR];`

引数	説明
`<source_charset>`	ソース・データベースの ICU 文字セット識別子または Oracle 文字セット識別子 Oracle データベースでは、Replicat パラメータ・ファイルの SETENV パラメータの NLS_LANG に指定された文字セットに変換する場合、Oracle GoldenGate は Oracle 識別子を対応する ICU 識別子に変換します。
REPLACEBADCHAR	変換が失敗した場合に Replicat が異常終了しないようにします。失敗した文字は、各ターゲット文字セットの置換文字に置換されます。置換文字は、文字セットごとにあらかじめ定義されています。

例 1 `TRAILCHARSET ISO-8859-9;`

例 2 `TRAILCHARSET windows-932, REPLACEBADCHAR;`

例 3 `TRAILCAHRSET EUC-CN;`

TRAILCHARSETASCII

適用対象 z/OS 上の DB2 の Extract

TRAILCHARSETASCII を使用すると、Extract が実行されているジョブのローカル ASCII コードページで文字データがトレイル・ファイルに書き込まれます。

- シングルバイト DB2 z/OS サブシステムでこのパラメータを指定すると、Unicode 以外の表からの文字データは、インストールされている ASCII シングルバイト CCSID でトレイル・ファイルに書き込まれます。EBCDIC の表からのデータは、この ASCII CCSID に変換されます。
- マルチバイト DB2 z/OS サブシステムでこのパラメータを指定すると、Extract は ASCII および Unicode の表のみ処理します。EBCDIC の表があると、Extract はエラーで異常終了します。ASCII の表からのデータは、インストールされている ASCII 混合 CCSID でトレイル・ファイルに書き込まれます。

ターゲットがマルチバイト・システムの場合、TRAILCHARSETASCII または TRAILCHARSETEBCDIC のいずれかが必要です。ASCII と EBCDIC の両方の表をマルチバイト DB2 z/OS ターゲットにレプリケートするには、各文字セットを別個の Oracle GoldenGate 処理ストリーム (ASCII 表用の Extract、データ・ポンプおよび Replicat と、EBCDIC 表用の別の Extract、データ・ポンプおよび Replicat) で処理します。

デフォルト 文字データは、ホスト表の文字セットで書き込まれます。

構文 `TRAILCHARSETASCII`

TRAILCHARSETEBCDIC

適用対象 z/OS 上の DB2 および DB2 for i の Extract

TRAILCHARSETEBCDIC を使用すると、Extract が実行されているジョブのローカル EBCDIC コードページで文字データがトレイル・ファイルに書き込まれます。

- このパラメータを指定すると、Extract が実行されているジョブの EBCDIC コードページですべての文字データがトレイル・ファイルに書き込まれます。
- シングルバイト DB2 z/OS サブシステムでこのパラメータを指定すると、Unicode 以外の表からの文字データは、インストールされている EBCDIC シングルバイト CCSID でトレイル・ファイルに書き込まれます。ASCII の表からのデータは、この EBCDIC CCSID に変換されます。
- マルチバイト DB2 z/OS サブシステムでこのパラメータを指定すると、Extract は EBCDIC および Unicode の表のみ処理します。ASCII の表があると、Extract はエラーで異常終了します。EBCDIC の表からのデータは、インストールされている EBCDIC 混合 CCSID でトレイル・ファイルに書き込まれます。

ターゲットがマルチバイト・システムの場合、TRAILCHARSETASCII または TRAILCHARSETEBCDIC のいずれかが必要です。ASCII と EBCDIC の両方の表をマルチバイト DB2 z/OS ターゲットにレプリケートするには、各文字セットを別個の Oracle GoldenGate 処理ストリーム (ASCII 表用の Extract、データ・ポンプおよび Replicat と、EBCDIC 表用の別の Extract、データ・ポンプおよび Replicat) で処理します。

デフォルト DB2 for i: 文字データは UTF-8 で書き込まれます。z/OS 上の DB2: 文字データはホスト表の文字セットで書き込まれます。

構文 `TRAILCHARSETEBCDIC`

TRAILCHARSETUTF8

適用対象 DB2 for i の Extract

TRAILCHARSETUTF8 を使用すると、Extract は、すべての文字 (グラフィック以外) データを UTF-8 でトレイルに書き込みます。Extract は必要に応じて変換を行います。グラフィック・データは UTF-16 で書き込まれます。

デフォルト 有効

構文 `TRAILCHARSETUTF8`

TRANLOGOPTIONS

適用対象 Extract

TRANLOGOPTIONS パラメータでは、Extract と、トランザクション・ログまたはトランザクション・データを渡す API(データベースやキャプチャ・モードによって異なる) とのやり取り方法を制御します。TRANLOGOPTIONS 文を同一のパラメータ・ファイル内で複数回使用することも、(オプションに許可されている場合は) 同一の TRANLOGOPTIONS 文内で複数のオプションを指定することもできます。

各 TRANLOGOPTIONS オプションは、対象のデータベースに対してのみ使用してください。

デフォルト なし

構文

```
TRANLOGOPTIONS {
[ALTARCHIVEDLOGFORMAT <string>] [INSTANCE <instance_name>] [THREADID <id>]
[ALTARCHIVELOGDEST [PRIMARY] [INSTANCE <instance_name>] <path name>] [ALTARCHIVELOGDEST
("<Backup Path>" [FILESPEC "<File Pattern>"]
    [[NOT] RECURSIVE] [PRIMARY])]
[ALTLOGDEST <path>]
[ARCHIVEDLOGONLY]
{[ASMBUFSIZE <size>] | [DBLOGREADERBUFSIZE <buffer size>]}
[ASMUSER SYS@<ASM_instance>, ASMPASSWORD <password> [<algorithm>
    ENCRYPTKEY {<keyname> | DEFAULT}]]
[ASYNCTRANSPROCESSING <transaction-buffer-size> | NOASYNCTRANSPROCESSING]
[BUFSIZE <size>]
[COMPLETEARCHIVEDLOGONLY]
[DBLOGREADER]
[EXCLUDETRANS <trans name>]
[EXCLUDEUSER <user name>]
[EXCLUDEUSERID <Oracle uid>]
[FILTERTABLE <table_name>]
[FORCEFETCHLOB]
[FETCHLOBIFERROR]
[FETCHPARTIALLOB]
[FETCHPARTIALXML]
[IGNOREDATACAPTURECHANGES | NOIGNOREDATACAPTURECHANGES]
[IGNOREDIRECTLOADINSERTS]
[INCLUDEREGIONID | INCLUDEREGIONIDWITHOFFSET]
[INTEGRATEDPARAMS (<parameter> <value> [, ...])]
[LEGACYLOBREADING]
[LOGRETENTION [ENABLED | SR | DISABLED][LOGSOURCE <platform>, [PATHMAP <path to logs]]
[MANAGESECONDARYTRUNCATIONPOINT | NOMANAGESECONDARYTRUNCATIONPOINT]
[MAXREADSIZE <records>]
[MAXWARNEOF <seconds>]
[MININGUSER {/ | <user id>}[, MININGPASSWORD <password>]
    [<algorithm> ENCRYPTKEY {<keyname> | DEFAULT}] [SYSDBA]]
[NODDLCHANGEWARNING]
[NOFLUSH]
[PATHMAP <NFS mount point> <log path>]
[PURGEORPHANEDTRANSACTIONS | NOPURGEORPHANEDTRANSACTIONS]
[QUERYRETRYCOUNT <number of retries>] |
[READBUFFER <byte length>]
[READQUEUESIZE <size>]
[READTIMEOUT <milliseconds>]
[REQUIRELONGDATACAPTURECHANGES | NOREQUIRELONGDATACAPTURECHANGES]
[TRANSCLEANUPFREQUENCY <minutes>]
[VAMCOMPATIBILITY {1 | 2}]
}
[, ...]
```

適用対象 Extract

オプション	説明

`ALTARCHIVEDLOGFORMAT <string> [INSTANCE <instance_name>] [THREADID <id>]`	(Oracle) クラシック・キャプチャ・モードの Extract に有効です。
	ソース・データベースのアーカイブ・ログ形式をオーバーライドする文字列を指定します。<string> は、Oracle のパラメータ LOG_ARCHIVE_FORMAT と同じ指定子を受け入れます。Extract は、指定されるフォーマット指定子を使用してログ・ファイル名を導出します。例 :
	`arch_%T.arc`
	RAC 上で ALTARCHIVEDLOGFORMAT を使用するときは、各ノード上で ALTARCHIVEDLOGFORMAT パラメータを使用します。
	ALTARCHIVEDLOGFORMAT を含む TRANLOGOPTIONS 文には、他のすべての TRANLOGOPTIONS オプションを含めることはできません。他のオプションを指定するには、別の TRANLOGOPTIONS 文を使用してください。
	
	Extract にログ・ストリームを識別させるには、次の条件の 1 つ満たす必要があります。
	◆ INSTANCE または THREADID オプションを使用して、スレッド指定子 ("%t" または "%T") を含むログ名フォーマットを指定できます。
	◆ または、各スレッドに一意のログ・ディレクトリを使用します。この目的には、ALTARCHIVELOGDEST を使用します。(スレッド指定子を使用するログ名フォーマットを使用している場合も、一意のディレクトリの使用をお薦めします。)
	
	注意 : Extract がユーザー定義ログ・フォーマットを見つけられない場合や、スレッドにデフォルト・フォーマットが指定されていない場合は、データベースの 1 つのスレッドに問合せて取得したデフォルトのログ・フォーマットが別のスレッドにも適用されます。
	
	次のオプションは、RAC とともに使用します。Extract は、データベース・カタログに対して、提供された入力を検証します。
	◆ INSTANCE <instance_name> は、特定の Oracle インスタンスに ALTARCHIVEDLOGFORMAT を適用します。
	次に例を示します。
	`TRANLOGOPTIONS ALTARCHIVEDLOGFORMAT & INSTANCE rac1 log_%t_%s_%r.arc`
	◆ THREADID <id> は、指定のログ・フォーマットを持つインスタンスのスレッド番号を指定します。
	次に例を示します。
	`TRANLOGOPTIONS ALTARCHIVEDLOGFORMAT & THREADID 2 log_%t_%s_%r.arc`

オプション	説明

ALTARCHIVELOGDEST
[PRIMARY]
[INSTANCE <instance_name>]
[THREADID <id>]
<path name>
	(Oracle) クラシック・キャプチャ・モードの Extract に有効です。
	デフォルト以外の場所に存在するときに、Extract にアーカイブ・ログまたはバックアップ Oracle トランザクション・ログの場所を示します。Extract はまず指定された場所を確認し、次にデフォルトの場所を確認します。
	◆ <path name> には、代替ディレクトリのアーカイブ・ログの完全修飾パスを指定します。このディレクトリは、Oracle GoldenGate が実行されているノードに NFS マウントされている必要があります。このマウント・ポイントを ALTARCHIVELOGDEST に使用します。
	オプション :
	◆ PRIMARY では、代替場所でログが見つからない場合に、Extract がデフォルトのログの場所を確認することを防ぎます。ALTARCHIVELOGDEST パスのみが確認されます。PRIMARY は、アーカイブ・ログ専用 (ALO) モードで実行している Extract のデフォルトで、それ以外の場合にはオプションです。
	◆ INSTANCE <instance_name> は、特定の Oracle インスタンスに、指定された ALTARCHIVELOGDEST 動作を適用します。RAC でこのオプションを使用する場合は、各ノードに ALTARCHIVELOGDEST パラメータを指定する必要があります。
	◆ THREADID <id> は、特定のスレッド番号に、指定された ALTARCHIVELOGDEST 動作を適用します。
	どの Oracle インスタンスにも、複数の ALTARCHIVELOGDEST パラメータを指定できます。そのような場合、Extract は ALTARCHIVELOGDEST で指定されている順番で各場所を検索します。
	次に例を示します。
	`TRANLOGOPTIONS ALTARCHIVELOGDEST PRIMARY INSTANCE rac1 /disk1/node1/arch,`
	`ALTARCHIVELOGDEST INSTANCE rac1 /disk2/node1/arch,`
	`ALTARCHIVELOGDEST`
	`INSTANCE rac2 /disk1/node2/arch`
	この例では、Extract はインスタンス "rac1" に関連するログを /disk1/node1/arch で検索し、最初の検索に失敗すると、次は /disk2/node1/arch で検索します。Extract は、"rac1" のデフォルトの場所は確認しません。"rac2" については、まず /disk1/node2/arch を確認し、次にデフォルトの場所を確認します。

オプション	説明
`ALTARCHIVELOGDEST ("<backup path>" [FILESPEC "<file pattern>"] [[NOT] RECURSIVE] [PRIMARY])`	(SQL Server) デフォルト以外の場所に存在するときに、Extract にアーカイブまたはバックアップ・ログの場所を示します。Extract はまず指定された場所を確認し、次にデフォルトの場所を確認します。 ◆ パラメータ引数はカッコで囲みます。 ◆ "<backup path>" には、バックアップ・ログのパス名を指定します (二重引用符で囲みます)。最後のバックスラッシュ (\) ・デリミタの後に、ワイルドカード記号を使用できます。 アスタリスク (*) は、ゼロまたはそれ以上の文字と一致します。 疑問符 (?) は一文字と一致します。 NOT RECURSIVE を使用している場合は、このオプションを使用しないでください。 SQL Server の Extract パラメータ・ファイルには、1 つの TRANLOGOPTIONS ALTARCHIVELOGDEST エントリのみ指定できます。複数のエントリが存在する場合は、最後のエントリのみが使用されます。 オプション : ◆ FILESPEC "<file pattern>" には、"<backup path>" 内のファイル・パターンを指定します。ファイル・パターンは二重引用符で囲みます。 アスタリスク (*) は、ゼロまたはそれ以上の文字と一致します。 疑問符 (?) は一文字と一致します。 バックスラッシュ (\) ・デリミタは使用しないでください。バックスラッシュを使用すると他のパスも指定できますが、これは無効です。 ◆ [[NOT] RECURSIVE] では、"<backup path>" で指定されているファイルを再帰的に検索 (すべてのサブディレクトリも検索) するかどうかを指定します。 ◆ PRIMARY では、代替場所でログが見つからない場合に、Extract がデフォルトのログの場所を確認することを防ぎます。ALTARCHIVELOGDEST パスのみが確認されます。これはデフォルトです。
`ALTLOGDEST <path>`	(MySQL) MySQL のログ索引ファイルの場所を指定します。Extract は、データベースのデフォルトの場所ではなく、この場所でログ・ファイルを探します。データベース構成にログの完全パス名が含まれていない場合、またはマシンに複数の MySQL がインストールされている場合に、ALTLOGDEST を使用できます。Extract ではログ索引ファイルを読み取り、読取りが必要なバイナリ・ログ・ファイルが検索されます。ALTLOGDEST を使用する場合、Extract では、ログと索引は同じ場所にあるとみなされます。 次の例のように、ディレクトリの完全パス名を指定します。 `TRANLOGOPTIONS ALTLOGDEST "C:\Program Files\MySQL\MySQL Server 5.1\log\test.index"` Windows では、パスに空白が含まれる場合、パスを二重引用符で囲みます。

オプション	説明
ARCHIVEDLOGONLY	(Oracle) クラシック・キャプチャ・モードの Extract に有効です。 ARCHIVEDLOGONLY では、v$log および v$archived_log などのシステム・ビューのログの問合せや検証を行わずに、Extract にアーカイブ・ログからのみ読取りを実行させます。このパラメータにより、Extract はアーカイブ・ログ専用 (ALO) モードに入ります。要件および詳細は、『Oracle GoldenGate *Oracle インストレーションおよびセットアップ・ガイド*』を参照してください。 デフォルトでは、Extract は接続先が物理スタンバイ・データベースの場合でも、アーカイブ・ログ専用モードを使用しません。
ASMBUFSIZE <size>	(Oracle) クラシック・キャプチャ・モードの Extract に有効です。 トランザクション・ログの各読取り結果を保持する内部バッファへの読取り操作の最大サイズ (バイト) を制御します。ソース Oracle のリリースが次のいずれかの場合、DBLOGREADERBUFSIZE オプションのかわりにこのオプションを使用します。 ◆ Oracle 10.2.0.5 より前 ◆ 11.2.0.2 より前の 11g ◆ 任意の Oracle 11*g* R1 リリース これらのリリースは、DBLOGREADER オプションでサポートされている Oracle リリースで使用できる新しい API をサポートしていません。お使いの Oracle リリースでサポートされている場合、DBLOGREADER オプションを DBLOGREADERBUFSIZE オプションとともに使用することをお薦めします。(「DBLOGREADER」を参照してください。) 値を大きくすると抽出速度は向上しますが、Extract はより多くのメモリーを消費します。値を小さくすると、メモリーの使用量は減りますが、Extract はキャッシュ・サイズを超えるデータをディスクに保持する必要があるため、I/O は増加します。 次に、有効範囲とデフォルト・サイズ (バイト) を示します。 ◆ 最小 : REDO ログの 1 ブロックのサイズ ◆ 最大 : 28672 ◆ デフォルト : 28672 BUFSIZE オプションの値は、常に少なくとも ASMBUFSIZE の値以上である必要があります。

オプション	説明
ASMUSER SYS@<ASM_instance>, ASMPASSWORD <password> [<algorithm> ENCRYPTKEY {<keyname> | DEFAULT}]	(Oracle) クラシック・キャプチャ・モードの Extract に有効です。 トランザクション・ログを読み取る ASM インスタンスへのログインのためのユーザーおよびパスワードを指定します。 ◆ <user> は SYS である必要があります。 ◆ < password> は、ENCRYPT PASSWORD コマンドの結果からコピーされる暗号化されたパスワードです。 ◆ <algorithm> は、パスワードの暗号化に使用した暗号化アルゴリズムを、AES128、AES192、AES256 または BLOWFISH の中から指定します。 ◆ ENCRYPTKEY <keyname> は、ENCKEYS 参照ファイル内のユーザー作成の暗号化鍵の論理名を指定します。ENCRYPT PASSWORD が KEYNAME <keyname> オプションとともに使用された場合に使用します。 ◆ ENCRYPTKEY DEFAULT は、Oracle GoldenGate に、ランダム鍵を使用するように指示します。ENCRYPT PASSWORD が KEYNAME DEFAULT オプションとともに使用された場合に使用します。 **注意 :** このパラメータは、標準の USERID パラメータのかわりには使用 *できません*。ASM 環境では、両方のパラメータが必要です。ASMUSER は、ログを読み取るために DBLOGREADER オプションを使用している場合は不要です。 パスワード・セキュリティ機能の詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。
[ASYNCTRANSPROCESSING <transaction-buffer-size> | NOASYNCTRANSPROCESSING]	(Oracle) 統合キャプチャ・モードの Extract に有効です。統合キャプチャを非同期処理モードで実行するか、同期処理モードで実行するかを制御し、Extract が非同期モードの場合のバッファ・サイズを制御します。 ASYNCTRANSPROCESSING がデフォルトです。非同期トランザクション処理モードには、2 つの制御スレッドがあります。 ◆ 一方のスレッドは、論理変更レコード (LCR) をトランザクションにグループ化し、オブジェクトレベルのフィルタリングと部分ロールバック処理を行います。 ◆ 他方のスレッドは、コミットされたトランザクションをフォーマットし、ユーザー指定のトランスフォーメーションを実行してトレイル・ファイルに書き込みます。 トランザクション・バッファは、これらの 2 つのスレッド間のバッファで、あるスレッドから別のスレッドへの転送に使用されます。デフォルト・トランザクション・バッファ・サイズはコミット済トランザクション 300 件分ですが、キャッシュ・メモリーが不足している場合は Oracle GoldenGate メモリー・マネージャによって下方に調整されます。 NOASYNCTRANSPROCESSING は、非同期処理を無効にし、Extract を同期モードで動作させます。このモードでは、1 つのスレッドですべての取得処理を行います。

オプション	説明
`BUFSIZE <size>`	(DB2 LUW、DB2 z/OS、Oracle) トランザクション・ログから読み取られるデータを格納するために割り当てられるバッファの最大サイズ (バイト) を制御します。 ◆ ExtractがファイルベースのREDOを処理しているOracleソースの場合、このパラメータは、バッファへの読取り操作の最大サイズ (バイト) も制御します。 ◆ Extract が ASM REDO を処理している Oracle ソースの場合、ASMBUFSIZE または DBLOGREADERBUFSIZE のいずれかが読取りサイズを制御し、どちらの場合も BUFSIZE がバッファ・サイズを制御します。 値を大きくすると抽出速度は向上しますが、Extract はより多くのメモリーを消費します。値を小さくすると、メモリーの使用量は減りますが、Extract はキャッシュ・サイズを超えるデータをディスクに保持する必要があるため、I/O は増加します。 このパラメータは、ASMBUFSIZE または DBLOGREADERBUFSIZE (どちらを使用しているかによります) に設定されている値以上である必要があります。 次に、有効範囲とデフォルト・サイズ (バイト) を示します。 *Oracle*: 最小 : 8,192 最大 : 10,000,000 デフォルトのバッファ・サイズは REDO データのソースによって決定されます。 ◆ ファイルベースの REDO の場合、デフォルトは 1000KB (1024000) です。 ◆ ASM REDO の場合、デフォルトは 1000KB (1024000) です。 ◆ DBLOGREADER REDO の場合､デフォルトは 2MB (2097152) です。 *DB2 LUW:* 最小 : 40,960 最大 : 33,554,432 デフォルト : 131,072 上記の値は、4096 ページ・サイズの倍数である必要があります。指定された値がこの要件を満たさない場合、Extract は値を切り捨てて倍数にします。 システム管理者に依頼して、新しいバッファ・サイズをサポートするために十分な ECSA 領域を確保してください。

オプション	説明
COMPLETEARCHIVEDLOGONLY | NOCOMPLETEARCHIVEDLOGONLY	(Oracle) クラシック・キャプチャ・モードの Extract に有効です。 Extract のデフォルトのアーカイブ・ログ処理をオーバーライドします。 このパラメータに対して考えられる状態 : ◆ **通常モードのデフォルト :** NOCOMPLETEARCHIVEDLOGONLY。Extract は、アーカイブ・ログが使用可能になると、完全にディスクに書き込まれるのを待たずに、即座にアーカイブ・ログからの REDO データの処理を開始します。 **通常モードのオーバーライド** : COMPLETEARCHIVEDLOGONLY を使用して、アーカイブ・ログが完全にディスクに書き込まれるのを待ってから、Extract に REDO データの処理を開始させます。 ◆ **アーカイブ・ログ専用 (ALO) モードのデフォルト :** COMPLETEARCHIVEDLOGONLY。アーカイブ・ログが完全にディスクに書き込まれるのを待ってから、Extract に REDO データの処理を開始させます。 **ALO モードのオーバーライド :** NOCOMMPLETEARCHIVEDLOGONLY を使用して、アーカイブ・ログが使用可能になり次第、Extract に REDO データの処理を開始させます。 このパラメータは、本番 (ソース) アーカイブ・ログをセカンダリ・データベースにコピーし、データ・ソースとして使用するときに適用します。一部の Oracle プログラムは、最初のバイトから最後のバイトまで順序どおりにアーカイブ・ログを構築せずに、たとえばまず最初の 500MB、次に最後の 500MB、そして最後に中間の 1000MB をコピーする場合があります。Extract は、最初のバイトから読取りを開始した場合、バイト・シーケンスが中断する部分に到達したときに異常終了します。ファイル全体が書き込まれるまで待機させることにより、この問題を防止できます。 Extract は、完全にディスクに書き込まれるのを待たずにアーカイブ・ファイルの読取りを開始しますが、完了前にデータの取得を開始するかどうかは、前述の状態に依存することに注意してください。
COMPLETEARCHIVEDLOGTIMEOUT <seconds>	(Oracle) クラシック・キャプチャ・モードの Extract に有効です。 COMPLETEARCHIVEDLOGONLY モードの Extract が、REDO ログが完全にディスクに書き込まれたかどうかを検証できないときに、検証の再試行を待機する秒数を制御します。このオプションは、TRANLOGOPTIONS の COMPLETEARCHIVEDLOGONLY オプションとともに使用します。デフォルトでは、このオプションは無効化されており、Extract はファイルがディスクに書き込まれたかどうかを検証できない場合、10 秒後に異常終了します。検証では、最後のブロックからブロック・ヘッダーを読み取り、予測される順序番号と照合することにより、最後のブロックが書き込まれたかどうかを判断します。<seconds> には、0 よりも大きい任意の値を使用します。

オプション	説明
DBLOGREADER	(Oracle) クラシック・キャプチャ・モードの Extract に有効です。 Extract に、Oracle 10.2.0.5 以降の 10*g* R2 リリース、および Oracle 11.2.0.2 以降の 11*g* R2 リリース (ただし Oracle 11*g* R1 リリースではない) で使用可能な新しい ASM API を使用させます。この API は、Oracle ASM インスタンスに直接接続するのではなく、データベース・サーバーを使用して REDO ログおよびアーカイブ・ログにアクセスします。使用するデータベースは、この API モジュールを含むライブラリを含み、かつ実行中である必要があります。 この機能を使用するには、Extract データベース・ユーザーは SELECT ANY TRANSACTION 権限を持っている必要があります。 DBLOGREADER を使用する場合、Extract は最大 4MB の読取りサイズを使用できます。これは、DBLOGREADERBUFSIZE オプションで制御されます。 デフォルトの OCI バッファを使用するときの最大読取りサイズは、28672 バイトです。これは、ASMBUFSIZE オプションで制御されます。 REDO 率が高い場合、大きなバッファを使用することによって Extract のパフォーマンスが改善することがあります。 DBLOGREADER を使用するときは、TRANLOGOPTIONS の ASMUSER および ASMPASSWORD オプションを使用しないでください。この API は、USERID パラメータで指定されたユーザーおよびパスワードを使用します。
DBLOGREADERBUFSIZE <buffer size>	(Oracle) クラシック・キャプチャ・モードの Extract に有効です。 ASM でトランザクション・ログの各読取り結果を保持する内部バッファへの読取り操作の最大サイズ (バイト) を制御します。値を大きくすると抽出速度は向上しますが、Extract はより多くのメモリーを消費します。値を小さくすると、メモリーの使用量は減りますが、Extract はキャッシュ・サイズを超えるデータをディスクに保持する必要があるため、I/O は増加します。 ソース ASM インスタンスが Oracle 10.2.0.5 以降の 10*g* R2 リリース、または Oracle 11.2.0.2 以降の 11*g* R2 リリース (ただし Oracle 11*g* R1 ではない) である場合、DBLOGREADERBUFSIZE を DBLOGREADER オプションとともに使用します。これらのリリースで新しい ASM API を使用すると、古いリリースの場合よりもパフォーマンスが向上します。Oracle のリリースがこれらのリリースのいずれかではない場合、ASMBUFSIZE を使用する必要があります。

オプション	説明
	次に、有効範囲とデフォルト・サイズ (バイト) を示します。 ◆ 最小 : REDO ログの 1 ブロックのサイズ ◆ 最大 : 4 MB ◆ デフォルト : 2 MB (2097152) ほとんどのケースでは、デフォルト値で十分に機能するはずです。 BUFSIZE オプションの値は、常に少なくとも DBLOGREADERBUFSIZE の値以上である必要があります。
`EXCLUDETRANS <trans name>`	(Sybase、SQL Server)Replicat データベース・ユーザーまたは他の任意のユーザーのトランザクション名を指定し、Extract によるこれらのトランザクションの取得を防ぎます。双方向処理で、データベース間でのデータのループを防止するために使用します。 Replicat が使用するデフォルトのトランザクション名は ggs_repl ですが、EXCLUDETRANS では任意のトランザクションを指定できます。双方向同期の詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。
`EXCLUDEUSER <user name>`	(DB2 LUW、DB2 z/OS、Oracle、Sybase) GETREPLICATES または IGNOREREPLICATES パラメータのルール適用対象となるトランザクションを特定するフィルタとして使用する、Replicat データベース・ユーザーまたは他の任意のユーザーの名前を指定します。 通常このオプションは、双方向またはカスケード処理構成で、除外または取得する Replicat トランザクションを特定するために使用します。ただしこのパラメータは、特定のビジネス・アプリケーションなど、その他のユーザーのトランザクションの特定にも使用できます。 **z/OS 上の DB2 での考慮事項 :** z/OS 上の DB2 では、このユーザーは常にトランザクションのプライマリ許可 ID で、通常はログオンした元の RACF ユーザーの ID ですが、トランザクション・プロセッサまたは DB2 イグジットによって変更されているときは、別の許可 ID の場合もあります。

オプション	説明
	Oracle での考慮事項 : Oracle データベースでは、複数の EXCLUDEUSER 文を使用できます。指定されたすべてのユーザーは、GETREPLICATES または IGNOREREPLICATES ルールの対象になるという意味で、Replicat ユーザーと同一とみなされます。 1 つの文でユーザーの範囲を指定するために、ワイルドカードは使用できません。 EXCLUDEUSER と EXCLUDEUSERID は、同一のパラメータ・ファイルで使用できます。 ユーザー名は有効である必要があります。Oracle GoldenGate は、データベースに問い合せて関連するユーザー ID を取得し、この数値識別子をユーザー名にマップします。そのため、名前解決がデフォルトの DYNAMICRESOLUTION(191 ページ) に設定されているときに、指定されたユーザーが削除および再作成された場合、EXCLUDEUSER は有効のままです。名前解決が NODYNAMICRESOLUTION に設定されているときに同一の DDL が実行された場合、EXCLUDEUSER は無効になるため、EXCLUDEUSER を有効にするために Extract を停止および再起動する必要があります。
`EXCLUDEUSERID <Oracle uid>`	GETREPLICATES または IGNOREREPLICATES パラメータのルール適用対象となるトランザクションを特定するフィルタとして使用する、Replicat データベース・ユーザーまたは他の任意のユーザーの Oracle ユーザー ID (UID) を指定します。 使用方法は EXCLUDEUSER と同じです。 ユーザー ID は、負でない整数で、最大値は 2147483638 です。ユーザー ID を取得するために問い合せることのできるシステムはいくつかあります。最も簡便なものは ALL_USERS ビューです。Oracle GoldenGate はユーザー ID の検証を行いません。 注意 : 特定のユーザー ID に関連付けられているユーザーが削除および再作成された場合、このユーザーには新しいユーザー ID が割り当てられるため、このユーザーには EXCLUDEUSERID は無効になります。

オプション	説明
FETCHLOBIFERROR	(Oracle) クラシック・キャプチャ・モードの Extract に有効です。 REDO ログからの LOB キャプチャの結果が、不完全なデータなどのエラーになる場合、Extract のデフォルトの異常終了をオーバーライドします。REDO ログから LOB を読み取る際にエラーが発生する場合、Extract に、データベースから LOB をフェッチさせます。 **注意**: フェッチが発生する前に値が削除された場合、Extract によって NULL がトレイルに書き込まれます。フェッチの前に値が更新された場合、Extract によって更新された値が書き込まれます。これらの不正を回避するには、Extract の待機時間を少なくしてください。Oracle GoldenGate ドキュメントにプロセスのパフォーマンス・チューニングのガイドラインが記載されています。また、フェッチ・オプションの設定方法は、『Oracle インストレーションおよびセットアップ・ガイド』を参照してください。 「FORCEFETCHLOB」も参照してください。
FETCHPARTIALLOB	(Oracle) 統合キャプチャ・モードの Extract に有効です。Oracle 以外のターゲットにレプリケートする場合または完全な LOB イメージが必要なその他の状況では、このオプションを使用します。REDO レコードからの部分的な変更オブジェクトを使用せずに、Extract に完全な LOB オブジェクトをフェッチさせます。デフォルトでは、データベース・ログマイニング・サーバーは、ソース LOB のすべてが更新されたか一部が更新されたかに応じて、LOB の全体または一部を Extract に送信します。LOB のスナップショットが確実に正しいものになるように、Oracle Flashback 機能を表に対して有効にし、Extract が使用するように構成する必要があります。Extract の FETCHOPTIONS パラメータはフェッチを制御し、USESNAPSHOT (NOUSESNAPSHOT がない場合のデフォルト) に設定する必要があります。Flashback スナップショットがない場合、Extract は表から LOB をフェッチしますが、REDO レコードが生成された時点とは異なるイメージである可能性があります。
FETCHPARTIALXML	(Oracle) 統合キャプチャ・モードの Extract に有効です。Oracle 以外のターゲットにレプリケートする場合または完全な LOB イメージが必要なその他の状況では、このオプションを使用します。REDO レコードからの部分的な変更イメージを使用せずに、Extract に完全な XML ドキュメントをフェッチさせます。デフォルトでは、データベース・ログマイニング・サーバーは、ソース XML のすべてが更新されたか一部が更新されたかに応じて、XML ドキュメントの全体または一部を Extract に送信します。XML のスナップショットが確実に正しいものになるように、Oracle Flashback 機能を表に対して有効にし、Extract が使用するように構成する必要があります。Extract の FETCHOPTIONS パラメータはフェッチを制御し、USESNAPSHOT (NOUSESNAPSHOT がない場合のデフォルト) に設定する必要があります。Flashback スナップショットがない場合、Extract は表から XML ドキュメントをフェッチしますが、REDO レコードが生成された時点とは異なるイメージである可能性があります。

オプション	説明
FILTERTABLE <table_name>	(SQL/MX の Extract) このオプションでは、Replicat が使用するチェックポイント表名を指定します。チェックポイント表上の操作は、ソースへのデータのループバックを防止する手段として、ローカル Extract によって無視されます。チェックポイント表の作成の詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。
FORCEFETCHLOB	(Oracle) クラシック・キャプチャ・モードの Extract に有効です。 REDO ログから LOB データを取得するデフォルトの動作をオーバーライドします。デフォルトでは、データベースから LOB をフェッチさせます。 **注意** : フェッチが発生する前に値が削除された場合、Extract によって NULL がトレイルに書き込まれます。フェッチの前に値が更新された場合、Extract によって更新された値が書き込まれます。これらの不正を回避するには、Extract の待機時間を少なくしてください。Oracle GoldenGate ドキュメントにプロセスのパフォーマンス・チューニングのガイドラインが記載されています。また、フェッチ・オプションの設定方法は、『Oracle インストレーションおよびセットアップ・ガイド』を参照してください。
IGNOREDATACAPTURECHANGES | NOIGNOREDATACAPTURECHANGES	(DB2 LUW) IGNOREDATACAPTURECHANGES は、DATA CAPTURE CHANGES が設定されていない表を無視します。表がワイルドカードで指定されている場合に、変更取得セットが含まれている表の処理を継続するために使用します。スキップされた表に対する警告がエラー・ログに発行されます。デフォルトは NOIGNOREDATACAPTURECHANGES です。
IGNOREDIRECTLOADINSERTS	(Oracle) クラシック・キャプチャ・モードの Extract に有効です。 Extract にすべての Oracle ダイレクト・ロード INSERT を無視させます。デフォルトの動作 (このパラメータなし) では、Oracle ダイレクト・ロード INSERT を取得します。このオプションは、Oracle 10g 以上のログと互換性がある Oracle ログに適用されます。
INCLUDEREGIONID | INCLUDEREGIONIDWITHOFFSET	(Oracle) どちらのキャプチャ・モードの Extract にも有効です。 これらのオプションは、TZR ("US/Pacific" などのタイムゾーン・リージョンを表す) として指定された Oracle データ型 TIMESTAMP WITH TIME ZONE をサポートします。デフォルトでは、Extract は、タイムゾーン・リージョンが含まれる場合、TIMESTAMP WITH TIME ZONE で異常終了します これらのオプションにより、ターゲット・データベース・タイプに基づいて、このタイムスタンプを処理できます。 ◆ Oracle ソースから、同じリリース以降の Oracle ターゲットにレプリケートする場合、INCLUDEREGIONID を使用します。INCLUDEREGIONID を指定すると、Extract は、列索引と 2 バイトの TMZ 値をタイムゾーン・マッピング・トークンとして追加し、それを YYYY-MM-DD HH:MI.SS.FFFFFF +00:00 の UTC フォーマットでトレイルに出力します。

オプション	説明
	◆ v10*g*以降のOracleソースから10*g*より前のOracleターゲットに、または、Oracle ソースから Oracle データベースではないターゲットに、TZR として TIMESTAMP WITH TIME ZONE をレプリケートする場合、INCLUDEREGIONIDWITHOFFSET を使用します。INCLUDEREGIONIDWITHOFFSET を指定すると、Extract は、タイムゾーン・リージョン値を、日時に基づいて夏時間を考慮する時間オフセットに変換します。タイムスタンプ・データは、ローカル時間で、YYYY-MM-DD HH:MI.SS.FFFFFF TZH:TZM (TZH:TZM は時間オフセットに変換されたリージョン ID) のフォーマットで、トレイルに書き込まれます。 ソースのデータ型が TIMESTAMP で、リージョン ID マッピング・トークンがあることが検出された場合、Replicat は次のようにタイムスタンプを適用します。 ◆ ターゲットの Oracle リリースでサポートされている場合、TIMESTAMP WITH TIME ZONE と TZR が適用されます。 ◆ Oracle 以外のデータベース、または、TIMESTAMP WITH TIME ZONE と TZR をサポートしていない以前のリリースの Oracle には、タイムスタンプと UTC オフセットが適用されます。
`INTEGRATEDPARAMS (<parameter> <value> [, ...])`	(Oracle) 統合キャプチャ・モードの Extract に有効です (Oracle Standard または Enterprise Edition 11.2.0.3 以降)。 Extract が統合キャプチャ・モードの場合、パラメータを Oracle データベース・ログマイニング・サーバーに渡します。 入力は次のように <parameter> <value> の形式である必要があります。 `TRANLOGOPTIONS INTEGRATEDPARAMS (downsream_real_time_mine Y)` 有効な値 : ◆ max_sga_size データベース・ログマイニング・サーバーによって使用される SGA メモリーの量を指定します。正の整数をメガバイト単位で指定できます。streams_pool_size が 1 GB より大きい場合、デフォルトは 1 GB です。それ以外の場合、streams_pool_size の 75% がデフォルトになります。 ◆ parallelism データベース・ログマイニング・サーバーをサポートしているプロセスの数を指定します。正の整数を指定できます。デフォルトは 0 です。

オプション	説明
	◆ downstream_real_time_mine 統合キャプチャがダウンストリーム・マイニング・データベースをリアルタイム・モードでマイニングするかどうかを指定します。値 Y では、リアルタイムの取得を指定し、スタンバイ REDO ログがダウンストリーム・マイニング・データベースで構成されている必要があります。値 N は、ダウンストリーム・マイニング・データベースに転送されたアーカイブ・ログから取得することを指定します。デフォルトは N です。
LEGACYLOBREADING	SQL Server および Sybase に有効です。VAM モジュールで、Extract バージョン 11.1 以前に使用されていた LOB 格納メカニズムが使用されます。バージョン 11.2.1 より、別の LOB 格納メカニズムが使用されています。
LOGRETENTION [ENABLED | SR | DISABLED]	(Oracle) クラシック・キャプチャ・モードの Extract に有効です。 (Oracle Enterprise Edition 10.2 以降) Extract がクラシック・キャプチャ・モードで動作する場合に使用して、Extract がリカバリに必要なログ・ファイルを Oracle Recovery Manager (RMAN) が保持するかどうかを指定します。REGISTER EXTRACT コマンドを使用している場合、現在のデータベース SCN に基づいて、コマンドが発行された時点からログを保持します。ログは、手動で削除するまで保持されます。*注意*: このパラメータは、データベース自体の中の RMAN を有効化または無効化しません。 ENABLED ログ保持機能を有効化します。Oracle データベースの Extract がアーカイブ・ログ専用 (ALO) モードの場合を除いて、これはデフォルトです。REGISTER EXTRACT コマンドを LOGRETENTION オプションとともに使用して、Extract をデータベースに登録する必要があります。 **重要** : Oracle RAC で RMAN ログ保持をサポートするには、Extract グループを追加する前に、BUGFIX 11879974 で提供されているデータベース・パッチをダウンロードおよびインストールする必要があります。 ENABLED は、Bounded Recovery チェックポイントの SCN を適用し、そのポイントまで (そのポイントも含む) のログを保持します。このチェックポイントは、オープンしている最も古い *non-persisted* トランザクションのログ・ファイルを表します。Bounded Recovery の問題が永続データに影響する場合、オープンしている最も古いトランザクションを再処理するのに必要なログが使用できる必要があります。より保守的にするには、かわりに SR オプションを使用できます。(Bounded Recovery を理解するには、134 ページの「BR」を参照してください。)

オプション	説明
	SR ログ保持機能を有効化しますが、Extract が標準 (通常) リカバリ・モードに戻るのに必要なログの SCN まで (その SCN も含む) のログを保持します。標準モードでは、Extract は、メモリーにあったオープンしている最も古いトランザクションを含むログにアクセスする必要があります。SR を使用することは、Bounded Recovery モード (デフォルト) で保持されるログより多いログを保持する保守的な方法ですが、Bounded Recovery が失敗した場合にデータの可用性を確認します。REGISTER EXTRACT コマンドを LOGRETENTION オプションとともに使用して、Extract をデータベースに登録する必要があります。 DISABLED REGISTER EXTRACT が使用されている場合でも、ログ保持機能を無効にします。Oracle ソースの Extract がアーカイブ・ログ専用 (ALO) モードで動作している場合、これはデフォルト設定ですが、必要に応じてオーバーライドできます。 UNREGISTER EXTRACT コマンドを使用して、ログ保持の無効化後に関連する Extract グループをデータベースから登録解除します。52 ページを参照してください。 LOGRETENTION に関するその他の情報 : ◆ Oracle フラッシュ・リカバリ記憶域がいっぱいの場合、Extract で必要とされていても、RMAN によってアーカイブ・ログがパージされます。この制限は、Extract (および他の Oracle レプリケーション・コンポーネント) の要件がデータベースへの REDO の可用性を妨げないようにするために存在します。 ◆ LOGRETENTION では、基盤の (ただし機能しない) Oracle Streams 取得プロセスを利用します。そのため、データベースは Oracle の Enterprise Edition リリース 10.2 以上である必要があります。Oracle Standard Edition および Express Edition はこの機能をサポートしていません。LOGRETENTION 機能は、他の Streams インストールと同時に動作できます。 ◆ Extract に割り当てられていて、USERID パラメータで指定されているデータベース・ユーザーは、DBLOGIN(77 ページを参照してください) に必要な権限と同じ権限を持っている必要があります。

オプション	説明
LOGSOURCE <platform>, [PATHMAP <path to logs]	(Oracle) クラシック・キャプチャ・モードの Extract に有効です。 (Oracle) REDO ログまたはアーカイブ・ログ、またはその両方が、データベースをホストしているプラットフォーム以外のプラットフォームに保持されているときに、そのオペレーティング・システムおよびパス名 (オプション) を指定します。次の値が有効です。 ◆ AIX ◆ HPUX ◆ LINUX ◆ MVS ◆ SOLARIS ◆ VMS ◆ WINDOWS ◆ S390 オプションで、PATHMAP オプションを使用して、ログのパスを指定します。 正しいデータ位置を保持するために、LOGSOURCE プラットフォームと Extract を実行するプラットフォームは、次の要素が同一である必要があります。 ◆ エンディアン・オーダー ◆ ビット幅 (32 ビット、64 ビットなど) 次に互換性のあるエンディアン・プラットフォームを示します。 ◆ ビッグ・エンディアン : AIX、HPUX、MVS、SOLARIS、S290 ◆ リトル・エンディアン : LINUX、VMS、WINDOWS たとえば、Extract を HPUX で実行している場合、LOGSOURCE を AIX に設定すると有効ですが、LINUX は無効です。 LOGSOURCE を使用するときは、TRANLOGOPTIONS 文全体を 1 行にまとめてください。アンパサンド (&) 行終了文字を使用して行を複数に分割しないでください。
MANAGESECONDARYTRUNCATIONPOINT | NOMANAGESECONDARYTRUNCATIONPOINT	(SQL Server 2005 および Sybase) Oracle GoldenGate が 2 次切捨てポイントを維持するかどうかを指定します。 **SQL Server 2005:** MANAGESECONDARYTRUNCATIONPOINT は、Oracle GoldenGate が SQL Server レプリケーションと同時に実行 *されない* ため、Oracle GoldenGate が 2 次切捨てポイントを維持するときに使用します。NOMANAGESECONDARYTRUNCATIONPOINT は、Oracle GoldenGate が SQL Server レプリケーションと同時に実行されるときに使用します。SQL Server レプリケーションに 2 次切捨てポイントを管理させます。

オプション	説明
	Sybase: MANAGESECONDARYTRUNCATIONPOINT は、Oracle GoldenGate が Sybase Replication Server と同時に実行されないときに使用します。Extract が 2 次切捨てポイントを管理します。 NOMANAGESECONDARYTRUNCATIONPOINT は、Sybase トランザクション・ログを切り捨てないときに使用します。Extract は 2 次切捨てポイントを管理しません。このオプションは、Extract がデバッグ目的で以前のログ位置から Sybase トランザクション・ログの再読取りを行う必要があるときに使用できます。
`MAXREADSIZE <records>`	Sybase に有効です。Extract でトランザクション・ログから一度に読み取られるレコードの数を指定します。パフォーマンスの向上に使用できます。有効な値は、1 から 50000 までの整数です。デフォルトは 256 レコードです。このパラメータを高い値に調整するときは注意してください。これにより、Extract が 2 次切捨てポイントを調整する頻度が減り、ログ・データが蓄積されることがあります。高い値に調整する前に、10000 から開始して、パフォーマンスを評価してください。
`MAXWARNEOF <seconds>`	(Oracle) クラシック・キャプチャ・モードの Extract に有効です。 警告メッセージを生成するまで、Extract が新しいログ・ファイルを待機する秒数を指定します。Extract は、特定の順序番号に対して警告メッセージを 1 つのみ生成します。MAXWARNEOF を指定しない場合、Extract はデフォルトで 1 時間待機します。値 0 を指定すると、Extract が待機した時間にかかわらず、警告メッセージは生成されません。

`MININGUSER {/ | <user id>}`
`[, MININGPASSWORD <password>]`
`[<algorithm> ENCRYPTKEY {<keyname> |`
`DEFAULT}]`
`[SYSDBA]]`
	(Oracle) 統合キャプチャ・モードの Extract に有効です。
	ログマイニング・サーバーとやり取りするダウンストリーム Oracle マイニング・データベースにログインするための、統合キャプチャ・モードでの Extract のログイン資格証明を指定します。ユーザーは、次を満たす必要があります。
	◆ dbms_goldengate_auth.grant_admin_privilege で付与される権限を持っている。
	◆ この MININGUSER に関連付けられている Extract グループに、MININGDBLOGIN、および REGISTER EXTRACT または UNREGISTER EXTRACT を発行するユーザーである。
	◆ Extract が統合キャプチャ・モードである間は変更されない。

オプション	説明

	MININGUSER のオプションは、次のとおりです。
	◆ / は、データベース・ユーザー・ログインではなく、Oracle 用のオペレーティングシステム・ログインを使用するように Oracle GoldenGate に指示します。この引数は、データベースによってオペレーティングシステム・レベルでの認証が許可されている場合にのみ使用します。データベースレベルの認証をバイパスすることにより、アプリケーションのパスワードが頻繁に変更される場合に、Oracle GoldenGate パラメータ・ファイルを更新する必要がなくなります。
	このオプションを使用するには、Oracle OS_AUTHENT_PREFIX 初期化パラメータとの関連で、正しいユーザー名がデータベースに存在している必要があります。OS_AUTHENT_PREFIX で指定されている値は、ユーザーのオペレーティング・システム・アカウント名の先頭に追加され、データベース名と比較されます。この 2 つの名前は一致する必要があります。
	OS_AUTHENT_PREFIX が " " (NULL 文字列) に設定されている場合は、ユーザー名を "identified externally" として作成する必要があります。たとえば、OS ユーザー名が "ogg" の場合は、次のようにしてデータベース・ユーザーを作成します。
	`CREATE USER ogg IDENTIFIED EXTERNALLY;`
	OS_AUTHENT_PREFIX が OPS$ または別の文字列に設定されている場合は、ユーザー名は次のフォーマットで作成する必要があります。
	`<OS_AUTHENT_PREFIX_value><OS_user_name>`
	たとえば、OS ユーザー名が "ogg" の場合は、次のようにしてデータベース・ユーザーを作成します。
	`CREATE USER ops$ogg IDENTIFIED BY oggpassword;`
	◆ <user id> は、マイニング・データベース・ユーザー名または SQL*Net 接続文字列を指定します。
	◆ <password> はパスワードです。データベース・ユーザーのパスワードを指定するためにデータベース認証が必要な場合に使用します。
	パスワードが ENCRYPT PASSWORD コマンドによって暗号化されている場合は、暗号化されたパスワードを指定します。それ以外の場合は、クリアテキストのパスワードを使用します。パスワードに大文字と小文字の区別がある場合は、そのとおりに入力します。
	ユーザー ID またはパスワードのいずれかが変更されると、必要に応じて、パスワードの再暗号化など、Oracle GoldenGate パラメータ・ファイルの変更を行う必要があります。

オプション	説明

	◆ <algorithm> は、ENCRYPT PASSWORD でパスワードの暗号化に使用した暗号化アルゴリズムを指定します。次のいずれかになります。
	`AES128`
	`AES192`
	`AES256`
	`BLOWFISH`
	
	◆ ENCRYPTKEY {<keyname> | DEFAULT} は、ENCRYPT PASSWORD で指定した暗号化鍵を指定します。
	ENCRYPTKEY <keyname> は、ENCKEYS 参照ファイル内のユーザー作成の暗号化鍵の論理名を指定します。ENCRYPT PASSWORD が KEYNAME <keyname> オプションとともに使用された場合に使用します。
	ENCRYPTKEY DEFAULT は、Oracle GoldenGate に、ランダム鍵を使用するように指示します。ENCRYPT PASSWORD が KEYNAME DEFAULT オプションとともに使用された場合に使用します。
	
	◆ SYSDBA は、ユーザーを sysdba としてログインするように指定します。

`NODDLCHANGEWARNING`	(SQL Server)Extract がデータを取得するソース・オブジェクトに DDL 操作が実行されたときに、Extract が警告を記録することを防ぎます。デフォルトでは、問題を修正できるように警告がレポートされます。Oracle GoldenGate は、SQL Server に対して DDL 取得およびレプリケーションをサポートしていないため、ソースおよびターゲットのメタデータは一定のままであるとみなします。DDL 変更によって Extract が異常終了しなくても、このような変更が発生するたびに警告が記録されます。NODDLCHANGEWARNING を使用すると、Oracle GoldenGate ログにこのようなメッセージが蓄積されることを防止できます。
`NOFLUSH`	(DB2 z/OS) ログ・バッファのフラッシュを防止します。
`PATHMAP <NFS mount point> <log path>`	(Oracle) クラシック・キャプチャ・モードの Extract に有効です。 REDO ログまたはアーカイブ・ログ、またはその両方が、データベースをホストしているシステム以外のシステムに保持されているときに、その場所を指定します。NFS マウント・ポイントに続き、Oracle ログ構造のパスを指定します。1 つ以上の PATHMAP 文を使用できます。 このシステムがデータベースをホストしているシステムとは異なるプラットフォームの場合は、LOGSOURCE オプションとともに使用できます。 PATHMAP を使用するときは、TRANLOGOPTIONS 文全体を 1 行にまとめてください。アンパサンド (&) 行終了文字を使用して行を複数に分割しないでください。

オプション	説明
PURGEORPHANEDTRANSACTIONS | NOPURGEORPHANEDTRANSACTIONS	(Oracle) クラシック・キャプチャ・モードの Extract に有効です。 Oracle RAC ノードに障害が発生し、Extract がロールバックを取得できないときに発生する孤立トランザクションのパージを有効化または無効化します。トランザクションは、パージの実行前に、自身の開始時間とノードの開始時間の比較によって孤立しているかどうかが検証され、ノードよりも開始時間が早いトランザクションはパージされます。デフォルトは PURGEORPHANEDTRANSACTIONS です。
QUERYRETRYCOUNT <number of retries>	(SQL Server の Extract) タイムアウト後に表のメタデータを取得するために問合せを再試行する回数を指定します。デフォルトは、30 秒の待機後に 1 回の再試行で、この再試行が失敗するとプロセスは異常終了します。 QUERYRETRYCOUNT では、指定される入力値に従って 30 秒間隔で複数回再試行するように指定できます。すべての再試行が失敗すると、Extract は通常の接続エラー・メッセージとともに異常終了します。 表を作成した長時間におよぶトランザクションでは、タイムアウトが発生することがあります。システム表はロックされ、Extract の問合せは完了できなくなります。 次に例を示します。 `TRANLOGOPTIONS QUERYRETRYCOUNT 4` この例では、Extract は 30 秒間隔で 4 回問合せを試行します。
READQUEUESIZE <size>	Sybase に有効です。トランザクション・データの内部キューのサイズをバイト単位で指定します。これによりパフォーマンスが向上します。有効な値は、10 から 50000 までの整数です。デフォルトは 256 バイトです。高い値に調整する前に、10000 から開始して、パフォーマンスを評価してください。
REQUIRELONGDATACAPTURECHANGES | NOREQUIRELONGDATACAPTURECHANGES	(DB2 LUW) 次の状況での Extract のレスポンスを制御します。 ◆ DATA CAPTURE が NONE、または INCLUDE LONGVAR COLUMNS なしで CHANGES に設定されている および ◆ パラメータ・ファイルに、一部またはすべての列値のビフォア・イメージを必要とする Oracle GoldenGate パラメータ (GETBEFOREUPATES、NOCOMPRESSUPDATES、および NOCOMPRESSDELETES) が含まれている この 2 つの DATA CAPTURE 設定では、LONGVAR 列のビフォア値が記録されません。Extract がこれらの列を使用できない場合、ターゲット・データの整合性に影響が及ぶことがあります。

オプション	説明
	◆ REQUIRELONGDATACAPTURECHANGES Extract はエラーとともに異常終了します。 ◆ NOREQUIRELONGDATACAPTURECHANGES Extract は警告を出しますが、データ・レコードの処理を継続します。
`TRANSCLEANUPFREQUENCY <minutes>`	(Oracle) クラシック・キャプチャ・モードの Extract に有効です。 Oracle GoldenGate が孤立トランザクションのスキャン、再スキャン、および削除の一連の動作を実行する間隔 (分) を指定します。最初のスキャンでは、孤立していると思われるトランザクションをマーク付けします。2 回目のスキャンで孤立していることを確認し、次にこれらのトランザクションを削除します。有効な値は 1 ～ 43200 分です。デフォルトは 10 分です。
`VAMCOMPATIBILITY {1 | 2}`	(MySQL、SQL M/X、SQL Server、Sybase、Teradata) *Vendor Access Module* (VAM) と呼ばれる Oracle GoldenGate API 間で、各 VAM 実装の必要性に応じて、異なるメタデータ構造が渡されることを許可します。 ◆ 値 1 は、元の VAM API メタデータ構造の使用を指定します。このメタデータ構造は、Oracle GoldenGate for Teradata のために実装されています。Teradata モジュールは、共有ライブラリとして構成されており、新しいバージョンのようにプログラムによって設定されないため、TRANLOGOPTIONS VAMCOMPATIBILITY の使用が必要な唯一の VAM モジュールです。TRANLOGOPTIONS VAMCOMPATIBILITY は、古い TAM モジュールとともに新しい Oracle GoldenGate for Teradata の Extract を使用している場合に、古いモジュールとの下位互換性をサポートするために使用します。VAM 互換性を VAMInitialize で設定している場合は、TRANLOGOPTIONS で設定する必要はありません。このパラメータは、Extract と TAM モジュールが同じバージョンの場合には必要ありません。 ◆ 値 2 は、元の構造をベースにフィールドが追加されている、拡張 VAM API メタデータ構造の使用を指定します。この構造は、現在 Oracle GoldenGate for Sybase 製品で使用されています。この値は VAM 内でプログラムによって設定されるため、TRANLOGOPTIONS VAMCOMPATIBILITY を使用する必要はありません。

例 1 次の例では、アーカイブ・ログの場所を指定します。

```
TRANLOGOPTIONS ALTARCHIVELOGDEST /fs1/oradata/archive/log2
```

例 2 次の Oracle の例では、2 ユーザーを (1 つは名前、1 つはユーザー ID で) フィルタし、GETREPLICATES または IGNOREREPLICATES ルールに従ってこれらのユーザーのトランザクションを処理し、新しいトランザクション・バッファ・サイズを設定します。

```
TRANLOGOPTIONS EXCLUDEUSER ggsrep, EXCLUDEUSERID 90, BUFSIZE 100000
```

例 3 次の例では、SQL Server または Sybase 環境で Replicat トランザクション名を除外します。

```
TRANLOGOPTIONS EXCLUDETRANS "ggs_repl"
```

例 4 次に、データベースをホストするプラットフォームと別のプラットフォーム上にあるトランザクション・ログを処理する方法を示します。注意 : この文は、このドキュメントのスペースの制約によって、複数の行にまたがっています。

```
TRANLOGOPTIONS, LOGSOURCE VMS, PATHMAP DKA200:[RDBMS.ORACLE.ORA9201I.64.ADMIN.GGS.ARCH]
/net/deltan/uservol1/RDBMS.DIR/ORACLE.DIR/ORA9201I.DIR/
64.DIR/admin.DIR/ggs.DIR/ARCH.dir PATHMAP
DKA200:[RDBMS.ORACLE.ORA9201I.64.ORADATA.GGS]
/net/deltan/uservol1/rdbms.dir/oracle.dir/ora9201I.DIR/
64.dir/oradata.dir/ggs.dir
```

例 5 次に、ALTONLINELOGS のいくつかの使用例を示します。空白を含むパスは、二重引用符で囲まれています。

```
TRANLOGOPTIONS ALTONLINELOGS ("third one/log1.txt")
TRANLOGOPTIONS ALTONLINELOGS( "first one/log1.txt", second_one/log2.txt )
TRANLOGOPTIONS ALTONLINELOGS ( sixth_one/log1.txt, seventh_one/log2.txt, &
    "eighth one/log2.txt")
```

例 6 次に、ASM ユーザーのパスワードを暗号化する方法を示します。

```
TRANLOGOPTIONS ASMUSER SYS@asm1, &
ASMPASSWORD AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC, &
AES128, ENCRYPTKEY securekey1
```

TRANSACTIONTIMEOUT

適用対象 Replicat

TRANSACTIONTIMEOUT パラメータでは、コミットされていない Replicat ターゲット・トランザクションによる、ターゲット・データベースのロックおよびリソースの無駄な消費を防止します。Replicat が既存のアプリケーション・タイムアウト、およびターゲットのその他のデータベース要件内で動作できるように、このパラメータの値を変更できます。

TRANSACTIONTIMEOUT では、Replicat がターゲット・トランザクション内の最後のソース・トランザクションのトランザクション終了レコードを受信していない場合に、ターゲット・トランザクションをオープンしておく時間を制限します。デフォルトでは、Replicat は複数のソース・トランザクションを 1 つのターゲット・トランザクションにグループ化してパフォーマンスを向上させますが、ソース・トランザクションの一部のみのコミットは行わず、最後のレコードが到着するまで無制限に待ちます。Replicat パラメータの GROUPTRANSOPS では、グループ化されるターゲット・トランザクションの最小サイズを制御します。

次のイベントは、長期化して TRANSACTIONTIMEOUT をトリガーする場合があります。

- ネットワークの問題によって、トレイル・データがターゲット・システムに転送されない。
- いずれかのシステムのディスク容量が不足し、トレイル・データが書き込まれない。
- Collector の異常終了 (ほとんど発生しないイベントです)。

- トランザクションのレコードの書き込み中の Extract の異常終了または停止。
- Extract データ・ポンプの異常終了または停止。
- 停電やシステムのクラッシュなど、ソース・システムの障害。

TRANSACTIONTIMEOUT の動作

通常の操作では、Replicat は、トランザクションが異常終了し、再試行が必要になるときに備えて、現在のターゲット・トランザクション内の最初のソース・トランザクションの開始位置を記憶しています。TRANSACTIONTIMEOUT を有効にすると、Replicat は、現在のソース・トランザクションの最初のレコードの位置も記憶し、TRANSACTIONTIMEOUT がトリガーされた場合に、この位置を論理的な終了 (EOL) 位置として使用します。

TRANSACTIONTIMEOUT は、トリガーされると次のことを行います。

1. 現在のターゲット・トランザクションを中止します。
2. 中止されたターゲット・トランザクション内の最初のソース・トランザクションの開始位置に移動します。
3. 論理終了位置 (最後の完了していないソース・トランザクションの開始位置) まで、すべてのトレイル・レコードを処理します。
4. 論理 EOF 位置でトランザクションをコミットします。
5. 新しいトレイル・データを待機してから、次のトレイル・レコードを処理します。

TRANSACTIONTIMEOUT は、データの到着を遅らせ、TRANSACTIONTIMEOUT をトリガーする問題の性質に応じて、同一のソース・トランザクションに対して何度もトリガーできます。

TRANSACTIONTIMEOUT 状況の発生の確認

TRANSACTIONTIMEOUT が有効な場合に、Replicat がソース・トランザクションの残りの部分を待機しているかどうかを確認するには、SEND REPLICAT コマンドを STATUS オプションとともに発行します。次のステータスは、Replicat が待機していることを示します。

```
Performing transaction timeout recovery
Waiting for data at logical EOF after transaction timeout recovery
```

デフォルト 無効

構文

```
TRANSACTIONTIMEOUT <n> <units>
```

オプション	説明
<n>	待機する間隔を指定する整数。有効な値は 1 秒～ 1 週間 (7 日間) です。この値は、プライマリ Extract および関連するすべてのデータ・ポンプ両方で設定されている EOFDELAY パラメータの値より大きくする必要があります。
<units>	次のいずれか 1 つを使用します : S、SEC、SECS、SECOND、SECONDS、MIN、MINS、MINUTE、MINUTES、HOUR、HOURS、DAY、DAYS。

例

```
TRANSACTIONTIMEOUT 5 S
```

TRANSMEMORY

適用対象 z/OS 上の DB2 および NonStop SQL/MX の Extract

TRANSMEMORY では、コミットされていないトランザクション・データのキャッシュに使用可能なメモリーおよび一時ディスク領域量を制御します。Oracle GoldenGate はコミットされたトランザクションのみをターゲット・データベースに適用するため、コミットまたはロールバックのインジケータを受信するまでソース・システム上にトランザクション・データを保持するための十分なシステム・メモリーが必要です。

このパラメータは、z/OS 上の DB2 データベースおよび NonStop SQL/MX データベースとともに使用します。他のすべてのデータベースでは、CACHEMGR パラメータを使用します。

TRANSMEMORY を使用したメモリー管理について

TRANSMEMORY を使用して、Oracle GoldenGate のトランザクション用のキャッシュ・サイズを調整し、キャッシュ・サイズを超えるデータを保持するディスク上の一時的な場所を定義できます。合計キャッシュ・サイズ、トランザクション当たりのメモリー・サイズ、初期および増分メモリー割当て、ディスク記憶域領域を定義するためのオプションを使用できます。

トランザクションは、RAM によって指定されたメモリー・プールに追加され、TRANSRAM の値に達したときにディスクにフラッシュされます。初期メモリー量は、INITTRANSRAM に基づいて各トランザクションに割り当てられ、必要に応じて、RAMINCREMENT で指定された量ずつ、TRANSRAM で設定された最大値まで増分されます。したがって、TRANSRAM には (INITTRANSRAM + RAMINCREMENT) の合計によって割り切れる値を指定する必要があります。

現在の TRANSMEMORY 設定を表示するには、GGSCI で VIEW REPORT <group> コマンドを使用します。

z/OS での特別の考慮事項 :

z/OS システムでは、RAM オプションはすべてのキャッシュされたトランザクションの仮想メモリー割当ての合計を制御するだけではなく、起動中に割り当てられるヒープ・メモリーのサイズも制御します。大きなデフォルト値によって、仮想メモリー・プール内の断片化を防止できますが、一部のインストールでは、特にアプリケーションが主に小さなトランザクションを生成する場合に、仮想メモリーが浪費されることがあります。また、大量のヒープを割り当てると、z/OS が割当てを完了するまで、Extract が起動時に反応しなくなることがあります。

z/OS 上では、RAM は Extract のパフォーマンスに影響を与えない、トランザクション・アクティビティの保持に必要な最小限の値に設定してください。低すぎる値に設定すると、Extract がディスクにトランザクション・データを書き込み、Extract の動作速度の低下やディスク領域消費の原因になることがあります。適切な値を決定するには、テストが必要になることがあります。

デフォルト なし

構文

```
TRANSMEMORY
[RAM <size>]
[TRANSRAM <size>]
[TRANSALLSOURCES <size>]
[INITTRANSRAM <size>]
[RAMINCREMENT <size>]
[DIRECTORY (<directory name>, <max dir size>, <max file size>)]
```

オプション	説明
`RAM <size>`	キャッシュされたすべてのトランザクションが使用する合計メモリー量を指定します。z/OS では、この値は *トランザクション当たり* に割り当てる処理メモリー量でもあります。 デフォルトは 200MB です。この値は、バイト、または次のいずれかの形式の GB、MB または KB で指定できます。 GB | MB | KB | G | M | K | gb | mb | kb | g | m | k
`TRANSRAM <size>`	1 つのトランザクションが使用する合計メモリー量を指定します。デフォルトは 50MB です。この値は、バイト、または次のいずれかの形式の GB、MB または KB で指定できます。 GB | MB | KB | G | M | K | gb | mb | kb | g | m | k TRANSRAM は、最適な成果を得るために、INITTRANSRAM および RAMINCREMENT 両方の値によって割り切れる値に設定する必要があります。
`TRANSALLSOURCES <size>`	1 つのトランザクションが利用するメモリーおよびディスク領域の量の合計を指定します。デフォルトは、使用可能なメモリー (メモリーおよびディスク) 合計の 50% です。この値は、バイト、または次のいずれかの形式の GB、MB または KB で指定できます。 GB | MB | KB | G | M | K | gb | mb | kb | g | m | k
`INITTRANSRAM <size>`	(NonStop システムのみ) トランザクションに割り当てる初期メモリー量を指定します。デフォルトは 500KB です。この値は、バイト、または次のいずれかの形式の GB、MB または KB で指定できます。 GB | MB | KB | G | M | K | gb | mb | kb | g | m | k
`RAMINCREMENT <size>`	トランザクションがさらに多くのメモリーを必要とするときに増分するメモリー量を指定します。デフォルトは 500KB です。この値は、バイト、または次のいずれかの形式の GB、MB または KB で指定できます。 GB | MB | KB | G | M | K | gb | mb | kb | g | m | k

`DIRECTORY (<directory name>,`
`<max dir size>,`
`<max file size>)`
	トランザクション・データのサイズが TRANSRAM で指定した最大値を超えたときにこのデータが使用する一時ディスク記憶域を指定します。DIRECTORY は、複数回指定できます。
	◆ <directory name> は、ディレクトリの完全修飾名です。デフォルトは、Oracle GoldenGate ディレクトリの dirtmp サブディレクトリです。
	◆ <max dir size> は、ディレクトリ内のすべてのファイルの最大サイズです。デフォルトは 2GB です。指定した領域が使用できない場合は、使用可能なディスク領域の 75% が使用されます。
	◆ <max file size> は、ディレクトリ内の各ファイルの最大サイズです。デフォルトは 200MB です。

オプション	説明
	この値は、バイト、または次のいずれかの形式の GB、MB または KB で指定できます。 GB | MB | KB | G | M | K | gb | mb | kb | g | m | k ディレクトリ・サイズおよびファイル・サイズは、RAM で指定したメモリー・サイズより大きい必要があります。 ファイル名は、次のフォーマットを使用します。 <group>_trans_00001.mem または <PID>_trans_00001.mem グループ名は、オンライン処理で使用されます。システム・プロセス ID 番号 (PID) は、SPECIALRUN パラメータで指定したワンタイム実行で使用されます。 スレッド Extract のフォーマットは次のようになります (データベースによって異なります)。 <group>_<thread #>_00001.mem

例 1 次の例では、データをディスクにフラッシュする前に、各トランザクションに 10 回のメモリー追加を許可します (INITTRANSRAM で指定された初期割当て 1 回と、RAMINCREMENT で許可された 9 回)。

```
TRANSMEMORY DIRECTORY(c:\test\dirtmp, 3000000000,
300000000), RAM 8000K, TRANSRAM 1000K, INITTRANSRAM 100K,
RAMINCREMENT 100K
```

例 2 次の例は上記の例と同じですが、2 つ目のディレクトリも追加します。

```
TRANSMEMORY DIRECTORY(c:\test\dirtmp, 3000000000,
300000000), DIRECTORY (c:\test\dirtmp2, 1000000000,
5000000), RAM 8000K, TRANSRAM 1000K, INITTRANSRAM 100K,
RAMINCREMENT 100K
```

注意 **上記の例では、スペースの制約によってパラメータ指定が複数行にわたっています。実際のパラメータ・ファイルでは、パラメータ指定が複数行にわたる場合は、各行の末尾にアンパサンド (&) を含める必要があります。**

TRIMSPACES | NOTRIMSPACES

適用対象 Extract および Replicat

TRIMSPACES および NOTRIMSPACES パラメータでは、ソースの CHAR 列の末尾の空白を、ターゲットの CHAR または VARCHAR 列に適用するときに切り捨てるかどうかを制御します。

注意 **Sybase は、すべての CHAR 型を VARCHAR 型として処理し、このため TRIMSPACES は無効になります。Sybase では、TRIMVARSPACES パラメータを使用します。**

TRIMSPACES は、シングルバイトの空白 (U+0020) にのみ適用されます。表意空白 (U+3000) はサポートされていません。

TRIMSPACES および NOTRIMSPACES は、パラメータ・ファイル内の異なる TABLE または MAP 文の個別スイッチとして使用できます。また、個別の TABLE または MAP 文内で使用して、特定の MAP または TABLE 文のグローバル設定よりも優先させることができます。

Extract では、Extract が (TARGET 文を使用して)TABLE 文内でマッピングを実行している場合、TRIMSPACES のみが有効になります。

デフォルト TRIMSPACES

構文 TRIMSPACES | NOTRIMSPACES

例 次の例では、最後の MAP 文を除き、デフォルトの末尾空白切捨てを行います。

```
MAP fin.src1, TARGET fin.tgt1;
MAP fin.src2, TARGET fin.tgt2;
MAP fin.src3, TARGET fin.tgt3;
NOTRIMSPACES
MAP fin.src4, TARGET fin.tgt4;
```

TRIMVARSPACES | NOTRIMVARSPACES

適用対象 Extract および Replicat

TRIMVARSPACES および NOTRIMVARSPACES パラメータでは、ソースの VARCHAR 列の末尾の空白を、ターゲットの CHAR または VARCHAR 列に適用するときに切り捨てるかどうかを制御します。

VARCHAR 列の空白は実際はデータの一部であるため、デフォルトは NOTRIMVARSPACES です。TRIMVARSPACES を使用する前に、末尾の空白がターゲット・データにとって不可欠ではないことを確認してください。

TRIMVARSPACES および NOTRIMVARSPACES は、パラメータ・ファイル内の異なる TABLE または MAP 文の個別スイッチとして使用できます。また、個別の TABLE または MAP 文内で使用して、特定の MAP または TABLE 文のグローバル設定よりも優先させることができます。

Extract では、Extract が (TARGET 文を使用して)TABLE 文内でマッピングを実行している場合、TRIMVARSPACES のみが有効になります。

デフォルト NOTRIMVARSPACES

構文 TRIMVARSPACES | NOTRIMVARSPACES

例 次の例では、最後の MAP 文に対して、末尾空白切捨てを有効化します。

```
MAP fin.src1, TARGET fin.tgt1;
MAP fin.src2, TARGET fin.tgt2;
MAP fin.src3, TARGET fin.tgt3;
TRIMVARSPACES
MAP fin.src4, TARGET fin.tgt4;
```

UPDATEDELETES | NOUPDATEDELETES

適用対象 Replicat

UPDATEDELETES パラメータでは、パラメータ・ファイル内でこのパラメータ以降のすべての MAP 文の削除操作を更新操作に変換します。UPDATEDELETES を無効にするには、NOUPDATEDELETES を使用します。

UPDATEDELETES を使用するときは、NOCOMPRESSDELETES パラメータを使用します。このパラメータによって、Extract は、更新に使用できるように、すべての列をトレイルに書き込みます。

デフォルト NOUPDATEDELETES

構文

```
UPDATEDELETES | NOUPDATEDELETES
```

UPDATEINSERTS | NOUPDATEINSERTS

適用対象 Replicat

UPDATEINSERTS パラメータでは、パラメータ・ファイル内でこのパラメータ以降のすべての MAP 文の挿入操作を更新操作に変換します。UPDATEINSERTS を無効にするには、NOUPDATEINSERTS を使用します。

デフォルト NOUPDATEINSERTS

構文

```
UPDATEINSERTS | NOUPDATEINSERTS
```

UPREPORT

適用対象 Manager

UPREPORTMINUTES または UPREPORTHOURS パラメータでは、Manager が実行中の Extract および Replicat プロセスをレポートする間隔を指定します。これらのいずれかのプロセスが開始または停止するたびに、イベントが生成されます。エラー・ログが非常に大きい場合には、ログ内のこうしたメッセージは見逃されてしまいがちです。UPREPORTMINUTES および UPREPORTHOURS は、このようなプロセス・ステータスが見逃されないように、定期的にレポートします。

停止されたプロセスをレポートするには、DOWNREPORT パラメータを使用します。詳細は、188 ページを参照してください。

デフォルト 実行中のプロセスをレポートしない。

構文

```
UPREPORTMINUTES <minutes> | UPREPORTHOURS <hours>
```

引数	説明
<minutes>	実行中のプロセスをレポートする間隔 (分)。
<hours>	実行中のプロセスをレポートする間隔 (時間)。

例 次の例では、30 分間隔でレポートが生成されます。

```
UPREPORTMINUTES 30
```

USEANSISQLQUOTES

適用対象 GLOBALS

USEANSISQLQUOTES パラメータでは、Oracle GoldenGate に、大 / 小文字が区別される列名や特殊文字が含まれる列名を認識させます。USEANSISQLQUOTES によって、Oracle GoldenGate は、識別子およびリテラル文字列を区切るための引用符を使用する SQL-92 ルールに従うことができます。

USEANSISQLQUOTES を有効にすると、Oracle GoldenGate は、二重引用符で囲まれた文字列を列名として、一重引用符で囲まれた文字列をリテラルとして処理します。デフォルトでは、COLMAP、FILTER、WHERE、SQLEXEC、およびオブジェクト名を取得するその他のオプションで使用する場合、Oracle GoldenGate は二重引用符で囲まれた入力文字列をリテラルとして処理します。オブジェクトの選択およびマッピングの場合は、USEANSISQLQUOTES を指定しなくても、Oracle GoldenGate は二重引用符で囲まれた文字列をオブジェクト名として認識します。

たとえば、文字列長を返す @STRLEN 変換ファンクションの動作を考えてみます。

- デフォルト (USEANSISQLQUOTES を指定しない) では、Oracle GoldenGate は二重引用符で囲まれた "ABC" をリテラルとして解釈するため、@STRLEN は、値 3 を返します。

  ```
  COLMAP ( TGT1 = @STRLEN("ABC") )
  ```

- USEANSISQLQUOTES を指定すると、Oracle GoldenGate は二重引用符で囲まれた "ABC" を列名として解釈するため、@STRLEN は、列 "ABC" がどのような*値*でも、その長さを返します。

  ```
  COLMAP ( TGT1 = @STRLEN("ABC") )
  ```

- USEANSISQLQUOTES を指定すると、Oracle GoldenGate は一重引用符で囲まれた 'ABC' をリテラルとして解釈するため、@STRLEN は 3 を返します。

  ```
  COLMAP ( TGT1 = @STRLEN('ABC') )
  ```

USEANSISQLQUOTES は、TABLE および MAP パラメータの次のオプションをサポートします。これらのオプションは、リテラル文字列、列名、列変換ファンクション、またはこれらの組合せを入力として取得できます。

- リテラル文字列または列名を引数として取得する変換ファンクション
- COLMAP
- EVENTACTIONS
- FILTER
- SQLEXEC (TABLE または MAP、および単独の SQLEXEC 内)
- TOKENS
- WHERE

使用すると、USEANSISQLQUOTES は、ローカル Oracle GoldenGate インスタンス内のすべての TABLE および MAP 文に影響します。

デフォルト 無効 : 二重引用符はリテラルを示します。

構文 USEANSISQLQUOTES

USEIPV6

適用対象 GLOBALS

USEIPV6 パラメータでは、Oracle GoldenGate に、TCP/IP 接続用に Internet Protocol version 6 (IPv6) を使用させます。デフォルトでは、Oracle GoldenGate はデュアルスタック・モードの IPv6 を使用しますが、IPv4 に戻ってから IPv6 を使用します。USEIPV6 は、IPv4 へのフォールバック・ステップを除外します。ソケット選択の順序は次のようになります。

- IPv6 デュアルスタック
- IPv6

USEIPV6 を使用する場合、Oracle GoldenGate が動作するネットワーク全体が、IPv6 と互換性がある必要があります。

デフォルト 無効

構文 USEIPV6

USERID

適用対象 Manager、Extract、Replicat、DEFGEN

USERID パラメータでは、データベースにログインするときに使用する Oracle GoldenGate プロセスの認証タイプの指定、およびパスワード暗号化情報の指定を行います。Oracle GoldenGate の暗号化の詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』のセキュリティ・ガイドラインを参照してください。

USERID は、パラメータ・ファイルのどの TABLE エントリよりも先に指定します。

USERID を使用するケースと使用方法

USERID は常に必要なわけではなく、PASSWORD は USERID が必要なときに常に必要なわけではありません。データベース認証の構成方法に応じては、USERID の使用のみで十分な場合、SOURCEDB または TARGETDB パラメータのみでも十分な場合があります。

SOURCEDB および TARGETDB を参照してください。

データベース・タイプ別の USERID の要件

注意 **USERID で指定するユーザーに必要な権限の詳細は、お使いのデータベースの Oracle GoldenGate インストール・ガイドを参照してください。**

DB2 for LUW

USERID と PASSWORD は、SOURCEDB または TARGETDB とともに、データベース認証を使用して DB2 LUW データベースに接続するすべての Oracle GoldenGate プロセスで使用します。データベースがオペレーティングシステム・レベルでの認証を許可するように構成されている場合は、USERID および PASSWORD を省略できます (SOURCEDB または TARGETDB のみを使用できます)。この場合は、オペレーティング・システムのユーザーが『Oracle GoldenGate *DB2 LUW インストレーションおよびセットアップ・ガイド*』で説明されている適切な権限を持っている必要があります。

DB2 for z/OS データベース

Oracle GoldenGate プロセスに割り当てられているユーザーが、プロセスを正常に機能させるために必要な DB2 権限を持っていない場合に、USERID を PASSWORD とともに使用します。

MySQL

USERID は、PASSWORD とともに、MySQL データベースに接続するすべての Oracle GoldenGate プロセスで使用します。

Oracle

USERID は、Oracle データベースに接続する Oracle GoldenGate プロセスで使用します。

- オペレーティング・システムのログインを使用するには、USERID を / 引数とともに使用します。
- データベース・ユーザー名およびパスワードを使用するには、USERID を PASSWORD とともに使用します。
- オプションで、ログインするユーザーを sysdba として指定できます。

(*Oracle Enterprise Edition リリース 10.2 以降*) Extract が LOGRETENTION を使用するように構成されている場合、USERID ユーザーには特別なデータベース権限が必要です。 これらの権限は、Oracle GoldenGate のインストール時に付与されている可能性があります。システム要件については、『Oracle GoldenGate *Oracle インストレーションおよびセットアップ・ガイド*』を参照してください。LOGRETENTION の詳細は、386 ページの「TRANLOGOPTIONS」を参照してください。

(*Oracle Standard または Enterprise Edition 11.2.0.2 以降*) 統合キャプチャ用に構成されている Extract グループに USERID を使用するには、次に該当する必要があります。

- ユーザーは、dbms_goldengate_auth.grant_admin_privilege プロシージャで付与された権限を持っている必要があります。
- ユーザーは、この USERID に関連付けられている Extract グループに、DBLOGIN、および REGISTER EXTRACT または UNREGISTER EXTRACT を発行するユーザーである必要があります。

SQL/MX

- ソース SQL/MX データベースに接続する Oracle GoldenGate プロセスでは、USERID を PASSWORD なしで使用して、デフォルト・スキーマを指定します。また、SOURCEDB パラメータを使用して、カタログ名を指定します。
- ターゲット SQL/MX データベースに接続する Oracle GoldenGate プロセスでは、USERID を PASSWORD とともに使用します。また、TARGETDB パラメータを使用して、ターゲット ODBC データ・ソースを指定します。

注意　　パスワード暗号化は、SQL/MX に対してはサポートされていません。

SQL Server

Oracle GoldenGate プロセスによって使用される ODBC データソース接続がデータベース認証を提供するように構成されている場合は、USERID を PASSWORD とともに使用します。USERID には、このプロセス、または**システム管理者**か**サーバー管理者**固有サーバー・ロールのアカウントの任意のメンバーに割り当てられている特定のログインを使用できます。

- ソース SQL Server システムでは、SOURCEDB パラメータも使用してソース ODBC データ・ソースを指定します。
- ターゲット SQL Server システムでは、TARGETDB パラメータも使用してターゲット ODBC データ・ソースを指定します。

Sybase

USERID および PASSWORD は、Sybase データベースに接続するすべての Oracle GoldenGate プロセスで使用します。

Teradata

USERID と PASSWORD は、Teradata データベースに接続するすべての Oracle GoldenGate プロセスで使用します。

- ソース Teradata システムでは、SOURCEDB パラメータも使用してソース ODBC データ・ソースを指定します。
- ターゲット Teradata システムでは、TARGETDB パラメータも使用してターゲット ODBC データ・ソースを指定します。

TimesTen

Replicat によって使用される ODBC データ ソース接続がデータベース認証を提供するように構成されている場合は、USERID を PASSWORD とともに使用します。また、TARGETDB パラメータを使用して、ターゲット ODBC データ・ソースを指定します。

デフォルト なし

構文

```
USERID {/ | <user id>}[, PASSWORD <password>]
[<algorithm> ENCRYPTKEY {<keyname> | DEFAULT}] [SYSDBA]
```

引数	説明

/	(Oracle) データベース・ユーザー・ログインではなく、Oracle 用のオペレーティングシステム・ログインを使用するように Oracle GoldenGate に指示します。この引数は、データベースによってオペレーティングシステム・レベルでの認証が許可されている場合にのみ使用します。データベースレベルの認証をバイパスすることにより、アプリケーションのパスワードが頻繁に変更される場合に、Oracle GoldenGate パラメータ・ファイルを更新する必要がなくなります。
	このオプションを使用するには、Oracle OS_AUTHENT_PREFIX 初期化パラメータとの関連で、正しいユーザー名がデータベースに存在している必要があります。OS_AUTHENT_PREFIX で指定されている値は、ユーザーのオペレーティング・システム・アカウント名の先頭に追加され、データベース名と比較されます。この 2 つの名前は一致する必要があります。
	OS_AUTHENT_PREFIX が " " (NULL 文字列) に設定されている場合は、ユーザー名を "identified externally" として作成する必要があります。
	
	たとえば、OS ユーザー名が "ogg" の場合は、次のようにしてデータベース・ユーザーを作成します。
	`CREATE USER ogg IDENTIFIED EXTERNALLY;`
	OS_AUTHENT_PREFIX が OPS$ または別の文字列に設定されている場合は、ユーザー名は次のフォーマットで作成する必要があります。
	`<OS_AUTHENT_PREFIX_value><OS_user_name>`
	たとえば、OS ユーザー名が "ogg" の場合は、次のようにしてデータベース・ユーザーを作成します。
	`CREATE USER ops$ogg IDENTIFIED BY oggpassword;`

引数	説明
<user id>	データベース・ユーザー名、スキーマ (SQL/MX) または SQL*Net 接続文字列 (Oracle) を指定します。
<password>	データベース・ユーザーのパスワードを指定するためにデータベース認証が必要な場合に使用します。 パスワードが ENCRYPT PASSWORD コマンドによって暗号化されている場合は、暗号化されたパスワードを指定します。それ以外の場合は、クリアテキストのパスワードを使用します。パスワードに大文字と小文字の区別がある場合は、そのとおりに入力します。 ユーザー ID またはパスワードのいずれかが変更されると、必要に応じて、パスワードの再暗号化など、Oracle GoldenGate パラメータ・ファイルの変更を行う必要があります。
<algorithm>	ENCRYPT PASSWORD でパスワードの暗号化に使用した暗号化アルゴリズムを指定します。次のいずれかになります。 AES128 AES192 AES256 BLOWFISH
ENCRYPTKEY {<keyname> | DEFAULT}	ENCRYPT PASSWORD で指定した暗号化鍵を指定します。 ◆ ENCRYPTKEY <keyname> は、ENCKEYS 参照ファイル内のユーザー作成の暗号化鍵の論理名を指定します。ENCRYPT PASSWORD が KEYNAME <keyname> オプションとともに使用された場合に使用します。 ◆ ENCRYPTKEY DEFAULT は、Oracle GoldenGate に、ランダム鍵を使用するように指示します。ENCRYPT PASSWORD が KEYNAME DEFAULT オプションとともに使用された場合に使用します。
SYSDBA	(Oracle) ユーザーが sysdba としてログインするように指定します。

例 1
```
USERID /
```

例 2
```
USERID ogg
```

例 3
```
USERID ogg@ora1.ora, &
PASSWORD AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC AES128, &
ENCRYPTKEY securekey1
```

例 4
```
USERID ogg, PASSWORD AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC &
AES128, ENCRYPTKEY securekey1
```

例 5
```
USERID ogg, PASSWORD AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC &
BLOWFISH, ENCRYPTKEY DEFAULT
```

例 6
```
USERID ogg, &
PASSWORD AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC AES128, &
ENCRYPTKEY securekey1 SYSDBA
```

VAM

適用対象 Extract

VAM パラメータでは、Vendor Access Module (VAM) を使用して Extract プロセスのためのデータ取得ファンクションを実行し、Extract API にデータを送信することを指定します。このパラメータは、VAM API への必須入力を指定します。

デフォルト なし

構文 `VAM <library name>, PARAMS ("<param>" [, "<param>"] [, ...])`

引数	説明
`<library name>`	データベース・ベンダーによって Windows DLL または UNIX 共有オブジェクトとして提供されているライブラリ名。このライブラリが Oracle GoldenGate ディレクトリ以外のディレクトリに存在する場合は、完全パス名を使用します。(注意 : Teradata は、このライブラリを *Teradata Access Module (TAM)* と呼びます。) このプログラム (ライブラリ) は、Oracle GoldenGate VAM API と通信します。

`PARAMS <param>`	◆ PARAMS は必須のキーワードです。
	◆ <param> は、Oracle GoldenGate API に渡される、引用符で囲まれた任意のパラメータです。次のパラメータ・オプションを参照してください。
	
	SQL/MX データベース用 **PARAMS オプション :**
	
	ARLIBError <error>, <action>
	
	(オプション)VAM による TMFARLIB エラーの処理方法を指定します。
	◆ <error> は、ARLIB エラー番号です。
	◆ <action> には、ABEND | WARN | IGNORE を指定できます。
	デフォルトは ABEND です。エラー -1000 および -2000 は、他にアクションが指定されていても、常に ABEND になります。
	例 :
	`Vam Params (arliberror (-16,-14), Warn)`
	`Vam Params (arliberror -2000, Abend)`
	`Vam Params (arliberror -1000, Abend)`
	
	AREErrorReportInterval <seconds>
	
	同一の TMFARLIB エラーが Extract にレポートで戻される間隔 (秒) を設定します。これにより、蓄積される各エラー・タイプのメッセージ量が削減されます。
	◆ <seconds> は、ゼロ以上である必要があります。
	◆ デフォルトは 60 秒です。

引数	説明
	Teradata データベース用 **PARAMS オプション :** inifile, <ini file>, callbackLib, extract.exe Teradata TAM に必須のパラメータです。 ◆ inifile は、次のパラメータに TAM 初期化ファイルが指定されることを示します。 ◆ <ini file> は、TAM 初期化ファイル名です。このファイルが Extract プログラムのインストール先と同じディレクトリに存在する場合を除き、完全修飾パス名を指定します。 ◆ callbackLib は、次のパラメータに TAM とのインタフェースになるプログラムが指定されることを示します。このパラメータは、大 / 小文字が区別され、ここに示したとおりに正確に入力する必要があります。 ◆ extract.exe は Extract プログラムで、TAM にとってコールバック・プログラムです。

例 `VAM tam.dll, PARAMS (inifile, tam.ini, callbackLib, extract.exe)`

VARWIDTHNCHAR | NOVARWIDTHNCHAR

適用対象 Extract、Replicat、DEFGEN

VARWIDTHNCHAR および NOVARWIDTHNCHAR パラメータでは、NCHAR データのトレイルへの書込み方法と Replicat による解釈方法を制御します。

- VARWIDTHNCHAR を使用すると、NCHAR、NVARCHAR2 または NCLOB 文字セットは、可変長文字セット (UTF-8) として処理されます。
- NOVARWIDTHNCHAR を使用すると、NCHAR、NVARCHAR2 または NCLOB 文字セットは、UTF-16 として処理されます。
- いずれのオプションも指定しない場合は、NCHAR、NVARCHAR2 または NCLOB 文字セットの処理方法を決定するために、データベースの NLS_NCHAR_CHARACTERSET プロパティ値が使用されます。

デフォルト データベースから NLS_NCHAR_CHARACTERSET プロパティを使用する

構文 `VARWIDTHNCHAR | NOVARWIDTHNCHAR`

WARNLONGTRANS

適用対象 Extract

WARNLONGTRANS パラメータでは、Extract によってトランザクションが長時間におよんでいることの警告メッセージが生成されるまでの、トランザクションのオープン時間を指定します。また WARNLONGTRANS も使用して、Oracle GoldenGate が長時間におよぶトランザクションをチェックする頻度を制御します。

WARNLONGTRANS を指定すると、Oracle GoldenGate は指定されたしきい値を満たすトランザクションを確認し、検出した最初のトランザクションを Oracle GoldenGate エラー・ログ、Extract レポート・ファイル、およびシステム・ログにレポートします。デフォルトでは、Oracle GoldenGate はこの確認を 5 分ごとに繰り返します。

オープンしているトランザクションのリストのオンデマンド表示、トランザクション詳細のファイルへの出力、これらのトランザクションの中止またはトレイルへの強制的な書込みの方法の詳細は、SEND EXTRACT コマンド (36 ページ) を参照してください。

このパラメータは、MySQL、Oracle、SQL Server および Sybase に有効です。

MySQL の場合、MySQL ログにはコミット済トランザクションのみが含まれるため、WARNLONGTRANS では、オープンしているトランザクションは 1 つ表示されるか、何も表示されません。ただし、WARNLONGTRANS は、Extract によって処理中のままのトランザクションの警告に使用することができ、ログの蓄積およびスケジュールのアーカイブの観点から長時間かかることがあります。

デフォルト 1 時間 (および別の処理スレッドを使用して 5 分間隔で確認)

構文

```
WARNLONGTRANS <duration><unit>
[, CHECKINTERVAL <interval><unit>]
[, NOUSETHREADS]
[, USELASTREADTIME]
```

引数	説明

`<duration><unit>`	オープンしているトランザクションを長時間におよぶトランザクションとみなすまでの時間を設定します。デフォルトは 1 時間です。
	◆ <duration> には、時間の長さを整数で指定します。
	◆ <unit> には、second、minutes、hours または days を完全なスペリングまたは短縮形式で指定します。
	`S|SEC|SECS|SECOND|SECONDS`
	`M|MIN|MINS|MINUTE|MINUTES`
	`H|HOUR|HOURS`
	`D|DAY|DAYS`
	<duration> および <unit> の間にスペースを入れないでください。

`CHECKINTERVAL <interval><unit>`	Oracle GoldenGate が WARNLONGTRANS を満たすトランザクションを確認し、最も長時間におよぶトランザクションレポートする間隔を設定します。
	◆ <interval> には、確認する時間間隔の長さを整数で指定します。
	◆ <unit> には、second、minutes、hours または days を完全なスペリングまたは短縮形式で指定します。
	`S|SEC|SECS|SECOND|SECONDS`
	`M|MIN|MINS|MINUTE|MINUTES`
	`H|HOUR|HOURS`
	`D|DAY|DAYS`
	<interval> および <unit> の間にスペースを入れないでください。デフォルトは最小値の 5 分です。

`NOUSETHREADS`	メイン・プロセス・スレッドによってモニタリングを実施することを指定します。デフォルトでは、パフォーマンスの理由で、モニタリングは個々のスレッドによって実施されます。NOUSETHREADS は、システムがマルチスレッドをサポートしていない場合にのみ使用するようにしてください。

引数	説明
USELASTREADTIME	(Oracle のみ) トランザクションが長時間におよんでいるかどうかの判断基準として、Extract に常に REDO ログの最後の読取り時間を使用させます。デフォルトでは、Extract は REDO ログから読み取った最後のレコードのタイムスタンプを使用します。このパラメータは、ARCHIVEDLOGONLY オプションを使用して TRANLOGOPTIONS でアーカイブ・ログ専用モードで実行している Extract に適用されます。

例 `WARNLONGTRANS 2h, CHECKINTERVAL 3m`

WARNRATE

適用対象 Replicat

WARNRATE パラメータでは、プロセス・レポートまたはエラー・ログにレポートされるまでの、ターゲット表で許容される SQL エラー数のしきい値を設定します。エラーは警告としてレポートされます。お使いの環境で、多数の SQL エラーを許容できるときは、WARNRATE の値を増やすことによってこれらのファイルのサイズを最小限に抑えられます。

デフォルト 100 エラー

構文 `WARNRATE <num errors>`

引数	説明
<num errors>	警告発行のしきい値にする SQL エラー数。

例 `WARNRATE 1000`

WILDCARDRESOLVE

適用対象 Extract および Replicat

WILDCARDRESOLVE パラメータでは、TABLE、SEQUENCE、または MAP 文でワイルドカードで指定されている表を処理するルールを変更します。WILDCARDRESOLVE は、パラメータ・ファイル内の関連する TABLE、SEQUENCE、または MAP 文よりも先に指定する必要があります。

ターゲット・オブジェクトは、ワイルドカードの解決を試みるときに、ターゲット・データベース上に存在している必要があります。ターゲット・オブジェクトが存在しない場合、Replicat は異常終了します。

デフォルト `DYNAMIC`

構文 `WILDCARDRESOLVE {DYNAMIC | IMMEDIATE}`

引数	説明
DYNAMIC	ワイルドカード定義に一致するソース・オブジェクトは、ワイルドカード・ルールに一致するたびに解決されます。これはデフォルトです。 SOURCEISTABLE または GENLOADFILES を指定している場合は、このオプションを使用しないでください。これらのパラメータを使用すると、WILDCARDRESOLVE は常に暗黙的に IMMEDIATE に設定されます。 これは、Teradata に必須の引数です。 SEQUENCE パラメータを指定し、ワイルドカードを使用して Oracle 順序をレプリケートするときは、DYNAMIC を使用する必要があります。 デフォルトの動作が要求される場合、明示的な WILDCARDRESOLVE DYNAMIC パラメータ文はオプションです。ただし指定しておくと、パラメータ・ファイルを確認する担当者が、使用されている方法を明確に把握できます。
IMMEDIATE	ワイルドカード定義に一致するソース・オブジェクトは、起動時に処理されます。このオプションは、Teradata ではサポートされていません。SOURCEISTABLE を使用する場合は、これが強制的にデフォルトになります。 このオプションは、Oracle インターバル・パーティション機能をサポートしていません。Oracle GoldenGate が新しいパーティションを見つけられるように、動的な解決を行う必要があります。

例　次に、起動時にワイルドカードを解決する例を示します。

```
WILDCARDRESOLVE IMMEDIATE
TABLE hq.acct_*;
```

第 4 章
Collector パラメータ

この章では、Collector プロセス・パラメータについて説明します。Collector プロセスは、ターゲット・システム上で動作し、受信データを受け取ってトレイルに書き込みます。

動的 Collector

通常 Oracle GoldenGate ユーザーは、この Collector プロセスとやり取りを行いません。これは Manager プロセスによって動的に起動されます。これが*動的 Collector* と呼ばれます。

静的 Collector

静的 Collector は、次に示す構文と入力パラメータを使用してコマンドラインで SERVER プログラムを実行することにより、手動で実行できます。

構文

```
server <parameter> [<parameter>] [...]
```

Collector パラメータは、大 / 小文字が区別され、先頭にダッシュを付ける必要があります。

表 42　Collector パラメータ

パラメータ	説明
`-cp <checkpoint file>`	別名 Extract グループ用に Collector で保持するチェックポイント・ファイルの名前を指定します。<checkpoint file> は、別名 Extract グループに関連付けられているパッシブ Extract グループの名前です。 チェックポイント・ファイルは、パッシブ Extract グループが実行中かどうかを確認するために使用されます。チェックポイント・ファイルが Collector(server プログラム) によってロックされている場合は実行中です。-h および -p パラメータとともに使用する必要があります。 パッシブおよび別名 Extract グループの使用の詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。
`-d <definitions file>`	エクスポートされた表定義を含む、DEFGEN ユーティリティによって生成されたローカル・ファイル。
`-E`	受信ヘッダーおよびデータを、ASCII から EBCDIC フォーマットに変換します。デフォルトでは、Oracle GoldenGate はデータ変換を行いません。
`-e <version error type> <action>`	Collector にカスタマイズした方法で特定のフォーマット・エラーにレスポンスするように指示します。たいていの場合はデフォルト値で問題ありません。複数のエラー・タイプを指定するときは、-e を複数回使用します。次に例を示します。 `-e OLD CONTINUE -e NEW DISCARD.` <version error type> には、次のいずれかを指定できます。

表 42　Collector パラメータ (続き)

パラメータ	説明
	NEW 想定よりも多くのデータ (現在の定義よりも多くの列) を含むレコードを確認します。Collector プロセスは、更新されたソース表が必要になることがあります (つまり DEFGEN をもう一度実行する必要があります)。デフォルトのアクションは ABEND です。 OLD 想定より少ないデータを含むレコードを確認します。これは、通常はレコードに含まれる列が表の現在の定義よりも少ない状況で、正常な状態と考えられます。デフォルトのアクションは CONTINUE です。 OUTOFSYNC 提供されている定義に従って変換できないレコードを確認します。デフォルトのアクションは ABEND です。 <action> には、次のいずれかを指定できます。 ABEND レコードを破棄し、Extract プロセスに即座に終了するよう指示します。 CONTINUE 検出された変換エラーにかかわらず、(可能な場合) レコードを処理します。 DISCARD (破棄ファイルが -x で指定されている場合)、レコードを破棄ファイルに出力します。Collector は、エラー・ファイルに最初に破棄されたレコードの警告を送信し、レコードの処理を継続します。
-ENCRYPT <encrypt type>	Extract パラメータ・ファイルの RMTHOST パラメータで指定され、Extract プロセスから渡される暗号化のタイプ。 次の値が有効です。 ◆ NONE ◆ BLOWFISH BLOWFISH を使用するときは、-KEYNAME オプションも指定します。Oracle GoldenGate のセキュリティの詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。

表 42　Collector パラメータ (続き)

パラメータ	説明
-f	Extract プロセスに成功ステータスを返す前に、常にファイルへの書込みをディスクにフラッシュさせます。操作のたびにディスクにフラッシュするよりも効率的なため、デフォルトではファイル・システムは I/O をバッファします。I/O 成功の確認後、バッファがディスクにフラッシュされる前にシステムが停止し、データが損失するという小さなリスクよりも、通常はパフォーマンスのメリットのほうが重要です。このリスクが許容できない場合は、Oracle GoldenGate のパフォーマンスが低下する可能性を理解した上で、-f を使用してください。
-g	2GB より大きなファイルをサポートします (Solaris のみ)。
-h <host name or IP address>	ソース・システムの名前または IP アドレスを指定します。このオプションは、ソース上で**パッシブ**・モードで実行されている Extract と関連付けられているターゲット上の別名 Extract を使用するときに使用します。これにより、Collector を*接続モード*で動作させます。このモードでは、ソース Extract からの接続リクエストを待機するかわりに、ソース Extract への TCP/IP 接続を開始します。-p Collector オプションとともに使用する必要があります。 パッシブおよび別名 Extract グループの使用の詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。
-k	サービス提供先の Extract プロセスが接続を切断したときに終了するように Collector に指示します。このオプションは、Collector プロセスを起動するときに Manager プロセスによって使用されます。
-KEYNAME <name>	ローカル ENCKEYS 参照ファイルで定義されている鍵名を指定します。-ENCRYPT に BLOWFISH を指定しているときに使用します。
-l <file name>	指定したファイルに出力を記録します。完全修飾名を使用します。
-m	割り当てる最大ログ・ファイル数を指定します。
-P <parameter file>	Collector パラメータを含むローカル・ファイル。このファイルのパラメータは、Extract プロセスから送信されるパラメータより優先されます。
-p <port number>	次のように指定する TCP/IP ポート番号 : ◆ 通常の Extract または通常のデータ・ポンプの場合 : Collector プロセスが Extract からの接続リクエストをリスニングするポート。 ◆ **パッシブ**・モードで実行されている Extract またはデータ・ポンプの場合 : Extract またはデータ・ポンプが Collector からの接続リクエストをリスニングするポート。この場合は、-h <host> パラメータとともに使用する必要があります。 デフォルトはポート 7819 です。 パッシブおよび別名 Extract グループの使用の詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。

表 42　Collector パラメータ (続き)

パラメータ	説明
-R <alternate value>	無効な数値 ASCII データを代替値に置換します。デフォルトでは 0 に置換します。代替値は次のいずれかです。 <number> 代替値を指定します。 NULL ターゲット列が NULL 値を受け付ける場合は NULL を指定し、それ以外の場合はゼロに置換します。 UNPRINTABLE 出力不可能なデータを含むすべての列を拒否します。プロセスは停止し、不正な値をレポートします。 NONE 数値データの置換を行いません。Oracle GoldenGate はそのままのデータのレプリケートを試みます。
-x <discard file>	フォーマットできないレコードを出力する破棄ファイルを指定します。完全修飾名を使用します。

第 5 章
列変換ファンクション

Oracle GoldenGate の列変換ファンクションを使用すると、ソース値をターゲット列に適切なフォーマットに変換できます。これらのファンクションでは、数字と文字の操作、テストの実行、パラメータ値の抽出、環境情報のリターンなどを行うことができます。

列変換ファンクションの概要

この概要は、Oracle GoldenGate ファンクションで実行できる処理タイプ別に構成されています。アルファベット順のリファレンスは 432 ページから記載しています。

表 43　テストの実行

ファンクション	説明
CASE	一連の値テストに応じて値を選択します。
EVAL	一連の独立テストに基づいて値を選択します。
IF	条件文から TRUE または FALSE のどちらを返されるかによって、2 つの値から 1 つを選択します。

表 44　行方不明列の処理

ファンクション	説明
COLSTAT	列が MISSING、NULL、または INVALID であることのインジケータを返します。
COLTEST	列が PRESENT、MISSING、NULL、または INVALID かどうかをテストする条件計算を実行します。

表 45　日付の操作

ファンクション	説明
DATE	ソース列に渡されたフォーマットに基づいて日付と時刻を返します。
DATEDIFF	2 つの日付または時刻の差を返します。
DATENOW	現在の日付と時刻を返します。

表 46　算術計算を実行します

ファンクション	説明
COMPUTE	算術式の結果を返します。

表 47　文字列の操作

ファンクション	説明
NUMBIN	バイナリ文字列を数字に変換します。
NUMSTR	文字列を数字に変換します。
STRCAT	1 つ以上の文字列を連結します。
STRCMP	2 つの文字列を比較します。
STREXT	文字列の一部を抽出します。
STREQ	2 つの文字列が等しいかどうかを確認します。
STRFIND	文字列内の文字列を検索します。
STRLEN	文字列の長さを返します。
STRLTRIM	先行する空白を切り捨てます。
STRNCAT	1 つ以上の文字列を最大長まで連結します。
STRNCMP	指定された文字数に基づいて 2 つの文字列を比較します。
STRNUM	数字を文字列に変換します。
STRRTRIM	末尾の空白を切り捨てます。
STRSUB	文字列を別の文字列に置換します。
STRTRIM	先行の空白と末尾の空白を切り捨てます。
STRUP	文字列を大文字に変更します。
VALONEOF	文字列または文字列用の列を値のリストを比較します。

表 48　その他のファンクション

ファンクション	説明
BINARY	ソース列が文字用の列として定義されているときに、ソース列のバイナリ・データをターゲット列でバイナリ・データとして保持します。
BINTOHEX	バイナリ文字列を 16 進数文字列に変換します。
GETENV	環境情報を返します。
GETVAL	FILTER または COLMAP 句の入力としてストアド・プロシージャからパラメータを抽出します。
HEXTOBIN	16 進数文字列をバイナリ文字列に変換します。
HIGHVAL | LOWVAL	値を最大値または最小値までに抑止します。
RANGE	並列処理のために行を複数のデータ・グループに分割します。
TOKEN	トレイル・レコード・ヘッダーからトークン・データを取得します。

BINARY

@BINARY ファンクションは、列変換ファンクションに参照されるソース列が、文字用の列として定義されているものの、ターゲットではバイナリのまま保持する必要があるバイナリ・データを含んでいるときに使用します。デフォルトでは、文字用の列のバイナリ・データは (必要な場合に)ASCII に変換され、NULL で終了された文字列とみなされます。@BINARY ファンクションは、任意のバイナリ・データをターゲット列にコピーします。

構文

```
@BINARY(<column name>)
```

引数	説明
<column name>	データのコピー先のターゲット列名。

例

次に、@BINARY を使用してソース列 ACCT_CREATE_DATE からターゲット列 ACCT_CHIEF_COMPLAINT にデータをコピーする方法の例を示します。

```
ACCT_CHIEF_COMPLAINT =
@IF ( @NUMBIN (ACCT_CREATE_DATE ) < 48633, "xxxxxx",
@BINARY(ACCT_CHIEF_COMPLAINT))
```

BINTOHEX

@BINTOHEX ファンクションでは、指定されたバイナリ・データを対応する 16 進数に変換します。

構文

```
@BINTOHEX(<data>)
```

引数	説明
<data>	ソース列名、式、または引用符で囲まれたリテラル文字列。

例 @BINTOHEX("12345") は "3132333435" に変換されます。

CASE

@CASE ファンクションでは、一連の値テストに応じて値を 1 つ選択します。@CASE でテストできるケース数に制限はありません。ケース数が多い場合は、パフォーマンスを最適化するために最も頻繁に検出される条件を最初にリストします。

Oracle GoldenGate では、このファンクションで、Unicode、および Microsoft Windows、UNIX、Linux オペレーティング・システムのネイティブ・エンコーディングの文字列を格納する列内の文字を表すために、エスケープ・シーケンスの使用をサポートしています。引数が Unicode で指定される場合、ターゲット列は SQL Unicode データ型である必要があります。

このファンクションは、NCHAR または NVARCHAR データ型をサポートしていません。

構文

```
@CASE (<value>, <test value1>, <test result1>
[, <test value2>, <test result2>] [, ...]
[, <default result>]
```

引数	説明
<value>	テストする値 (列名など)。リテラルは引用符で囲みます。
<test value>	<value> に有効な結果。リテラルは引用符で囲みます。
<test result>	<test value> の値に基づいて返される値。リテラルは引用符で囲みます。
<default result>	<value> がどの <test value> 値にも該当しない場合に返されるデフォルト値。リテラルは引用符で囲みます。

例 1 次の例では、PRODUCT_CODE が "CAR" の場合は "A car" が返され、PRODUCT_CODE が "TRUCK" の場合は "A truck" が返されます。 PRODUCT_CODE が最初の 2 つのケースのいずれにも該当しない場合は、デフォルト値が指定されていないため、FIELD_MISSING インジケーションが返されます。

```
@CASE (PRODUCT_CODE, "CAR", "A car", "TRUCK", "A truck")
```

例 2 次の例は上記の例に似ていますが、ここではデフォルト値が指定されています。PRODUCT_CODE が "CAR" にも "TRUCK" にも該当しない場合、このファンクションは "A vehicle" を返します。

```
@CASE (PRODUCT_CODE, "CAR", "A car", "TRUCK", "A truck", "A vehicle")
```

COLSTAT

@COLSTAT ファンクションでは、列が行方不明、NULL、または無効であることのインジケータを Extract または Replicat に返します。このインジケータは、別の変換ファンクションを使用する大規模な操作式の一部として使用できます。

構文

```
@COLSTAT ({MISSING | NULL | INVALID})
```

例 1

次の例では、ターゲット列 ITEM に NULL を返します。

```
ITEM = @COLSTAT (NULL)
```

例 2

次の @IF 演算は、@COLSTAT を使用して、PRICE および QUANTITY がゼロより小さい場合にターゲット列に NULL を返します。

```
ORDER_TOTAL = PRICE * QUANTITY, @IF (PRICE < 0 AND QUANTITY < 0, @COLSTAT(NULL))
```

COLTEST

@COLTEST ファンクションでは、1 つ以上の列条件を使用したテストによる条件演算を有効にします。条件が満たされた場合、@COLTEST は TRUE を返します。条件演算を実行するには、@IF ファンクションを使用します。

構文

```
@COLTEST (<source column>, <test item> [, <test item>] [, ...])
```

引数	説明	
<source column>	ソース列名。	
<test item>	次の値が有効です。	
	PRESENT	列がソース・レコードに存在し、NULL でないことを示します。データベースが、変更されない NULL と同じはない列の値を記録しない場合、列の値は MISSING になることがあります。
	NULL	列がソース・レコードに存在し、NULL であることを示します。
	MISSING	列がソース・レコードに存在しないこと示します。
	INVALID	列がソース・レコードに存在するが無効なデータを含んでいることを示します。

例 1

次の例では、@IF を使用して、ソース・レコードに BASE_SALARY 列が存在し (および NULL でなく)、かつ 250000 より大きい場合に、HIGH_SALARY 列に値をマップします。そうでない場合は NULL が返されます。

```
HIGH_SALARY =
@IF (@COLTEST (BASE_SALARY, PRESENT) AND
BASE_SALARY > 250000,
BASE_SALARY, @COLSTAT (NULL))
```

例 2 次の例では、AMT 列が MISSING または INVALID の場合には 0 が返され、それ以外の場合には AMT の値が返されます。

```
AMOUNT = @IF (@COLTEST (AMT, MISSING, INVALID), 0, AMT)
```

COMPUTE

@COMPUTE ファンクションでは、算術式の値をターゲット列に返します。ファンクションから返される値は、文字列形式です。

算術式の値を別の Oracle GoldenGate ファンクションに返すときは、次の例のように @COMPUTE 句を省略できます。

```
@STRNUM ((AMOUNT1 + AMOUNT2), LEFT)
```

上記は、次と同じ結果を返します。

```
@STRNUM ((@COMPUTE (AMOUNT1 + AMOUNT2), LEFT)
```

算術式では、次の要素を組み合せることができます。

- 数字
- 数字を含む列名
- 数字を返すファンクション
- 算術演算子 :

  +(加算)
  -(減算)
  *(乗算)
  /(除算)
  \ (余り)

- 比較演算子 :

  > (より大きい)
  >=(以上)
  <(より少ない)
  <=(以下)
  =(等しい)
  <>(等しくない)

  比較から導出した結果はゼロ (FALSE を示す) またはゼロ以外 (TRUE を示す) になります。

- カッコ (式の結果をグループ化)
- 結合演算子 AND、OR。Oracle GoldenGate は、結合式の必要部分のみを評価します。文が FALSE になると、式の残りの部分は無視されます。この動作は、行方不明または NULL の可能性があるフィールドを評価するときに役立ちます。たとえば、COL1 の値が 25 で、COL2 の値が 10 の場合は、次のようになります。

  `@COMPUTE (COL1 > 0 AND COL2 < 3)` は 0 を返します。
  `@COMPUTE (COL1 < 0 AND COL2 < 3)` は 0 を返します。COL2 < 3 は評価されません。
  `@COMPUTE ((COL1 + COL2)/5)` は 7 を返します。

構文

```
@COMPUTE(<expression>)
```

引数	説明
<expression>	有効な算術式。

例 1 AMOUNT_TOTAL = @COMPUTE (AMT + AMT2)

例 2 AMOUNT_TOTAL = @IF (AMT >= 0, AMT * 100, 0)

例 3 ANNUAL_SALARY = @COMPUTE (MONTHLY_SALARY * 12)

DATE

@DATE ファンクションでは、ソース列に渡された形式に基づいて、ターゲット列に様々な形式で日付と時間を返します。@DATE は、実質上すべてのタイプの入力を有効な SQL 日付に変換します。@DATE は、日付列の一部の抽出や、日付に基づいた数値タイムスタンプの計算にも使用できます。

構文

```
@DATE ("<output descriptor>", "<input descriptor>", <source col>
[, "<input descriptor>", <source col>] [, ...])
```

引数	説明
"<output descriptor>"	ターゲット列が必要とする日付ディスクリプタおよびオプションのリテラル値 (空白やコロンなど) を含むターゲット文字列。日付ディスクリプタは、必要に応じて結合できます。436 ページの表 49 の日付ディスクリプタの説明を参照してください。ターゲットの形式ディスクリプタは、date/time/timestamp 形式に適合する必要があります。Oracle GoldenGate は、正しい形式にするために、必要に応じて指定された形式を無効にします。
"<input descriptor>"	日付ディスクリプタおよびオプションのリテラル値 (空白やコロンなど) を含むソース文字列。日付ディスクリプタは、必要に応じて結合できます。次に例を示します。 ディスクリプタ文字列 "YYYYMMDD" は、次に指定される数字または文字用の列に、順番に 4 桁の西暦 (YYYY)、月 (MM) および日 (DD) が含まれていることを示します。 ディスクリプタ文字列 "DD/MM/YY" は、フィールドに日、スラッシュ、月、スラッシュ、西暦の下 2 桁が含まれていることを示します。 日付ディスクリプタの詳細は、表 49 を参照してください。
<source col>	上記の入力を提供するソース列名。

表 49　日付ディスクリプタ

ディスクリプタ	説明	有効
CC	100 年	入力 / 出力
YY	西暦の下 2 桁	入力 / 出力

表 49　日付ディスクリプタ（続き）

ディスクリプタ	説明	有効
YYYY	4 桁の西暦	入力 / 出力
MM	数字の月	入力 / 出力
MMM	英語の月 (APR、OCT など)	入力 / 出力
DD	月単位の日付	入力 / 出力
DDD	年単位の日付 (001、365 など)	入力 / 出力
DOW0	数値で表す曜日 (Sunday = 0)	入力 / 出力
DOW1	数値で表す曜日 (Sunday = 1)	入力 / 出力
DOWA	英語で表す曜日 (SUN、MON、TUE など)	入力 / 出力
HH	時間	入力 / 出力
MI	分	入力 / 出力
SS	秒	入力 / 出力
JTSLCT	すでにローカル時間のユリウス・タイムスタンプに対して、またはユリウス・タイムスタンプに変換するときにローカル時間を保持するために使用します。	入力 / 出力
JTSGMT	ユリウス・タイムスタンプ (JTS と同じ)。	入力 / 出力
JTS	ユリウス・タイムスタンプ。JUL および JTS は、数式で使用できる数字を生成します。単位はマイクロ秒です。Windows のタイムスタンプはミリ秒の粒度を使用するので、Windows マシンでは値にゼロが埋め込まれます。	入力 / 出力
JUL	ユリウス日。JUL および JTS は、数式で使用できる数字を生成します。	入力 / 出力
TTS	NonStop の 48 ビット・タイムスタンプ	入力
PHAMIS	PHAMIS アプリケーション日付形式	入力
FFFFFF	端数 (マイクロ秒まで)	入力 / 出力
STRATUS	STRATUS アプリケーション・タイムスタンプ	入力 / 出力
CDATE	エポック以降の C タイムスタンプ (秒)	入力 / 出力

例 1　下 2 桁の西暦が提供されているインスタンスで、出力で 4 桁の西暦が必要な場合には、複数のオプションを使用して正しい 100 年を取得できます。

- 100 年を次のようにハードコードできます。
  ```
  "CC", 19 or "CC", 20
  ```
- @IF ファンクションを使用して、次のように条件を設定できます。
  ```
  "CC", @IF (YY > 70, 19, 20)
  ```
  これにより、下 2 桁の西暦が 70 より大きい場合に、100 年を 19 に設定し、それ以外の場合は 20 に設定できます。
- システムは100年を自動的に計算できます。下2桁の西暦が50未満の場合、システムは100年を20として算出し、それ以外の場合は 100 年を 19 として算出します。

例 2 次の例では、西暦、月、および日列を日付に変換します。

```
date_col = @DATE ("YYYY-MM-DD", "YY", date1_yy, "MM", date1_mm, "DD", date1_dd)
```

例 3 次の例では、日付と時間を変換し、秒をデフォルトでゼロにします。

```
date_col = @DATE ("YYYY-MM-DD:HH:MI:00", "YYMMDD", date1, "HHMI", time1)
```

例 4 次の例では、YYYYMMDDHHMISS として保持されている数値列を SQL 日付に変換します。

```
datetime_col = @DATE ("YYYY-MM-DD:HH:MI:SS", "YYYYMMDDHHMISS", numeric_date)
```

例 5 次の例では、YYYYMMDDHHMISS として保持されている数値列をユリウス・タイムスタンプに変換します。

```
julian_ts_col = @DATE ("JTS", "YYYYMMDDHHMISS", numeric_date)
```

例 6 次の例では、ユリウス・タイムスタンプ列を 2 つの異なる列 (YYYY-MM-DD:HH:MI:SS 形式の日付時間列と、タイムスタンプのマイクロ秒部分を保持する端数列) に変換します。

```
datetime_col = @DATE ("YYYY-MM-DD:HH:MI:SS", "JTS", jts_field), fraction_col = @DATE
("FFFFFF", "JTS", jts_field)
```

例 7 次の例では、受注が処理された時間を生成します。内部の @DATE 式は、order_taken 列をユリウス・タイムスタンプに変更した後、マイクロ秒に変換された order_minutes 列をこのタイムスタンプに追加します。この式は、新しいユリウス・タイムスタンプとして外部 @DATE 式に戻され、読みやすい日付と時間に再度変換されます。

```
order_filled = @DATE ("YYYY-MM-DD:HH:MI:SS", "JTS", @DATE ("JTS",
"YYMMDDHHMISS", order_taken) + order_minutes * 60 * 1000000)
```

例 8 次の例では、完全な時間の計算を実行します。ソース日付列 (名前 "dt") に 5 時間を足して変換された日時が、ターゲット列 (名前 "dt5") に返されます。また、ソース・タイムスタンプ列 (名前 "ts") に 5 時間を足して変換されたタイムスタンプが、ターゲット列 (名前 "ts5") に返されます。

```
MAP scratch.t4, TARGET scratch.t4_copy,
COLMAP (USEDEFAULTS,
dt5 = @DATE ("YYYY-MM-DD:HH:MI:SS", "JTS",
@COMPUTE (@DATE ("JTS", "YYYY-MM-DD:HH:MI:SS", dt) + 18000000000 ) ),
ts5 = @DATE ("YYYY-MM-DD:HH:MI:SS.FFFFFF", "JTS",
@COMPUTE ( @DATE ("JTS", "YYYY-MM-DD:HH:MI:SS.FFFFFF", ts) + 18000000000 ) )
) ;
```

DATEDIFF

@DATEDIFF ファンクションでは、2 つの日付または日時の差異を日数または秒数で計算します。

構文

```
@DATEDIFF ("difference", "<date>", "<date>")
```

引数	説明
<difference>	指定した日付の差異。次の値が有効です。 DD 差異を日数で計算します。 SS 差異を秒数で計算します。
<date>	引用符で囲んだ YYYY-MM-DD[*HH:MI[:SS]] **形式の文字列** (* はコロン (:) または空白でも可)、または現在の日付を返す引用符なしの @DATENOW ファンクション。

例 1

次の例では、2011 年 1 月 1 日からの日数を計算します。

```
YTD = @DATEDIFF ("DD", "2011-01-01", @DATENOW ())
```

例 2

次の例では、その年の現在までの日数を計算します。((@DATEDIFF は、2011-01-01 に対して 0 を返します)

```
todays_day = @COMPUTE (@DATEDIFF ("DD", "2011-01-01", @DATENOW ()) +1)
```

DATENOW

@DATENOW ファンクションでは、現在の日時を YYYY-MM-DD HH:MI:SS 形式で返します。日時は、夏時間の調整後、ローカル時間で返されます。@DATENOW は引数を取りません。

構文

```
@DATENOW ()
```

DDL

@DDL ファンクションでは、DDL 操作に関する情報を返します。

構文

```
@DDL ({TEXT | OPTYPE | OBJNAME | OBJTYPE | OBJOWNER})
```

引数	説明
OBJNAME	DDL によって影響を受けるオブジェクトの名前を返します。
OBJOWNER	DDL によって影響を受けるオブジェクトの所有者の名前を返します。
OBJTYPE	DDL によって影響を受けるオブジェクトのタイプ (TABLE や INDEX など) を返します。

引数	説明
OPTYPE	DDL の操作タイプ (CREATE や ALTER など) を返します。
TEXT	DDL 文のテキストの最初の 200 文字を返します。

例 次の例では、EVENTACTIONS シェル・コマンド内の @DDL からの出力を使用します。

```
DDL INCLUDE OBJNAME src.t* &
EVENTACTIONS (SHELL ("echo The DDL text is var1> out.txt ", &
VAR var1 = DDL(TEXT));
```

リダイレクトされた出力ファイルには、次のような文字列が含まれることがあります。

```
"The DDL text is CREATE TABLE src.test_tab (col1 int);"
```

EVAL

@EVAL ファンクションでは、一連の独立したテストに基づいて値を選択します。テストできる条件数に制限はありません。ケース数が多い場合は、パフォーマンスを最適化するために最も頻繁に検出される条件を最初にリストします。

Oracle GoldenGate では、このファンクションで、Unicode、および Microsoft Windows、UNIX、Linux オペレーティング・システムのネイティブ・エンコーディングの文字列を格納する列内の文字を表すために、エスケープ・シーケンスの使用をサポートしています。引数が Unicode で指定される場合、ターゲット列は SQL Unicode データ型である必要があります。

構文

```
@EVAL (<condition1>, <result1>
[<condition2>, <result2>] [, ....]
[, <default result>])
```

引数	説明
<condition>	標準の条件演算子を使用する条件テスト。
<result>	条件テストの結果に基づいて返される値または文字列。リテラルは二重引用符で囲みます。
<default result>	どの条件も満たされなかったときに返されるデフォルトの結果。デフォルト結果はオプションです。

例 1 次の例では、AMOUNT 列が 10000 より大きい場合に、"high amount" が返されます。AMOUNT が 5000 より大きい (かつ 10000 以下) の場合 (前の条件が満たされなかった場合) は、"somewhat high" が返されます。いずれの条件も満たさない場合、デフォルト結果が指定されていないため、COLUMN_MISSING インジケーションが返されます。

```
AMOUNT_DESC = @EVAL (AMOUNT > 10000, "high amount", AMOUNT > 5000, "somewhat high" )
```

例 2 次に、上記を変更した例を示します。同じ結果が返されますが、デフォルト値が指定されたため、AMOUNT が 5000 以下の場合には結果 "lower" が返されます。

```
@EVAL (AMOUNT > 10000, "high amount", AMOUNT > 5000, "somewhat high", "lower")
```

GETENV

@GETENV ファンクションでは、Oracle GoldenGate 環境に関する情報を返します。この情報は、次への入力として使用できます。

- ストアド・プロシージャまたは問合せ (SQLEXEC を使用)
- 列マップ (TABLE または MAP の COLMAP オプションを使用)
- ユーザー・トークン (TABLE の TOKENS オプションで定義され、@TOKENS ファンクションによってターゲット列にマッピング済)
- GET_ENV_VALUE ユーザー・イグジット・ファンクション (506 ページを参照)

表 50　GETENV オプションの概要

情報タイプ	説明
一般情報タイプ	
("LAG" , "<unit>")	ラグ情報を返します。
("LASTERR" , "<option>")	最後にレプリケートされた操作 (詳細なエラー情報を含む) に関する情報を返します。
("JULIANTIMESTAMP")	現在のシステム時間をユリウス形式で返します。
("RECSOUTPUT")	プロセス起動以降に Extract がトレイル・ファイルに書き込んだレコード数を返します。
表レベルの統計情報タイプ	
("STATS", ["TABLE", "<table_name"], "<operation_type">)	1 つ以上の指定した表に対する DML および DDL 操作の統計を返します。
("DELTASTATS", ["TABLE", "<Table_Name"], "<operation_type">)	1 つ以上の指定した表に対する DML および DDL 操作の統計をデルタ値として返します。
Oracle GoldenGate 情報タイプ	
("GGENVIRONMENT", "<option>")	Oracle GoldenGate 環境情報を返します。
("GGFILEHEADER", "<option>")	ファイル・ヘッダーに保持されている Oracle GoldenGate トレイル・ファイルの形式およびプロパティを返します。
("GGHEADER", "<option>")	Oracle GoldenGate レコード・ヘッダー情報を返します。
("RECORD", "<option>")	Oracle GoldenGate トレイル・ファイルのレコードの場所を示す順序番号および RBA を返します。
データベース情報タイプ	
("DBENVIRONMENT", "<option>")	グローバル・データベース環境情報を返します。

表 50　GETENV オプションの概要 (続き)

情報タイプ	説明
("TRANSACTION", "<option>")	ソース・トランザクションに関する情報を返します。
オペレーティング・システム情報タイプ	
("OSVARIABLE", "<variable>")	指定されたオペレーティングシステム環境変数の文字列値を返します。
Base 24 情報タイプ	
("TLFKEY", "SYSKEY" "<unique key>")	一意キーを ACI の Base24 アプリケーションの TLF/PTLF レコードに関連付けます。

例　次の例では、TABLE 文の TOKENS 句で @GETENV ファンクションを使用して、Oracle GoldenGate レコード・ヘッダー内にユーザー・トークンを移入します。次のようにこのファンクションの複数のオプションを組み合せて使用して、特定の情報を返すことができます。

```
TABLE fin.product, TOKENS (
TKN-OSUSER = @GETENV ("GGENVIRONMENT", "OSUSERNAME"),
TKN-DOMAIN = @GETENV ("GGENVIRONMENT", "DOMAINNAME"),
TKN-COMMIT-TS = @GETENV ("GGHEADER", "COMMITTIMESTAMP"),
TKN-TABLE = @GETENV ("GGHEADER", "TABLENAME"),
TKN-OP-TYPE = @GETENV ("GGHEADER", "OPTYPE"),
TKN-LENGTH = @GETENV ("GGHEADER", "RECORDLENGTH"),
TKN-LAG-SEC = @GETENV ("LAG", "SECONDS"),
TKN-DB-USER = @GETENV ("DBENVIRONMENT", "DBUSER"),
TKN-DB-VER = @GETENV ("DBENVIRONMENT", "DBVERSION"),
TKN-ROWID = @GETENV ("RECORD", "GDVN"));
```

LAG 情報タイプの使用

@GETENV の LAG オプションでは、ラグ情報を返します。ラグは、Extract または Replicat がレコードを処理した時間と、データ・ソースのそのレコードのタイムスタンプとの差異です。LAG と <environment value> は、両方とも二重引用符で囲む必要があります。

構文　@GETENV ("LAG", "<unit>")

環境値	戻り値
"SEC"	ラグ (秒) を返します。LAG に対して単位が明示的に指定されていないときは、これがデフォルトです。
"MSEC"	ラグ (ミリ秒) を返します。
"MIN"	ラグ (分) を返します。

LASTERR 情報タイプの使用

@GETENV の LASTERR オプションでは、Replicat が最後に失敗した操作の情報を返します。このオプションはエラー情報を提供します。LASTERR は、Replicat プロセスのみとの使用が有効です。LASTERR と <environment value> は、両方とも二重引用符で囲む必要があります。

構文

```
@GETENV ("LASTERR", "<return value>")
```

環境値	戻り値
"DBERRNUM"	失敗した操作に関連するデータベース・エラー番号を返します。
"DBERRMSG"	失敗した操作に関連するデータベース・エラー・メッセージを返します。
"OPTYPE"	試行した操作タイプを返します。Oracle GoldenGate 操作タイプのリストは、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』の付録を参照してください。
"ERRTYPE"	エラーのタイプを返します。次のような結果になります。 ◆ DB(データベース・エラーの場合) ◆ MAP(マッピング・エラーの場合)

JULIANTIMESTAMP 情報タイプの使用

@GETENV の JULIANTIMESTAMP オプションでは、現在の時刻をユリウス形式で返します。単位はマイクロ秒 (百万分の 1 秒) です。Windows のタイムスタンプはミリ秒 (1000 分の 1 秒) の粒度を使用するので、Windows マシンでは値にゼロが埋め込まれます。次に、一般的な列マッピングの例を示します。

```
MAP dbo.tab8451, Target targ.tabjts, COLMAP (USEDEFAULTS, &
JTSS = @GETENV ("JULIANTIMESTAMP")
JTSFFFFFF = @date ("yyyy-mm-dd:hh:mi:ss.ffffff", "JTS", &
@getenv ("JULIANTIMESTAMP") ) )
;
```

JTSS および JTSFFFFFF 列は、次のような値になります。

```
212096320960773000 2010-12-17:16:42:40.773000
212096321536540000 2010-12-17:16:52:16.540000
212096322856385000 2010-12-17:17:14:16.385000
212096323062919000 2010-12-17:17:17:42.919000
212096380852787000 2010-12-18:09:20:52.787000
```

数字の最後の 3 桁 (マイクロ秒) はすべて、埋め込まれたゼロが含まれています。

構文

```
@GETENV ("JULIANTIMESTAMP")
```

RECSOUTPUT 情報タイプの使用

@GETENV の RECSOUTPUT オプションでは、起動後に Extract がトレイル・ファイルに書き込んだレコード数の現在のカウントを取得します。返される値は、表またはトランザクションではなく、Extract セッション自体に固有の値です。Extract が停止して再起動されると、カウントは 1 にリセットされます。

構文

```
@GETENV ("RECSOUTPUT")
```

STATS および DELTASTATS 情報タイプの使用

@GETENV の STATS および DELTASTATS オプションでは、次のいずれかまたはすべてについて、表ごとに Extract または Replicat が処理した操作数を返します。

- INSERT 操作
- UPDATE 操作
- DELETE 操作
- TRUNCATE 操作
- 合計 DML 操作
- 合計 DDL 操作
- 競合の検出および解決 (CDR) 機能を使用する場合、発生した競合の数。
- 成功した CDR 解決の数
- 失敗した CDR 解決の数

STATS はプロセス起動以降の数を返しますが、DELTASTATS は最後の DELTASTATS の実行以降の数を返します。

実行ロジックは次のようになります。

- Extract が STATS または DELTASTATS で @GETENV を満たすトランザクション・レコードを処理すると、表名は TABLE 文内の解決されたソース表と照合されます。
- Replicat が STATS または DELTASTATS で @GETENV を満たすトレイル・レコードを処理すると、表名は MAP 文の TARGET 句内の解決されたターゲット表と照合されます。

@GETENV 構文内のすべての入力要素は、二重引用符で囲む必要があります。

解決されていない表エントリや不適切な構文など、このファンクションの処理中にエラーが発生した場合、リクエストされた統計値にゼロ (0) を返します。

"TABLE" を使用した出力の制約

これらのファンクションの出力を、指定した表に制限できます。それ以外の場合は、@GETENV が実行するプロセスに応じて、すべての TABLE または MAP 文内のすべての表に操作数が返されます。

たとえば、次の場合、"hr" スキーマの表の DML 操作のみを数えます。

```
MAP fin.*, TARGET fin.*;
MAP hr.*, TARGET hr.*;
MAP hq.rpt, TARGET hq.rpt, COLMAP (USEDEFAULTS, CNT = @GETENV ("STATS", "TABLE", "hr.*",
"DML"));
```

同様に、次の場合は "hr" スキーマの "emp" 表の DML 操作のみを数えます。

```
MAP fin.*, TARGET fin.*;
MAP hr.*, TARGET hr.*;
MAP hq.rpt, TARGET hq.rpt, COLMAP (USEDEFAULTS, CNT = @GETENV ("STATS", "TABLE",
"hr.emp", "DML"));
```

これに対して、次の例では、STATS に特定の表を指定していないため、MAP 文の TARGET 句内にあるすべてのスキーマのすべての表の、すべての INSERT、UPDATE および DELETE 操作を数えます。

```
MAP fin.*, TARGET fin.*;
MAP hr.*, TARGET hr.*;
MAP hq.rpt, TARGET hq.rpt, COLMAP (USEDEFAULTS, CNT = &
@GETENV ("STATS", "DML"));
```

表の指定の繰返しが操作数に与える影響について

同じソース表を複数の出力トレイルに処理している Extract は、@GETENV にリンクした表が書き込まれるローカライズされた各出力トレイルに基づいて、統計を返します。たとえば、Extract が表 "ABC" への 100 件の挿入を取得し、表 ABC を 3 つのトレイルに書き込む場合、@GETENV の結果は 300 になります。

```
EXTRACT ABC
...
EXTTRAIL c:\ogg\dirdat\aa;
TABLE TEST.ABC;
EXTTRAIL c:\ogg\dirdat\bb;
TABLE TEST.ABC;
TABLE EMI, TOKENS (TOKEN-CNT = @GETENV ("STATS", "TABLE", "ABC", "DML"));
EXTTRAIL c:\ogg\dirdat\cc;
TABLE TEST.ABC;
```

ソース表を単一の出力トレイルに複数回書き込む Extract の場合、または同じ TARGET 表に対して複数の MAP 文がある Replicat の場合、統計の結果は、一致するすべての TARGET エントリに基づいています。たとえば、Replicat が、REGION "WEST" の 20 行、REGION "EAST" の 10 行、REGION "NORTH" の 5 行、REGION "SOUTH" の 2 行のすべてを表 "ABC" に対してフィルタする場合、@GETENV の結果は 37 になります。

```
REPLICAT ABC
...
MAP TEST.ABC, TARGET TEST.ABC, FILTER (@STREQ (REGION, "WEST"));
MAP TEST.ABC, TARGET TEST.ABC, FILTER (@STREQ (REGION, "EAST"));
MAP TEST.ABC, TARGET TEST.ABC, FILTER (@STREQ (REGION, "NORTH"));
MAP TEST.ABC, TARGET TEST.ABC, FILTER (@STREQ (REGION, "SOUTH"));
MAP TEST.EMI, TARGET TEST.EMI, &
    COLMAP (CNT = @GETENV ("STATS", "TABLE", "ABC", "DML"));
```

複数の統計の取得

@GETENV の複数のインスタンスを実行して、異なる操作タイプの数を取得できます。

次の例では、INSERT および UPDATE の操作のみの統計を返します。

```
REPLICAT TEST
..
..
MAP TEST.ABC, TARGET TEST.ABC, COLMAP (USEDEFAULTS, IU = @COMPUTE(@GETENV &
    ("STATS", "TABLE", "ABC", "DML") - (@GETENV ("STATS", "TABLE", &
    "ABC", "DELETE"));
```

次の例では、DDL および TRUNCATE の操作の統計を返します。

```
REPLICAT TEST2
..
..
MAP TEST.ABC, TARGET TEST.ABC, COLMAP (USEDEFAULTS, DDL = @COMPUTE &
(@GETENV ("STATS", "DDL") + (@GETENV ("STATS", "TRUNCATE"));
```

ユースケースの例

次のユースケースでは、ソースからのすべての DML がターゲットに正常に適用された場合、SEND REPLICAT を RESUME にして GGSCI から再開されるまで、EVENTACTIONS を SUSPEND にすることにより、Replicat は一時停止します。

Extract パラメータ・ファイルで使用される GETENV は次のとおりです。

```
TABLE HR1.HR*;
TABLE HR1.STAT, TOKENS ("env_stats" = @GETENV("STATS", "TABLE", &
    "HR1.HR*", "DML"));
```

Replicat パラメータ・ファイルで使用される GETENV は次のとおりです。

```
MAP HR1.HR*, TARGET HR2.*;
MAP HR1.STAT, TARGET HR2.STAT, filter (
    @if (
    @token ("stats") =
    @getenv("STATS", "TABLE", "TSSCAT.TCUSTORD", "DML"), 1, 0 )
    ),
    eventactions (suspend);
```

構文

```
@GETENV ({"STATS" | "DELTASTATS"}, ["TABLE", "<table_name"], "<statistic_type">)
```

入力値	戻り値
"TABLE", "<table_name>"	オプションで、指定した表に対してのみ STATS または DELTASTATS を実行します。このオプションを指定しないと、パラメータ・ファイルで TABLE (Extract) または MAP (Replicat) パラメータに指定したすべての表に対する数が返されます。 <table_name> の有効な値は次のとおりです。 ◆ ""<schema>.<table>" は特定の表を指定します。 ◆ "<table>" はデフォルト・スキーマの特定の表を指定します。 ◆ ""<schema>.*" はスキーマのすべての表を指定します。 ◆ ""*" はデフォルト・スキーマのすべての表を指定します。
"<statistic_type>" は次のいずれかです。	
"INSERT"	処理された INSERT 操作の数を返します。
"UPDATE"	処理された UPDATE 操作の数を返します。
"DELETE"	処理された DELETE 操作の数を返します。
"DML"	処理された INSERT、UPDATE および DELETE 操作の合計を返します。
"TRUNCATE"	処理された TRUNCATE 操作の数を返します。この変数は、Oracle GoldenGate の DDL レプリケーションが使用されていない場合のみ、数を返します。DDL レプリケーションが使用されている場合、この変数はゼロを返します。
"DDL"	TRUNCATE、および、すべての範囲 (MAPPED、UNMAPPED、OTHER) の DDL パラメータの INCLUDE 句および EXCLUDE 句で指定した DDL など、処理された DDL 操作の数を返します。 この変数は、Oracle GoldenGate の DDL レプリケーションが使用されている場合のみ、数を返します。この変数は、"DELTASTATS" には無効です。
CDR_CONFLICTS	競合の検出および解決 (CDR) 機能の実行時に Replicat で検出された競合の数を返します。 特定の表に対する例 : @GETENV("STATS","TABLE","HR.EMP","CDR_CONFLICTS") Replicat で処理されるすべての表に対する例 : @GETENV("STATS","CDR_CONFLICTS")

入力値	戻り値

CDR_RESOLUTIONS_SUCCEEDED	競合の検出および解決 (CDR) 機能の実行時に Replicat で解決した競合の数を返します。
	特定の表に対する例 :
	`@GETENV("STATS","TABLE","HR.EMP", "CDR_RESOLUTIONS_SUCCEEDED")`
	Replicat で処理されるすべての表に対する例 :
	`@GETENV("STATS","CDR_RESOLUTIONS_SUCCEEDED")`

CDR_RESOLUTIONS_FAILED	競合の検出および解決 (CDR) 機能の実行時に Replicat で解決できなかった競合の数を返します。
	特定の表に対する例 :
	`@GETENV("STATS","TABLE","HR.EMP", "CDR_RESOLUTIONS_FAILED")`
	Replicat で処理されるすべての表に対する例 :
	`@GETENV("STATS","CDR_RESOLUTIONS_FAILED")`

GGENVIRONMENT 情報タイプの使用

@GETENV の GGENVIRONMENT オプションでは、Oracle GoldenGate 環境に関する情報を返します。このオプションは、Extract および Replicat プロセスに有効です。GGSENVIRONMENT と <environment value> は、両方とも二重引用符で囲む必要があります。

構文 `@GETENV ("GGENVIRONMENT", "<return value>")`

環境値	戻り値
"DOMAINNAME"	(Windows のみ) プロセスを開始したユーザーに関連付けられているドメイン名を返します。
"GROUPDESCRIPTION"	GGSCI の ADD コマンドでのグループ作成時に DESCRIPTION オプションで説明が指定されている場合に、チェックポイント・ファイルから取得したグループの説明。
"GROUPNAME"	プロセス・グループ名を返します。
"GROUPTYPE"	プロセスのタイプ (EXTRACT または REPLICAT) を返します。
"HOSTNAME"	Extract または Replicat プロセスが実行されているシステムの名前を返します。
"OSUSERNAME"	プロセスを起動したオペレーティング・システムのユーザー名を返します。
"PROCESSID"	オペレーティング・システムによってプロセスに割り当てられているプロセス ID。

GGHEADER 情報タイプの使用

@GETENV の GGHEADER オプションでは、Oracle GoldenGate トレイル・レコードのヘッダー部分の情報を返します。Oracle GoldenGate トレイル内のすべてのデータ・レコードには、レコードのトランザクション環境が記述されたヘッダーが含まれています。レコード・ヘッダーの詳細は、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』を参照してください。

このオプションは、Extract および Replicat プロセスに有効です。GGHEADER と <environment value> は、両方とも二重引用符で囲む必要があります。

構文

```
@GETENV ("GGHEADER", "<return value>")
```

環境値	戻り値
"BEFOREAFTERINDICATOR"	レコードがビフォア・イメージかアフター・イメージかを示すビフォアまたはアフター・インジケータを返します。次のような結果になります。 ◆ BEFORE(ビフォア・イメージ) ◆ AFTER(アフター・イメージ)
"COMMITTIMESTAMP"	次の例のように、YYYY-MM-DD HH:MI:SS.FFFFFF 形式で表現されたトランザクション・タイムスタンプ (トランザクションがコミットされた時間) を返します。 2011-01-24 17:08:59.000000
"LOGPOSITION"	データ・ソース内の Extract プロセスの位置を返します。(LOGRBA オプションを参照してください。)
"LOGRBA"	LOGRBA および LOGPOSITION は、データ・ソース内のレコードの詳細な位置を保持します。トランザクション・ログベースの製品では、LOGRBA は順序番号、LOGPOSITION は相対バイト・アドレスです。ただしこれらの値は、取得方法とデータベースのタイプによって異なります。
"OBJECTNAME" | "TABLENAME"	表名または (順序の場合) オブジェクト名を返します。
"OPTYPE"	操作のタイプを返します。次のような結果になります。 INSERT UPDATE DELETE ENSCRIBE COMPUPDATE SQL COMPUPDATE PK UPDATE TRUNCATE 操作が上記のいずれのタイプでもない場合、このファンクションは単語 TYPE およびこのタイプに割り当てられている数字を返します。レコード・タイプの詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』の付録を参照してください。

環境値	戻り値
"RECORDLENGTH"	レコード長 (バイト) を返します。
"TRANSACTIONINDICATOR"	トランザクション・インジケータを返します。この値は、Logdump ユーティリティを使用して表示できる、レコード・ヘッダーの TransInd フィールドに相当します (『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください)。 次のような結果になります。 ◆ BEGIN(TransInD が 0、トランザクションの最初のレコードを示します) ◆ MIDDLE(TransInD が 1、トランザクションの中間のレコードを示します) ◆ END(TransInD が 2、トランザクションの最後のレコードを示します) ◆ WHOLE(TransInD が 3、トランザクションの唯一のレコードを示します)

GGFILEHEADER 情報タイプの使用

@GETENV の GGFILEHEADER オプションでは、ファイル・ヘッダーに保持されている Oracle GoldenGate の抽出ファイルまたはトレイル・ファイルの属性を返します。トレイル内のすべてのファイルには、このヘッダーが含まれています。ヘッダーには、ファイル自体、およびファイルが使用されている環境について記述されています。

ファイル・ヘッダーは、トレイル・ファイルの先頭部 (データ・レコードより先) にレコードとして保持されています。Oracle GoldenGate プロセスは、トレイル・ヘッダーに保持されているレコードに関する情報を使用して、レコードが現在の Oracle GoldenGate リリースにサポートされているフォーマットかどうかを確認できます。

トレイルのヘッダー・フィールドはトークンとして保管され、トークン・フォーマットは Oracle GoldenGate のすべてのリリースで同じです。Oracle GoldenGate リリースにサポートされていないトークンは無視されます。以前の Oracle GoldenGate リリースとの互換性を維持するために、非推奨のトークンにはデフォルト値が割り当てられます。

このオプションは、Replicat プロセスに有効です。GGFILEHEADER と <environment value> は、両方とも二重引用符で囲む必要があります。

注意 **データベース、オペレーティング・システム、または Oracle GoldenGate リリースがトークンに関する情報を提供しない場合は、NULL 値が返されます。**

構文

```
@GETENV ("GGFILEHEADER", "<return_value>")
```

環境値	戻り値
"COMPATIBILITY"	トレイル・ファイルの Oracle GoldenGate 互換性レベルを返します。トレイル・ファイルのデータ・レコードを読み取るためには、現在の Oracle GoldenGate リリースの互換性レベルがトレイル・ファイルの互換性レベル以上である必要があります。現在有効な値は、0 または 1 です。 ◆ 1 は、トレイル・ファイルが Oracle GoldenGate リリース 10.0 以上 (ファイル・バージョニング情報を含むファイル・ヘッダーをサポート) であることを示します。 ◆ 0 は、トレイル・ファイルが Oracle GoldenGate リリース 10.0 よりも古いことを示します。これらのリリースでは、ファイル・ヘッダーはサポートされていません。値 0 は、これらの Oracle GoldenGate リリースとの下位互換性を維持するために使用されます。
トレイル・ファイルに関する情報	
"CHARSET"	トレイル・ファイルのグローバル・キャラクタ・セットを返します。次に例を示します。 WCP1252-1
"CREATETIMESTAMP"	トレイルが作成された時間を INIT64 のローカル GMT ユリウス時間で返します。
"FILENAME"	トレイル・ファイル名を返します。ファイル・システムに応じて、フォワード・スラッシュまたはバックワード・スラッシュを使用した絶対パスまたは相対パスの場合があります。
"FILEISTRAIL"	トレイル・ファイルが単一のファイル (バッチ実行で作成されたファイルなど) か、オンライン継続処理用トレイルの順序番号付けされたファイルかを示す、True/false フラグを返します。false の場合、SeqNum サブトークンは無効です。
"FILESEQNO"	トレイル・ファイルの順序番号 (先行ゼロを除く) を返します。たとえば、ファイルの順序番号が aa000026 の場合、FILESEQNO は 26 を返します。
"FILESIZE"	トレイル・ファイルのサイズを返します。アクティブ・ファイルの場合は NULL を返し、ファイルが一杯になりトレイルがロール・オーバーされているときはサイズ値を返します。
"FIRSTRECCSN"	トレイル・ファイルの最初のレコードのコミット順序番号 (CSN) を返します。トレイル・ファイルが完成するまで、値は NULL です。CSN の詳細は、309 ページの付録 4 を参照してください。
"LASTRECCSN"	トレイル・ファイルの最後のレコードのコミット順序番号 (CSN) を返します。トレイル・ファイルが完成するまで、値は NULL です。CSN の詳細は、309 ページの付録 4 を参照してください。
"FIRSTRECIOTIME"	最初のレコードがトレイル・ファイルに書き込まれた時間を返します。トレイル・ファイルが完成するまで、値は NULL です。

環境値	戻り値
"LASTRECIOTIME"	最後のレコードがトレイル・ファイルに書き込まれた時間を返します。トレイル・ファイルが完成するまで、値は NULL です。
"URI"	トレイル・ファイルを作成したプロセスのユニバーサル・リソース識別子を次の形式で返します。 <host_name>:<dir>:[:<dir>][:<dir_n>]<group_name> **条件 :** ◆ host_name は、プロセスをホストするサーバー名です。 ◆ dir は、Oracle GoldenGate インストール・パスのサブディレクトリです。 ◆ group_name は、プロセスにリンクされているプロセス・グループ名です。 **例** : sys1:home:oracle:v9.5:extora トレイルがどこでどのプロセスに処理されたかを示します。以前の実行の履歴も含まれます。
"URIHISTORY"	現在のプロセス以前にトレイル・ファイルに書き込んだプロセスの URI のリストを返します。 ◆ プライマリ Extract の場合、このフィールドは空です。 ◆ データ・ポンプの場合は、このフィールドは URIHistory+ 入力トレイル・ファイルの URI です。
トレイル・ファイルを作成した Oracle GoldenGate プロセスに関する情報	
"GROUPNAME"	トレイルを作成した Extract グループに関連付けられているグループ名を返します。グループ名は、ADD EXTRACT コマンドで指定された名前です。たとえば、"ggext" です。
"DATASOURCE"	プロセスによって読み取られたデータ・ソースを返します。次のいずれかになります。 ◆ DS_EXTRACT_TRAILS(ソースは、変更データが移入された Oracle GoldenGate 抽出ファイルです) ◆ DS_DATABASE(ソースは SOURCEISTABLE 主導の初期ロードで使用され、直接選択されてトレイルに書き込まれたデータベース表です) ◆ DS_TRAN_LOGS (ソースはデータベース・トランザクション・ログです) ◆ DS_INITIAL_DATA_LOAD(ソースは Extract で、データはソース表から直接取得されました) ◆ DS_VAM_EXTRACT(ソースは Vendor Access Module です) ◆ DS_VAM_TWO_PHASE_COMMIT(ソースは VAM トレイルです)
"GGMAJORVERSION"	トレイルを作成した Extract プロセスのメジャー・バージョン (整数で表現) を返します。たとえば、バージョン 1.2.3 の場合は 1 を返します。
"GGMINORVERSION"	トレイルを作成した Extract プロセスのマイナー・バージョン (整数で表現) を返します。たとえば、バージョン 1.2.3 の場合は 2 を返します。

環境値	戻り値
"GGVERSIONSTRING"	トレイルを作成した Extract プロセスのメンテナンス (またはパッチ) レベル (整数で表現) を返します。たとえば、バージョン 1.2.3 の場合は 3 を返します。
"GGMAINTENANCELEVEL"	プロセスのメンテナンス・バージョン (xx.xx.xx) を返します。
"GGBUGFIXLEVEL"	プロセスのパッチ・バージョン (xx.xx.xx.x) を返します。
"GGBUILDNUMBER"	プロセスのビルド番号を返します。
トレイル・ファイルのローカル・ホストに関する情報	
"HOSTNAME"	トレイルに書き込んだ Extract が実行されているマシンの DNS 名を返します。次に例を示します。 ◆ sysa ◆ sysb ◆ paris ◆ hq25
"OSVERSION"	トレイルに書き込んだ Extract が実行されているマシンのオペレーティング・システムのメジャー・バージョンを返します。次に例を示します。 ◆ Versions10_69 ◆ #1 SMP Fri Feb 24 16:56:28 EST 2006 ◆ 5.00.2195 Service Pack 4
"OSRELEASE"	トレイルに書き込んだ Extract が実行されているマシンのオペレーティング・システムのリリース・バージョンを返します。たとえば、OSVERSION で示した例のリリース・バージョンは次のようになることがあります。 ◆ 5.10 ◆ 2.6.9-34.ELsmp
"OSTYPE"	トレイルに書き込んだ Extract が実行されているマシンのオペレーティング・システムのタイプを返します。次に例を示します。 ◆ SunOS ◆ Linux ◆ Microsoft Windows
"HARDWARETYPE"	トレイルに書き込んだ Extract が実行されているマシンのハードウェアのタイプを返します。次に例を示します。 ◆ sun4u ◆ x86_64 ◆ x86
トレイル・ファイルのデータを生成したデータベースに関する情報	
"DBNAME"	データベース名 (findb など) を返します。

環境値	戻り値
"DBINSTANCE"	データベースのタイプに適切な場合、データベース・インスタンス名 (ORA1022A など) を返します。
"DBTYPE"	トレイル・ファイルのデータを生成したデータベースのタイプを返します。次のいずれかになります。 DB2 UDB DB2 ZOS CTREE MSSQL MYSQL ORACLE SQLMX SYBASE TERADATA NONSTOP ENSCRIBE ODBC
"DBCHARSET"	トレイル・ファイルのデータを生成したデータベースで使用されているキャラクタ・セットを返します。(一部のデータベースでは、これは空になります。)
"DBMAJORVERSION"	トレイル・ファイルのデータを生成したデータベースのメジャー・バージョンを返します。
"DBMINORVERSION"	トレイル・ファイルのデータを生成したデータベースのマイナー・バージョンを返します。
"DBVERSIONSTRING"	トレイル・ファイルのデータを生成したデータベースのメンテナンス (パッチ) ・レベルを返します。
"DBCLIENTCHARSET"	データベース・クライアントに使用されているキャラクタ・セットを返します。
"DBCLIENTVERSIONSTRING"	データベース・クライアントのメンテナンス (パッチ) ・レベルを返します。(一部のデータベースでは、これは空になります。)
以前のトレイル・ファイルから繰り越されるリカバリ情報	
"RECOVERYMODE"	内部で Oracle GoldenGate に使用されるリカバリ情報を返します。
"LASTCOMPLETECSN"	内部で Oracle GoldenGate に使用されるリカバリ情報を返します。
"LASTCOMPLETEXIDS"	内部で Oracle GoldenGate に使用されるリカバリ情報を返します。
"LASTCSN"	内部で Oracle GoldenGate に使用されるリカバリ情報を返します。
"LASTXID"	内部で Oracle GoldenGate に使用されるリカバリ情報を返します。
"LASTCSNTS"	内部で Oracle GoldenGate に使用されるリカバリ情報を返します。

RECORD 情報タイプの使用

@GETENV の RECORD オプションでは、Oracle GoldenGate トレイル・ファイルのレコードの場所を返します。このファンクションは、ファイルの順序番号、およびそのファイルの相対バイト・アドレスを返すことができます。この 2 つの値から、特定のレコードに関連する一意の値が得られます。

このオプションは、Extract データ・ポンプおよび Replicat プロセスに有効です。RECORD と <environment value> は、両方とも二重引用符で囲む必要があります。

構文

```
@GETENV ("RECORD", "<environment value>")
```

環境値	戻り値
"FILESEQNO"	トレイル・ファイルの順序番号 (先行ゼロを除く) を返します。
"FILERBA"	FILESEQNO ファイル内のレコードの相対バイト・アドレスを返します。
"RSN"	トランザクション内のレコード順序番号を返します。
TIMESTAMP	レコードのタイムスタンプを返します。

DBENVIRONMENT 情報タイプの使用

@GETENV の DBENVIRONMENT オプションでは、データベースのグローバル環境情報を返します。このオプションは、Extract および Replicat プロセスに有効です。DBENVIRONMENT と <environment value> は、両方とも二重引用符で囲む必要があります。

構文

```
@GETENV ("DBENVIRONMENT", "<return value>")
```

環境値	戻り値
"DBNAME"	データベース名を返します。
"DBVERSION"	データベース・バージョンを返します。
"DBUSER"	データベース・ログイン・ユーザーを返します。*注意:* Microsoft SQL Server ではユーザー ID は記録されません。
"SERVERNAME"	サーバー名を返します。

TRANSACTION 情報タイプの使用

@GETENV の TRANSACTION オプションでは、ソース・トランザクションに関する情報を返します。このオプションは、Extract プロセスに有効です。TRANSACTION と <environment value> は、両方とも二重引用符で囲む必要があります。

構文

```
@GETENV ("TRANSACTION", "<return value>")
```

環境値	戻り値
"TRANSACTIONID" | "XID"	トランザクション ID 番号を返します。トランザクション ID および CSN は、各トランザクションの最初のレコードに関連付けられており、トレイル・レコードにトークンとして保持されています。各トランザクション ID には、CSN が関連付けられています。トランザクション ID トークンは、相対値として評価されることはないため、どのプラットフォームでもゼロは埋め込まれません。これらは、一致するかしないかのみの評価が行われます。トレイルでは、トランザクション ID トークンは TRANID として表示されることに注意してください。
"CSN"	コミット順序番号 (CSN) を返します。CSN は、Oracle、DB2 LUW、および DB2 z/OS データベースに対して返されるときは、ゼロが埋め込まれません。サポートされている他のすべてのデータベースの場合は、CSN にゼロが埋め込まれます。Sybase CSN の場合、ドット (.) で区切られている各サブ文字列は、そのサブ文字列が変更されない長さまでゼロが埋め込まれます。 トレイルでは、CSN トークンは LOGCSN として表示されることに注意してください。CSN トークンの補足情報は、TRANSACTIONID | XID 環境値を参照してください。 CSN 自体の詳細は、309 ページの付録 4 を参照してください。
"TIMESTAMP"	トランザクションのコミット・タイムスタンプを返します。
"NAME"	使用可能な場合、トランザクション名を返します。
"USERID"	(Oracle) 最後のトランザクションをコミットしたデータベース・ユーザーの Oracle ユーザー ID を返します。
"USERNAME"	(Oracle) 最後のトランザクションをコミットしたデータベース・ユーザーの Oracle ユーザー名を返します。
"PLANNAME"	(z/OS 上の DB2) 現在のトランザクションを最初に実行した計画名を返します。計画名は、リカバリ・ログ・レコードの開始ユニットに含まれています。

OSVARIABLE 情報タイプの使用

@GETENV の OSVARIABLE オプションでは、指定されたオペレーティングシステム環境変数の文字列値を返します。このオプションは、Extract および Replicat に有効です。OSVARIABLE と <variable> は、両方とも二重引用符で囲む必要があります。

構文

```
@GETENV ("OSVARIABLE", "<variable>")
```

環境値	戻り値

"<variable>"	変数名。指定された変数と正確に一致するものが検索されます。たとえば、UNIX の grep コマンドでは、次のすべての変数が返されますが、@GETENV ("OSVARIABLE", "HOME") では HOME の値のみが返されます。
	`ANT_HOME=/usr/local/ant`
	`JAVA_HOME=/usr/java/j2sdk1.4.2_10`
	`HOME=/home/judyd`
	`ORACLE_HOME=/rdbms/oracle/ora1022i/64`
	オペレーティング・システムが大 / 小文字区別をサポートしている場合、検索では大 / 小文字が区別されます。

TLFKEY 情報タイプの使用

@GETENV の TLFKEY オプションでは、一意キーと ACI の Base24 アプリケーションの TLF/PTLF レコードを関連付けます。64 ビット・キーは、次のアイテムが連結されて構成されます。

- 2000 年以降の秒数。
- TLF/PTLF ブロックのレコードのブロック数に 10 を掛けた値。
- ユーザーによって指定されたノード (0 ～ 255 である必要があります)。

このオプションは、Extract および Replicat プロセスに有効です。TLFKEY は二重引用符で囲む必要があります。

構文 `@GETENV ("TLFKEY", SYSKEY <unique key>)`

環境値	戻り値
< 一意キー >	ソース TLF/PTLF ファイルの NonStop ノード番号。 例 : `GETENV ("TLFKEY", SYSKEY, 27)`

GETVAL

@GETVAL ファンクションでは、ストアド・プロシージャまたは問合せから値を抽出し、MAP または TABLE 文の FILTER または COLMAP 句の入力として使用できるようにします。

@GETVAL を使用してパラメータ値を抽出できるかどうかは、次のことに依存します。

1. ストアド・プロシージャまたは問合せの実行の成否。
2. ストアド・プロシージャまたは問合せの結果の失効の有無。

行方不明列値の処理

値を抽出できない場合、@GETVAL ファンクションの結果は " 列行方不明 " になります。一般的にこのような状態は、データベースが変更されたログのみを記録する場合に更新操作で発生します。

通常、これは列をマッピングできないことを意味します。行方不明の列値をテストするには、@COLTEST ファンクションを使用して @GETVAL の結果をテストした後、必要な場合には、行方不明の値を補うために、列に代替値をマップします。または、列値を使用可能にするために、TABLE または MAP パラメータの FETCHCOLS または FETCHCOLSEXCEPT オプションを使用して、値がログに存在しない場合にデータベースから値をフェッチさせます。必要な列のサプリメンタル・ロギングを有効にする方法も有益です。

構文

```
@GETVAL (<name>.<parameter>)
```

引数	説明
<name>	ストアド・プロシージャまたは問合せ名。SQLEXEC を使用してプロシージャまたは問合せを使用するときは、次の値が有効です。 問合せ : SQLEXEC 句の ID オプションで指定されている論理名を使用します。ID は、問合せで必須の SQLEXEC 引数です。 ストアド・プロシージャ : TABLE または MAP 文内でプロシージャを実行する回数に応じて、次のいずれかを使用します。 ◆ 複数回実行するときは、SQLEXEC 文の ID 句で定義されている論理名を使用します。ID は、1 つのプロシージャを複数回実行するときに必須です。 ◆ 1 回実行するときは、ストアド・プロシージャの実際の名前を使用します。
< パラメータ >	有効な値は、次のいずれかです。 ◆ データを抽出し、列マップに渡すストアド・プロシージャまたは問合せ内のパラメータ名。 ◆ ストアド・プロシージャまたは問合せが返す値を抽出するときは、RETURN_VALUE。

例 1

次の例では、各 MAP 文が @GETVAL ファンクション内で論理名 lookup1 および lookup2 を参照することによって、ストアド・プロシージャ lookup をコールし、各結果セットを適切に参照できるようにします。

```
MAP schema.srctab, TARGET schema.targtab,
SQLEXEC (SPNAME lookup, ID lookup1, PARAMS (param1 = srccol)),
COLMAP (targcol1 = @GETVAL (lookup1.param2));
MAP schema.srctab, TARGET schema.targtab2,
SQLEXEC (SPNAME lookup, ID lookup2, PARAMS (param1 = srccol)),
COLMAP (targcol2= @GETVAL (lookup2.param2));
```

例 2

次の例では、ストアド・プロシージャ lookup を 1 回実行します。このケースでは、実際のプロシージャ名を使用します。論理名は必要ありません。

```
MAP schema.tab1, TARGET schema.tab2,
SQLEXEC (SPNAME lookup, PARAMS (param1 = srccol)),
COLMAP (targcol = @GETVAL (lookup.param1));
```

例 3 次の例では、問合せを実行し、取得した値を @GETVAL を使用してマップします。

```
MAP sales.account, TARGET sales.newacct,
SQLEXEC (ID lookup,
QUERY " select desc_col into desc_param from lookup_table "
" where code_col = :code_param ",
PARAMS (code_param = account_code)),
COLMAP (newacct_id = account_id, newacct_val = @GETVAL (lookup.desc_param));
```

代替構文

SQLEXEC では、@GETVAL キーワードを明示的に使用せずにパラメータ結果を取得できます。次の形式で、プロシージャ名 (問合せまたはプロシージャの複数のインスタンスを使用する場合は論理名) およびパラメータを参照します。

構文

```
{<procedure name> | <logical name>}.<parameter>
```

例 1 次の例では、@GETVAL キーワードを使用せずに、@GETVAL が暗黙的に proc1.p2 で呼び出されます。

```
MAP test.tab1, TARGET test.tab2,
SQLEXEC (SPNAME proc1, ID myproc, PARAMS (p1 = sourcecol1)),
COLMAP (targcol1 = proc1.p2);
```

例 2 次の例では、@GETVAL キーワードを使用せずに、@GETVAL が暗黙的に lookup.desc_param で呼び出されます。

```
MAP sales.account, TARGET sales.newacct,
SQLEXEC (ID lookup,
QUERY " select desc_col into desc_param from lookup_table "
" where code_col = :code_param ",
PARAMS (code_param = account_code)),
COLMAP (newacct_id = account_id, newacct_val = lookup.desc_param);
```

HEXTOBIN

@HEXTOBIN ファンクションでは、指定された 16 進数データ文字列を RAW 形式に変換します。

構文

```
@HEXTOBIN(<data>)
```

引数	説明
<data>	ソース列名、式、または引用符で囲まれたリテラル文字列。

例 @HEXTOBIN("414243") は、3 バイト (0x41 0x42 0x43) に変換されます。

HIGHVAL | LOWVAL

@HIGHVAL および @LOWVAL ファンクションは、値を生成する必要があり、その値を上限または下限内で抑制する場合に使用します。これらのファンクションは、同一の名前の COBOL ファンクションをエミュレートします。

@HIGHVAL および @LOWVAL は、文字列およびバイナリ・データ型とのみ使用します。文字列と使用する場合は、@STRNCMP のみ有効です。10 進数または日付データ型、もしくは SQLEXEC 操作とともに使用すると、エラーを引き起こすことがあります。DOUBLE データ型では、-1 または 0(Oracle NUMBER の精度指定なし、位取り指定なし) になります。

構文

```
@HIGHVAL ([<length>]) | @LOWVAL ([<length>])
```

引数	説明
<length>	オプションです。バイナリ出力長 (バイト) を指定します。<length> の最大値は、ターゲット列長です。

例

次の例では、group_level 列のサイズを 5 バイトとみなします。

ファンクション文	結果
group_level = @HIGHVAL()	{0xFF, 0xFF, 0xFF, 0xFF, 0xFF}
group_level = @LOWVAL()	{0x00, 0x00, 0x00, 0x00, 0x00}
group_level = @HIGHVAL(3)	{0xFF, 0xFF, 0xFF}
group_level = @LOWVAL(3)	{0x00, 0x00, 0x00}

IF

@IF ファンクションでは、条件に基づいて 2 つの値のうち 1 つを返します。@IF ファンクションは、他の Oracle GoldenGate ファンクションとともに使用して、1 つ以上の例外条件をテストする条件引数を開始できます。テストの結果に基づいて処理を実行するように指示できます。@IF 文は、必要に応じてネストできます。

構文

```
@IF (<conditional expression>, <value if non-zero>, <value if zero>)
```

引数	説明
<conditional expression>	有効な条件式または Oracle GoldenGate ファンクション。数値演算子 (=、>、< など) は、数値比較でのみ使用します。文字比較では、文字比較ファンクションの 1 つを使用します。
<value if non-zero>	非ゼロを true とみなします。
<value if zero>	ゼロ (0) を false とみなします。

例 1 次の例では、AMT 列がゼロより大きい場合にのみ量が返され、それ以外の場合はゼロが返されます。

```
AMOUNT_COL = @IF (AMT > 0, AMT, 0)
```

例 2 次の例では、STATE 列が CA、AZ、または NV の場合に WEST が返され、それ以外の場合は EAST が返されます。

```
REGION = @IF (@VALONEOF (STATE, "CA", "AZ", "NV"), "WEST", "EAST")
```

例 3 次の例では、2 つの列が両方ともゼロより大きい場合に、PRICE 列に QUANTITY 列を掛けた結果が返されます。それ以外の場合は、@COLSTAT (NULL) ファンクションによって、ターゲット列に NULL 値が生成されます。

```
ORDER_TOTAL = @IF (PRICE > 0 AND QUANTITY > 0, PRICE * QUANTITY,
@COLSTAT (NULL))
```

NUMBIN

@NUMBIN ファンクションでは、8 バイト以下のバイナリ文字列を数字に変換します。このファンクションは、ソース列に定義されているバイト・ストリームが、実際には文字列として表現されている数字のときに使用します。

構文

```
@NUMBIN (<source column>)
```

引数	説明
<source column>	変換する文字列を含むソース列名。

例 次の例では、@NUMBIN と @DATE を組み合せて、48 ビット Tandem 列を 64 ビットのローカル時間のユリウス値に変換します。

```
DATE = @DATE ("JTSLCT", "TTS" @NUMBIN (DATE))
```

NUMSTR

@NUMSTR ファンクションでは、文字列 (文字) 用の列または値を数字に変換します。@NUMSTR では、次のいずれかの操作を行います。

- 文字列 (文字) を数字にマップする。
- 数字のみを含む文字列用の列を算術式で使用する。

構文

```
@NUMSTR (<input>)
```

引数	説明
< 入力 >	次のいずれかを指定できます。 ◆ 文字用の列名。 ◆ 引用符で囲まれたリテラル文字列。

例

```
PAGE_NUM = @NUMSTR (ALPHA_PAGE_NO)
```

RANGE

@RANGE ファンクションでは、表の行を複数の Oracle GoldenGate プロセス間に分割します。サイズが大きく頻繁にアクセスされる表のスループットを向上させるためや、異なる宛先に配布するデータを分割するためにも使用できます。各範囲は、TABLE または MAP 文の FILTER 句で指定します。

@RANGE は、安全で拡張性あるファンクションです。同一の行が常に同一のプロセス・グループに処理されることを保証することにより、データの整合性を維持します。

@RANGE は、入力で指定された列のハッシュ値を計算します。列が指定されない場合、TABLE または MAP 文に KEYCOLS 句が指定されていれば、KEYCOLS 句を使用して、ハッシュする列を確認します。それ以外の場合は、主キー列を使用します。

Oracle GoldenGate は、指定された範囲数で均等に分散できるように、合計範囲数を調整します。

このファンクションには任意の列を指定できるため、相互に関係制約を持つ表の行は、参照整合性を維持するために、同一のプロセスまたはトレイルにグループ化する必要があります。

注意 **範囲は、Extract を使用して計算するほうが、Replicat を使用するよりも効率的です。ターゲット側で範囲を計算するには、Replicat がトレイル全体を読取り、各範囲の指定を満たすデータを見つける必要があります。**

構文

```
@RANGE (<range>, <total ranges> [, <column>] [, <column>] [, ...])
```

引数	説明
<range>	指定のプロセスまたはトレイルに割り当てる範囲。<total ranges> で定義する値を最大値として、1、2、3 のように指定します。
<total ranges>	割り当てる範囲の合計数 たとえば、データを 3 つのグループに分割するには、値 3 を使用します。
<column>	範囲割当ての基準にする列名。この引数はオプションです。使用されない場合、Oracle GoldenGate は表の主キーに基づいて範囲を割り当てます。

例 1

次の例では、レプリケーション・ワークロードは、ソース表 acct の ID 列に基づいて、3 つの範囲 (3 つの Replicat プロセス間) に分割されます。

(Replicat グループ 1 のパラメータ・ファイル)

```
MAP sales.acct, TARGET sales.acct, FILTER (@RANGE (1, 3, ID));
```

(Replicat グループ 2 のパラメータ・ファイル)

```
MAP sales.acct, TARGET sales.acct, FILTER (@RANGE (2, 3, ID));
```

(Replicat グループ 3 のパラメータ・ファイル)

```
MAP sales.acct, TARGET sales.acct, FILTER (@RANGE (3, 3, ID));
```

例 2 次の例では、1 つの Extract プロセスが処理ロードを 2 つのトレイルに分割します。範囲計算の基準にする列が定義されていないため、Oracle GoldenGate は主キー列を使用します。

```
RMTTRAIL /ggs/dirdat/aa
TABLE fin.account, FILTER (@RANGE (1, 2));
RMTTRAIL /ggs/dirdat/bb
TABLE fin.account, FILTER (@RANGE (2, 2));
```

例 3 次の例では、2 つの表で order_ID 列に基づく関連操作が行われます。order_master 表は、キー order_ID を持ち、order_detail 表は、キー order_ID および item_number を持ちます。キー order_ID によって関係が確立されるため、参照整合性を維持するために、両方の表の @RANGE フィルタでこのキーが使用されます。ロードは 2 つの範囲に分割されます。

(パラメータ・ファイル #1)

```
MAP sales.order_master, TARGET sales.order_master,
FILTER (@RANGE (1, 2, order_ID));
MAP sales.order_detail, TARGET sales.order_detail,
FILTER (@RANGE (1, 2, order_ID));
```

(パラメータ・ファイル #2)

```
MAP sales.order_master, TARGET sales.order_master,
FILTER (@RANGE (2, 2, order_ID));
MAP sales.order_detail, TARGET sales.order_detail,
FILTER (@RANGE (2, 2, order_ID));
```

STRCAT

@STRCAT ファンクションでは、1 つ以上の文字列または文字列 (文字) 用の列を連結します。リテラル文字列は引用符で囲みます。

Oracle GoldenGate では、このファンクションで、Unicode、および Microsoft Windows、UNIX、Linux オペレーティング・システムのネイティブ・エンコーディングの文字列を格納する列内の文字を表すために、エスケープ・シーケンスの使用をサポートしています。引数が Unicode で指定される場合、ターゲット列は SQL Unicode データ型である必要があります。

構文

```
@STRCAT (<string1>, <string2> [, ...])
```

引数	説明
<string1>	連結する最初の列またはリテラル文字列。
<string2>	連結する次の列またはリテラル文字列。

例 次の例では、3 つの列から電話番号を作成し、リテラル・フォーマット値を含めます。

```
PHONE_NO = @STRCAT (AREA_CODE, PREFIX, "-", PHONE)
```

STRCMP

@STRCMP ファンクションでは、2 つの文字用の列またはリテラル文字列を比較します。リテラルは引用符で囲みます。

@STRCMP は、次を返します。

- 1 (最初の文字列が 2 つ目の文字列より小さい場合)
- 0(2 つの文字列が等しい場合)
- 1(最初の文字列が 2 つ目の文字列より大きい場合)

文字列の比較の前に、末尾の空白が切り捨てられます。

Oracle GoldenGate では、このファンクションで、Unicode、および Microsoft Windows、UNIX、Linux オペレーティング・システムのネイティブ・エンコーディングの文字列を格納する列内の文字を表すために、エスケープ・シーケンスの使用をサポートしています。このファンクションは、CHAR と NCHAR など、異なる文字データ型を比較できます。

このファンクションは、NCHAR または NVARCHAR データ型をサポートしていません。

構文

```
@STRCMP (<string1>, <string2>)
```

引数	説明
<string1>	比較する最初の列またはリテラル文字列。
<string2>	比較する 2 つ目の列またはリテラル文字列。

例

次の例では、2 つのリテラル文字列を比較し、最初の文字列が 2 つ目の文字列より大きいために 1 が返されます。

```
@STRNCMP ("JOHNSON", "JONES")
```

STREQ

@STREQ ファンクションでは、2 つの文字列 (文字) 用の列またはリテラル文字列が等しいかどうかを確認します。リテラルは引用符で囲みます。@STREQ は、次を返します。

- 1(true)(2 つの文字列が等しい場合)
- 0(false)(2 つの文字列が等しくない場合)

Oracle GoldenGate では、このファンクションで、Unicode、および Microsoft Windows、UNIX、Linux オペレーティング・システムのネイティブ・エンコーディングの文字列を格納する列内の文字を表すために、エスケープ・シーケンスの使用をサポートしています。このファンクションは、CHAR と NCHAR など、異なる文字データ型を比較できます。

このファンクションは、NCHAR または NVARCHAR データ型をサポートしていません。

構文

```
@STREQ (<string1>, <string2>)
```

引数	説明
<string1>	比較する最初の列またはリテラル文字列。
<string2>	比較する 2 つ目の列またはリテラル文字列。

例

次の例では、region 列の値と、リテラル値 "EAST" を比較します。region = EAST の場合、レコードはフィルタに渡されます。

```
FILTER (@STREQ (region, "EAST"))
```

次の例に示すように、@STREQ は結果を決定するための比較にも使用できます。state が "NY" の場合、式は "East Coast" を返します。それ以外の場合は、"Other" を返します。

```
@IF (@STREQ (state, "NY"), "East Coast", "Other")
```

STREXT

@STREXT ファンクションでは、文字列の一部を抽出します。

Oracle GoldenGate では、このファンクションで、Unicode、および Microsoft Windows、UNIX、Linux オペレーティング・システムのネイティブ・エンコーディングの文字列を格納する列内の文字を表すために、エスケープ・シーケンスの使用をサポートしています。引数が Unicode で指定される場合、ターゲット列は SQL Unicode データ型である必要があります。

構文

```
@STREXT (<string>, <begin position>, <end position>)
```

引数	説明
<string>	抽出元の文字列。この文字列には、文字の列の名前またはリテラル文字列を指定できます。リテラルは引用符で囲みます。
<begin position>	抽出を開始する文字の位置。
<end position>	抽出を終了する文字の位置。終了位置は抽出に含まれます。

次の例では、@STREXT ファンクションを使用して、電話番号を 3 つの異なる列に抽出します。

```
AREA_CODE = @STREXT (PHONE, 1, 3),
PREFIX = @STREXT (PHONE, 4, 6),
PHONE_NO = @STREXT (PHONE, 7, 10)
```

STRFIND

@STRFIND ファンクションでは、文字列用の列内の文字列の位置を確認し、文字列が見つからないときはゼロを返します。オプションで、@STRFIND は文字列内の開始位置を受け付けます。

Oracle GoldenGate では、このファンクションで、Unicode、および Microsoft Windows、UNIX、Linux オペレーティング・システムのネイティブ・エンコーディングの文字列を格納する列内の文字を表すために、エスケープ・シーケンスの使用をサポートしています。引数が Unicode で指定される場合、ターゲット列は SQL Unicode データ型である必要があります。

このファンクションは、NCHAR または NVARCHAR データ型をサポートしていません。

構文

```
@STRFIND (<string>, "search string>" [, < begin position>])
```

引数	説明
`<string>`	検索元の文字列。ここには、文字用の列名またはリテラル文字列を指定できます。リテラルは引用符で囲みます。
`"<search string>"`	検索する文字列。検索する文字列は引用符で囲みます。
`<begin position>`	検索を開始する文字の位置。

例 ACCT 列の文字列を ABC123ABC とすると、次のような結果になります。

ファンクション文	結果
@STRFIND (ACCT, "23")	5
@STRFIND (ACCT, "ZZ")	0
@STRFIND (ACCT, "ABC", 2)	7(検索が 2 番目の文字から開始されたため)

STRLEN

@STRLEN ファンクションでは、文字数で表された文字列長を返します。

Oracle GoldenGate では、このファンクションで、Unicode、および Microsoft Windows、UNIX、Linux オペレーティング・システムのネイティブ・エンコーディングの文字列を格納する列内の文字を表すために、エスケープ・シーケンスの使用をサポートしています。引数が Unicode で指定される場合、ターゲット列は SQL Unicode データ型である必要があります。

このファンクションは、NCHAR または NVARCHAR データ型をサポートしていません。

構文

```
@STRLEN (<string>)
```

引数	説明
<string>	文字列 (文字) 用の列名、またはリテラル文字列。リテラルは引用符で囲みます。

例 `@STRLEN (ID_NO)`

STRLTRIM

@STRLTRIM ファンクションでは、先行する空白を切り捨てます。

Oracle GoldenGate では、このファンクションで、Unicode、および Microsoft Windows、UNIX、Linux オペレーティング・システムのネイティブ・エンコーディングの文字列を格納する列内の文字を表すために、エスケープ・シーケンスの使用をサポートしています。引数が Unicode で指定される場合、ターゲット列は SQL Unicode データ型である必要があります。

構文 `@STRLTRIM (<string>)`

引数	説明
<string>	文字用の列名、またはリテラル文字列。リテラルは引用符で囲みます。

例 `birth_state = @strltrim (state)`

STRNCAT

@STRCAT ファンクションでは、1 つ以上の文字列を最大長まで連結します。

Oracle GoldenGate では、このファンクションで、Unicode、および Microsoft Windows、UNIX、Linux オペレーティング・システムのネイティブ・エンコーディングの文字列を格納する列内の文字を表すために、エスケープ・シーケンスの使用をサポートしています。引数が Unicode で指定される場合、ターゲット列は SQL Unicode データ型である必要があります。

構文 `@STRNCAT (<string>, <max length> [, <string>, <max length>] [, ...] )`

引数	説明
<string>	文字列 (文字) 用の列名、またはリテラル文字列。リテラルは引用符で囲みます。
<max length>	最大文字列長 (文字数)。

例 次の例では、2 つの文字列を連結し、結果が "ABC123" になります。

```
PHONE_NO = @STRNCAT ("ABCDEF", 3, "123456", 3)
```

STRNCMP

@STRNCMP ファンクションでは、特定の文字数に基づいて 2 つの文字列を比較します。文字列には、文字列 (文字) 用の列名、または引用符で囲んだリテラル文字列を指定できます。比較は、文字列の最初の文字から開始されます。

@STRNCMP は、次を返します。

- 1 (最初の文字列が 2 つ目の文字列より小さい場合)
- 0(2 つの文字列が等しい場合)
- 1(最初の文字列が 2 つ目の文字列より大きい場合)

このファンクションは、NCHAR または NVARCHAR データ型をサポートしていません。

構文

```
@STRNCMP (<string1>, <string2>, <max length>)
```

引数	説明
<string1>	比較する最初の文字列。
<string2>	比較する 2 つ目の文字列。
<max length>	文字列内の比較する最大文字数。

例

次の例では、<max length> で 2 が指定されているため、各文字列の最初の 2 文字を比較し、2 つの文字セットが同一なので 0 を返します。

```
@STRNCMP ("JOHNSON", "JONES", 2)
```

STRNUM

@STRNUM ファンクションでは、数字を文字列に変換し、出力形式および埋込みを指定します。

構文

```
@STRNUM (<column>, {LEFT | LEFTSPACE, | RIGHT | RIGHTZERO} [<length>] )
```

引数	説明
<column>	ソース数字列名。
LEFT	左に寄せます (埋込みなし)。
LEFTSPACE	左に寄せ、ターゲット列の残りを空白で埋めます。
RIGHT	右に寄せ、ターゲット列の残りを空白で埋めます。列の値が負の値の場合、空白はマイナス記号の前に追加されます。たとえば、ターゲット列で 7 桁が許可されているとすると、列値 -1.27 に strnum(Col1, right) を使用した場合の結果は、###-1.27 になります。マイナス記号は桁数にカウントされませんが、10 進数はカウントされます。

引数	説明
RIGHTZERO	右に寄せ、ターゲット列の残りをゼロで埋めます。列の値が負の値の場合、ゼロはマイナス記号の後ろで数値の前に追加されます。たとえば、ターゲット列で 7 桁が許可されているとすると、列値 -1.27 に strnum(Col1, rightzero) を使用した場合の結果は、-0001.27 になります。マイナス記号は桁数にカウントされませんが、10 進数はカウントされます。
<length>	埋込みを指定するオプション (LEFT 以外) を使用しているときに、出力長を指定します。次に例を示します。 ◆ 列値 -1.27 に strnum(Col1, right, 6) を使用した場合の結果は、##-1.27 になります。マイナス記号は桁数にカウントされませんが、10 進数はカウントされます。 ◆ 列値 -1.27 に strnum(Col1, rightzero, 6) が使用されている場合は、-001.27 になります。マイナス記号は桁数にカウントされませんが、10 進数はカウントされます。

例 次に、ソース列 NUM の値が 15 で、ターゲット列の最大長が 5 文字の場合に、フォーマット・オプションを使用して様々な結果を取得する例を示します。

ファンクション文	**結果** (# は空白を表します)
`CHAR1 = @STRNUM (NUM, LEFT)`	`15`
`CHAR1 = @STRNUM (NUM, LEFTSPACE)`	`15###`
`CHAR1 = @STRNUM (NUM, RIGHTZERO)`	`00015`
`CHAR1 = @STRNUM (NUM, RIGHT)`	`###15`

上記の例で出力の <length> を 4 とすると、次のように異なるタイプの結果が返されます。

ファンクション文	**結果** (# は空白を表します)
`CHAR1 = @STRNUM (NUM, LEFTSPACE, 4)`	`15##`
`CHAR1 = @STRNUM (NUM, RIGHTZERO, 4)`	`0015`
`CHAR1 = @STRNUM (NUM, RIGHT, 4)`	`##15`

STRRTRIM

@STRRTRIM ファンクションでは、末尾の空白を切り捨てます。

Oracle GoldenGate では、このファンクションで、Unicode、および Microsoft Windows、UNIX、Linux オペレーティング・システムのネイティブ・エンコーディングの文字列を格納する列内の文字を表すために、エスケープ・シーケンスの使用をサポートしています。引数が Unicode で指定される場合、ターゲット列は SQL Unicode データ型である必要があります。

構文

```
@STRRTRIM (<string>)
```

引数	説明
<string>	文字用の列名、またはリテラル文字列。リテラルは引用符で囲みます。

例 `street_address = @strrtrim (address)`

STRSUB

@STRSUB ファンクションでは、文字列 (文字) 用の列または定数内の文字を置換します。

Oracle GoldenGate では、このファンクションで、Unicode、および Microsoft Windows、UNIX、Linux オペレーティング・システムのネイティブ・エンコーディングの文字列を格納する列内の文字を表すために、エスケープ・シーケンスの使用をサポートしています。引数が Unicode で指定される場合、ターゲット列は SQL Unicode データ型である必要があります。

このファンクションは、NCHAR または NVARCHAR データ型をサポートしていません。

構文

```
@STRSUB
(<source string>, <search string>, <substitute string>
[, <search string>, <substitute string>] [, ...])
```

引数	説明
<source string>	置換される文字を含むソース文字列または列。
<search string>	置換される文字列。
<substitute string>	検索した文字列と置換する文字列。

例 1 次の例では、xxABCxx が返されます。

```
@STRSUB ("123ABC123", "123", "xx")
```

例 2 次の例では、023zBC023 が返されます。

```
@STRSUB ("123ABC123", "A", "z", "1", "0")
```

STRTRIM

@STRTRIM ファンクションでは、先行する空白と末尾の空白を切り捨てます。

Oracle GoldenGate では、このファンクションで、Unicode、および Microsoft Windows、UNIX、Linux オペレーティング・システムのネイティブ・エンコーディングの文字列を格納する列内の文字を表すために、エスケープ・シーケンスの使用をサポートしています。引数が Unicode で指定される場合、ターゲット列は SQL Unicode データ型である必要があります。

構文 `@STRTRIM (<string>)`

引数	説明
<string>	文字用の列名、またはリテラル文字列。リテラルは引用符で囲みます。

例 `pin_no = @strtrim (custpin)`

STRUP

@STRUP ファンクションでは、英数字文字列または文字列 (文字) 用の列を大文字に変換します。

Oracle GoldenGate では、このファンクションで、Unicode、および Microsoft Windows、UNIX、Linux オペレーティング・システムのネイティブ・エンコーディングの文字列を格納する列内の文字を表すために、エスケープ・シーケンスの使用をサポートしています。引数が Unicode で指定される場合、ターゲット列は SQL Unicode データ型である必要があります。

このファンクションは、NCHAR または NVARCHAR データ型をサポートしていません。

構文 `@STRUP (<string>)`

引数	説明
<string>	文字用の列名、またはリテラル文字列。リテラルは引用符で囲みます。

例 次の例では、"SALESPERSON" が返されます。

```
@STRUP ("salesperson")
```

TOKEN

@TOKEN ファンクションでは、Oracle GoldenGate レコード・ヘッダーのユーザー・トークン・エリアに保持されているトークン・データを取得します。トークン・データをターゲット列にマップするには、COLMAP 句のソース式で @TOKEN を使用します。そのかわりの方法としては、@TOKEN を SQLEXEC 文、Oracle GoldenGate マクロ、またはユーザー・イグジットで使用できます。

トークン・データを定義するには、Extract パラメータ・ファイルの TABLE 文で TOKENS 句を使用します。トークンの使用の詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。

構文 `@TOKEN ("<token name>")`

引数	説明
"<token name>"	引用符で囲まれた、データを取得するトークン名。

例　次の例では、10 のトークンをターゲット列にマップします。

```
MAP ora.oratest, TARGET ora.rpt,
COLMAP (
host = @token ("tk_host"),
gg_group = @token ("tk_group"),
osuser = @token ("tk_osuser"),
domain = @token ("tk_domain"),
ba_ind = @token ("tk_ba_ind"),
commit_ts = @token ("tk_commit_ts"),
pos = @token ("tk_pos"),
rba = @token ("tk_rba"),
tablename = @token ("tk_table"),
optype = @token ("tk_optype")
);
```

VALONEOF

@VALONEOF ファンクションでは、文字列または文字列 (文字) 用の列を値のリストと比較します。値または列がリストにある場合は 1 が返され、それ以外の場合は 0 が返されます。

Oracle GoldenGate では、このファンクションで、Unicode、および Microsoft Windows、UNIX、Linux オペレーティング・システムのネイティブ・エンコーディングの文字列を格納する列内の文字を表すために、エスケープ・シーケンスの使用をサポートしています。引数が Unicode で指定される場合、ターゲット列は SQL Unicode データ型である必要があります。

このファンクションは、NCHAR または NVARCHAR データ型をサポートしていません。

構文　`@VALONEOF (<expression>, <value> [, <value>] [, ...])`

引数	説明
`<expression>`	文字用の列名、または引用符で囲まれたリテラル文字列。
`<value>`	基準値。

例　次の例では、STATE が CA または NY の場合、値が非ゼロ (true) のときに @IF によって返されるレスポンスである "COAST" が返されます。それ以外の場合は、"MIDDLE" が返されます。

```
@IF (@VALONEOF (STATE, "CA", "NY"), "COAST", "MIDDLE")
```

第 6 章
ユーザー・イグジット・ファンクション

この章では、Oracle GoldenGate ユーザー・イグジット・ファンクションとその構文について説明します。Oracle GoldenGate ユーザー・イグジットの使用の詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。

ユーザー・イグジットのコール

C プログラミング・コードでユーザー・イグジットを記述します。CUSEREXIT パラメータを使用して、Oracle GoldenGate プロセス内の定義済イグジット・ポイントで、Windows DLL または UNIX 共有オブジェクトから、ユーザー・イグジットをコールします。ユーザー・イグジット・ルーチンは、Extract および Replicat プロセスから様々なイベントおよび情報を受け入れ、リクエストどおりに情報を処理し、コール元 (このルーチンをコールした Oracle GoldenGate プロセス) にレスポンスと情報を返す必要があります。CUSEREXIT パラメータの情報および構文は、153 ページを参照してください。

ユーザー・イグジット・ファンクションの概要

パラメータ	説明
ERCALLBACK	コールバック・ルーチンを実装します。コールバック・ルーチンは、レコードおよび Oracle GoldenGate コンテキスト情報を取得し、データ・レコードの内容を変更します。
EXIT_CALL_RESULT	ルーチンにレスポンスを提供します。
EXIT_CALL_TYPE	処理中にいつルーチンをコールするかを指定します。
EXIT_PARAMS	ルーチンに情報を提供します。

EXIT_CALL_TYPE の使用

EXIT_CALL_TYPE では、処理中にいつ Extract または Replicat プロセス (コール元) がユーザー・イグジット・ルーチンをコールするかを指定します。プロセスは、次のコールを使用してルーチンをコールできます。

表 51　ユーザー・イグジット・コール

コール・タイプ	処理ポイント
EXIT_CALL_ABORT_TRANS	RECOVERYOPTIONS モードが APPEND (デフォルト) の場合に有効です。データ・ポンプまたは Replicat がトレイルから RESTART ABEND レコードを読み取る際にコールされ、異常終了したライター・プロセスによってそこに配置されます。(ライター・プロセスは、データ・ポンプにより読み取られたローカル・トレイルに書き込むプライマリ Extract、または Replicat により読み取られたリモート・トレイルに書き込むデータ・ポンプになります。) このコール・タイプによって、ユーザー・イグジットは、ライター・プロセスの停止時に未完了のままのトランザクションを中止または破棄し、以前に完了したトランザクションの開始時の処理をリカバリおよび再開できます。
EXIT_CALL_BEGIN_TRANS	次のいずれかの直前にコールされます。 ◆ データ・ポンプにより読み取られるトランザクションの BEGIN レコード ◆ Replicat トランザクションの開始
EXIT_CALL_CHECKPOINT	Extract または Replicat チェックポイントが書き込まれた直後にコールされます。
EXIT_CALL_DISCARD_ASCII_RECORD	Extract の処理中に、ASCII 入力レコードが破棄ファイルに書き込まれる前にコールされます。関連する ASCII バッファは、コールバック・ルーチンを使用してユーザー・イグジットによって取得および操作できます。 このコール・タイプは、Replicat プロセスでは使用できません。
EXIT_CALL_DISCARD_RECORD	Replicat の処理中に、レコードが破棄ファイルに書き込まれる前にコールされます。レコードは、Oracle GoldenGate 変更レコードの値がターゲット表の現在のバージョンと異なっているときなど、複数の理由で破棄されます。関連する破棄バッファは、コールバック・ルーチンを使用してユーザー・イグジットによって取得および操作できます。 このコール・タイプは、Extract プロセスでは使用できません。
EXIT_CALL_END_TRANS	次のいずれかの直後にコールされます。 ◆ データ・ポンプにより読み取られるトランザクションの END レコード ◆ Replicat トランザクションの最後のレコード
EXIT_CALL_FATAL_ERROR	Extract または Replicat の処理中に致命的なエラーが発生後、Oracle GoldenGate が停止する直前にコールされます。

表 51　ユーザー・イグジット・コール (続き)

コール・タイプ	処理ポイント
EXIT_CALL_PROCESS_MARKER	Replicat の処理中に、NonStop サーバーのマーカーがトレイルから読み取られ、マーカー履歴ファイルに書き込まれる前にコールされます。
EXIT_CALL_PROCESS_RECORD	◆ Extract の場合は、レコード・バッファがトレイルに出力される前にコールされます。 ◆ Replicat の場合は、レプリケートされた操作が実行される直前にコールされます。 このコールは、ほとんどのユーザー・イグジット処理の基盤です。EXIT_CALL_PROCESS_RECORD がコールされると、レコード・バッファおよびその他のレコード情報がコールバック・ルーチンに使用可能になります。ソースとターゲットのマッピングがパラメータ・ファイルで指定されている場合、マッピングは EXIT_CALL_PROCESS_RECORD イベントが発生する前に実行されます。ユーザー・イグジットは、レコード内の他のすべての操作を、マップ、変換、削除、または実行できます。ユーザー・イグジットは、コール元がレコードを処理または無視する必要があるかを示すステータスを返すことができます。
EXIT_CALL_START	処理開始時にコールされます。ユーザー・イグジットは、ファイルのオープンや変数の初期化などの初期化作業を実行できます。
EXIT_CALL_STOP	プロセスが正常に停止または異常終了する前にコールされます。ユーザー・イグジットは、ファイルのクローズや合計の出力などの完了作業を実行できます。
EXIT_CALL_RESULT	各イグジット・コールの完了時に、コール元にレスポンス方法を指示するためにユーザー・イグジット・ルーチンによって設定されます。

EXIT_CALL_RESULT の使用

EXIT_CALL_RESULT では、ルーチンにレスポンスを提供します。

表 52　ユーザー・イグジット・レスポンス

コール結果	説明
EXIT_ABEND_VAL	コール元に即座に停止するように指示します。
EXIT_IGNORE_VAL	レコードのそれ以上の処理を拒否します。EXIT_IGNORE_VAL は、ユーザー・イグジットが特定のレコードに必要なすべての処理を実行し、データ・レコードの出力やレプリケートが必要ないときに適切です。

表 52　ユーザー・イグジット・レスポンス (続き)

コール結果	説明
EXIT_OK_VAL	ルーチンがイベントに対して何も実行しない場合は、EXIT_OK_VAL とみなされます。イグジット・コール・タイプが次のいずれかの場合 EXIT_CALL_PROCESS_RECORD EXIT_CALL_DISCARD_RECORD EXIT_CALL_DISCARD_ASCII_RECORD ... かつ EXIT_OK_VAL が返される場合、Oracle GoldenGate はユーザー・イグジットから返されたレコード・バッファを処理します。
EXIT_PROCESSED_REC_VAL	Extract または Replicat に、レコードをスキップし、レポート・ファイルに出力される統計の該当する表および操作タイプを更新するように指示します。
EXIT_STOP_VAL	コール元に処理を正常に停止するように指示します。EXIT_STOP_VAL または EXIT_ABEND_VAL は、ユーザー・イグジットでエラー状態が発生したときに適切です。

EXIT_PARAMS の使用

EXIT_PARAMS では、ユーザー・イグジット・ルーチンにプログラム名やユーザー定義パラメータなどの情報を提供します。単一のデータ・レコードを複数回処理できます。

表 53　ユーザー・イグジット入力

イグジット・パラメータ	説明
PROGRAM_NAME	コール元プロセスの完全パスおよび名前を指定します (\ggs\extract や \ggs\replicat など)。このパラメータは、Windows API を使用して Oracle GoldenGate コールバック・ルーチンをロードするときや、ユーザー・イグジットが Extract および Replicat 両方の処理で使用されているときに、コール元プログラムを特定するために使用します。
FUNCTION_PARAM	◆ リテラル文字列のパラメータをユーザー・イグジットに渡すために使用します。パラメータを渡す側である TABLE または MAP 文の EXITPARAM オプションでパラメータを指定します。252 ページを参照してください。これは、特定のレコードを処理するイグジット・コール中にのみ有効です。 ◆ FUNCTION_PARAM は、CUSEREXIT パラメータの PARAMS オプションで指定されているパラメータを渡すために、イグジット・コール起動イベントでも使用できます。(153 ページを参照してください。) これは、イグジットの起動時にグローバル・パラメータを提供するためにのみ有効です。
MORE_RECS_IND	イグジットからのリターンに対して設定します。データベース・レコードに対して、Extract または Replicat プロセスがレコードをもう一度処理するかどうかを決定します。これにより、Enscribe を SQL に変換する (データ正規化) ときの一般的なファンクションで、ユーザー・イグジットは Extract によって処理される各レコードに対して多くのレコードを出力できます。同一のレコードをもう一度リクエストするには、MORE_RECS_IND を CHAR_NO_VAL または CHAR_YES_VAL に設定します。

ERCALLBACK の使用

ERCALLBACK では、コールバック・ルーチンを実行します。ユーザー・コールバック・ルーチンは、Extract または Replicat プロセスからコンテキスト情報を取得し、コール・タイプが次のいずれかのときにレコード自体を含むコンテキスト値を設定します。

- EXIT_CALL_PROCESS_RECORD
- EXIT_CALL_DISCARD_RECORD
- EXIT_CALL_DISCARD_ASCII_RECORD

構文

```
ERCALLBACK (<function_code>, <buffer>, <result_code>);
```

引数	説明
<function_code>	コールバック・ルーチンによって実行されるファンクション。ユーザー・コールバック・ルーチンは、コールバック・ルーチンに渡されるファンクション・コードに基づいて異なる動作をします。一部のファンクションは、Extract および Replicat 両方が使用できますが、各プロセスでのファンクションの有効性は、コールバック・ルーチン中にそのファンクションに対して設定される入力パラメータに依存します。使用可能なファンクション・コードの詳細は、479 ページの「ファンクション・コード」を参照してください。
<buffer>	指定するファンクション・コードに関連付けられている事前定義済構造体を含むバッファへの void ポインタ。
<result_code>	コールバック・ルーチンによって実行されるファンクションのステータス。コールバック・ルーチンによって返される結果コードは、コールバック・ファンクションが成功したかどうかを示します。結果コードは、表 54 の値の 1 つになります。

表 54　結果コード

コード	説明
EXIT_FN_RET_BAD_COLUMN_DATA	列データを取得または設定中に無効なデータを検出しました。
EXIT_FN_RET_BAD_DATE_TIME	列の日付、タイムスタンプまたは間隔タイプに、無効な日付または時刻の値が含まれています。
EXIT_FN_RET_BAD_NUMERIC_VALUE	列の数値タイプに、無効な数値が含まれています。
EXIT_FN_RET_COLUMN_NOT_FOUND	圧縮更新レコードで列が見つかりませんでした。
EXIT_FN_RET_ENV_NOT_FOUND	指定された環境値がレコードに見つかりませんでした。

表 54　結果コード（続き）

コード	説明
EXIT_FN_RET_EXCEEDED_MAX_LENGTH	表または列名が割り当てられたバッファに収まらなかったため、メタデータを取得できませんでした。
EXIT_FN_RET_FETCH_ERROR	レコードをフェッチできませんでした。エラー・メッセージを表示して理由を確認してください。
EXIT_FN_RET_INCOMPLETE_DDL_REC	DDL レコードの処理中に内部エラーが発生しました。レコードが不完全の可能性があります。
EXIT_FN_RET_INVALID_CALLBACK_FNC_CD	無効なコールバック・ファンクション・コードがコールバック・ルーチンに渡されました。
EXIT_FN_RET_INVALID_COLUMN	ファンクション・コールで、存在しない列が参照されました。
EXIT_FN_RET_INVALID_COLUMN_TYPE	ルーチンは、その目的では Oracle GoldenGate にサポートされていないデータ型を操作しようとしています。
EXIT_FN_RET_INVALID_CONTEXT	コールバック・ファンクションが不適切なときにコールされました。
EXIT_FN_RET_INVALID_PARAM	無効なパラメータがコールバック・ファンクションに渡されました。
EXIT_FN_RET_NO_SRCDB_INSTANCE	ソース・データベース・インスタンスが見つかりませんでした。
EXIT_FN_RET_NO_TGTDB_INSTANCE	ターゲット・データベース・インスタンスが見つかりませんでした。
EXIT_FN_RET_NOT_SUPPORTED	このファンクションは、このプロセスに対してサポートされていません。
EXIT_FN_RET_OK	コールバック・ファンクションは成功しました。
EXIT_FN_RET_SESSION_CS_CNV_ERR	文字セット変換ルーチンに ULIB_ERR_INVALID_CHAR_FOUND エラーが返されました。変換は失敗しました。
EXIT_FN_RET_TABLE_NOT_FOUND	無効な表名が指定されました。
EXIT_FN_RET_TOKEN_NOT_FOUND	指定されたトークンがレコードに見つかりませんでした。

ファンクション・コード

ファンクション・コードは、コールバック・ルーチンの出力を決定します。コールバック・ルーチンは、データ・バッファの内容が指定されたファンクション・コードの構造体と一致するとみなします。コールバック・ルーチン・ファンクション・コードとそのデータ・バッファは、次の項で説明します。次に、使用可能なファンクションの概要を示します。

表 55　Oracle GoldenGate ファンクション・コードの概要

ファンクション・コード	説明
COMPRESS_RECORD	COMPRESS_RECORD ファンクションは、マッピングの後にターゲット表の列がすべてでなく一部のみ存在する場合で、個々の列値ではなくレコード全体を操作する必要があるときに使用します。
DECOMPRESS_RECORD	DECOMPRESS_RECORD ファンクションは、マッピングの後にターゲット表の列がすべてでなく一部のみ存在する場合で、個々の列値ではなくレコード全体を操作する必要があるときに使用します。
GET_BEFORE_AFTER_IND	GET_BEFORE_AFTER_IND ファンクションでは、レコードがデータベース操作のビフォア・イメージかアフター・イメージかを確認します。
GET_CATALOG_NAME_ONLY	GET_CATALOG_NAME_ONLY ファンクションでは、データベース・カタログの名前を返します。
GET_COL_METADATA_FROM_INDEX	GET_COL_METADATA_FROM_INDEX ファンクションでは、特定の列索引に関連付けられている列メタデータを確認します。
GET_COL_METADATA_FROM_NAME	GET_COL_METADATA_FROM_NAME ファンクションでは、特定の列名に関連付けられている列メタデータを確認します。
GET_COLUMN_INDEX_FROM_NAME	GET_COLUMN_INDEX_FROM_NAME ファンクションでは、特定の列名に関連付けられている列索引を確認します。
GET_COLUMN_NAME_FROM_INDEX	GET_COLUMN_NAME_FROM_INDEX ファンクションでは、特定の列索引に関連付けられている列名を確認します。
GET_COLUMN_VALUE_FROM_INDEX	GET_COLUMN_VALUE_FROM_INDEX ファンクションでは、指定する列索引を使用して、データ・レコードから列値を返します。
GET_COLUMN_VALUE_FROM_NAME	GET_COLUMN_VALUE_FROM_NAME ファンクションでは、特定の列名を使用して、データ・レコードから列値を返します。

表 55　Oracle GoldenGate ファンクション・コードの概要 (続き)

ファンクション・コード	説明
GET_DATABASE_METADATA	GET_DATABASE_METADATA ファンクションでは、データベース・メタデータを返します。
GET_DDL_RECORD_PROPERTIES	GET_DDL_RECORD_PROPERTIES ファンクションでは、DDL 操作に関する情報を返します。
GET_ENV_VALUE	GET_ENV_VALUE ファンクションでは、Oracle GoldenGate 環境に関する情報を返します。
GET_ERROR_INFO	GET_ERROR_INFO ファンクションでは、破棄レコードに関連付けられているエラー情報を返します。
GET_GMT_TIMESTAMP	GET_GMT_TIMESTAMP ファンクションでは、操作コミット・タイムスタンプを GMT フォーマットで返します。
GET_MARKER_INFO	GET_MARKER_INFO ファンクションでは、データを送信するときにマーカー情報を返します。マーカーは、ユーザー・イグジット内でカスタム処理をトリガーするために使用します。
GET_OPERATION_TYPE	GET_OPERATION_TYPE ファンクションでは、レコードに関連付けられている操作のタイプを確認します。
GET_POSITION	GET_POSITION ファンクションでは、Oracle GoldenGate トレイル内の Extract データ・ポンプまたは Replicat の読取り位置を取得します。
GET_RECORD_BUFFER	GET_RECORD_BUFFER ファンクションでは、カスタム列変換に関する情報を取得します。
GET_RECORD_LENGTH	GET_RECORD_LENGTH ファンクションでは、データ・レコードの長さを返します。
GET_RECORD_TYPE	GET_RECORD_TYPE ファンクションでは、処理されているレコードのタイプを返します。
GET_SCHEMA_NAME_ONLY	GET_SCHEMA_NAME_ONLY ファンクションでは、表のスキーマ名のみを返します。
GET_SESSION_CHARSET	GET_SESSION_CHARSET ファンクションでは、ユーザー・イグジット・セッションの文字セットを返します。
GET_STATISTICS	GET_STATISTICS ファンクションでは、Extract または Replicat プロセスの現在の処理統計を返します。
GET_TABLE_COLUMN_COUNT	GET_TABLE_COLUMN_COUNT ファンクションでは、表内の列の合計数を返します。
GET_TABLE_METADATA	GET_TABLE_METADATA ファンクションでは、処理中のレコードに関連付けられている表のメタデータを返します。

表 55　Oracle GoldenGate ファンクション・コードの概要 (続き)

ファンクション・コード	説明
GET_TABLE_NAME	GET_TABLE_NAME ファンクションでは、処理中のレコードに関連付けられているソースまたはターゲット表の名前を返します。
GET_TABLE_NAME_ONLY	GET_TABLE_NAME_ONLY ファンクションでは、表の名前のみを返します。
GET_TIMESTAMP	GET_TIMESTAMP ファンクションでは、ソース・データ・レコードに関連付けられている I/O タイムスタンプを返します。
GET_TRANSACTION_IND	GET_TRANSACTION_IND ファンクションでは、データ・レコードがトランザクションの最初、最後、または中間の操作かを確認します。
GET_USER_TOKEN_VALUE	GET_USER_TOKEN_VALUE ファンクションでは、トレイル・レコードからユーザー・トークン値を取得します。
OUTPUT_MESSAGE_TO_REPORT	OUTPUT_MESSAGE_TO_REPORT ファンクションでは、レポート・ファイルにメッセージを出力します。
RESET_USEREXIT_STATS	RESET_USEREXIT_STATS ファンクションでは、Oracle GoldenGate プロセスの統計をリセットします。
SET_COLUMN_VALUE_BY_INDEX	SET_COLUMN_VALUE_BY_INDEX ファンクションでは、データ・レコード全体を操作せず、単一の列値のみを変更します。
SET_COLUMN_VALUE_BY_NAME	SET_COLUMN_VALUE_BY_NAME ファンクションでは、データ・レコード全体を操作せず、単一の列値のみを変更します。
SET_OPERATION_TYPE	SET_OPERATION_TYPE ファンクションでは、レコードに関連付けられている操作のタイプを変更します。
SET_RECORD_BUFFER	SET_RECORD_BUFFER ファンクションは、HP NonStop ユーザー・イグジットとの互換性の維持、および複雑なデータ・レコード操作のために使用します。
SET_SESSION_CHARSET	SET_SESSION_CHARSET ファンクションでは、ユーザー・イグジット・セッションの文字セットを設定します。
SET_TABLE_NAME	SET_TABLE_NAME ファンクションでは、レコードに関連付けられている表名を変更します。

COMPRESS_RECORD

適用対象 Extract および Replicat

COMPRESS_RECORD ファンクションでは、DECOMPRESS_RECORD ファンクションで解凍されたレコードを再度圧縮します。COMPRESS_RECORD は、DECOMPRESS_RECORD を使用した*後*にのみ使用します。

レコード・バッファの内容は、ユーザー・イグジットの文字セットとの変換が行われません。これはそのままで渡されます。

構文

```
#include "usrdecs.h"
short result_code;
compressed_rec_def compressed_rec;
ERCALLBACK (COMPRESS_RECORD, &compressed_rec, &result_code);
```

バッファ

```
typedef struct
{
char *compressed_rec;
long compressed_len;
char *decompressed_rec;
long decompressed_len;
short *columns_present;
short source_or_target;
char requesting_before_after_ind;
} compressed_rec_def;
```

入力 次のようになります。

入力	説明
decompressed_rec	圧縮前レコードが含まれるバッファへのポインタ。レコードは、デフォルトの Oracle GoldenGate 正規フォーマットとみなされます。
decompressed_len	解凍されたレコードの長さ。
source_or_target	ソースまたはターゲットのどちらのレコードが圧縮されているかを示す次の一方。 EXIT_FN_SOURCE_VAL EXIT_FN_TARGET_VAL
requesting_before_after_ind	内部入力として使用されます。設定は不要です。設定しても無視されます。
columns_present	圧縮レコード内に存在する列を示す値の配列。たとえば、圧縮レコードに 1 番目、3 番目、6 番目の列が存在し、表内の列数の合計が 7 の場合、配列に次を含む必要があります。 1, 0, 1, 0, 0, 1, 0 表内の列数を取得するには、GET_TABLE_COLUMN_COUNT ファンクションを使用します (522 ページを参照してください)。

出力　次のようになります。

出力	説明
compressed_rec	圧縮フォーマットで返されるレコードへのポインタ。通常、compressed_rec はタイプ exit_rec_buf_def のバッファへのポインタです。exit_rec_buf_def バッファには、Extract または Replicat によってまもなく処理される実際のレコードが含まれます。このバッファは、コール・タイプが EXIT_CALL_DISCARD_RECORD のときに指定されます。イグジット・ルーチンは、たとえばカスタム・マッピング・ファンクションを実行するために、このバッファの内容を変えることがあります。コール元は、compressed_rec に割り当てる適切な量のメモリーを確保する必要があります。
compressed_len	返される圧縮レコードの長さ。

返される値　EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_OK
EXIT_FN_RET_INVALID_PARAM

DECOMPRESS_RECORD

適用対象　Extract および Replicat

DECOMPRESS_RECORD ファンクションは、GET_RECORD_BUFFER (513 ページを参照してください) または SET_RECORD_BUFFER ファンクション (538 ページを参照してください) で更新レコード全体を取得または操作する必要があるものの、レコードが圧縮されている場合に使用します。DECOMPRESS_RECORD は、レコードを論理列レイアウトに配置することにより、圧縮レコードの処理とマップを容易にします。存在する列は、索引および長さインジケータ (「圧縮レコード・フォーマット」を参照してください) なしで所定の位置に配置されます。行方不明の列は、ゼロとして表現されます。DECOMPRESS_RECORD を使用するときは、他の操作が発生する前に呼び出す必要があります。ユーザー・イグジットが処理を完了したら、Oracle GoldenGate プロセスに返す前に、COMPRESS_RECORD ファンクション (482 ページを参照してください) を使用してレコードを再度圧縮してください。

このファンクションは、UPDATE 操作の処理にのみ有効です。削除、挿入、および更新は、バッファに完全なレコード・イメージで保持されます。

レコード・バッファの内容は、ユーザー・イグジットの文字セットとの変換が行われません。これはそのままで渡されます。

圧縮レコード・フォーマット

圧縮 SQL 更新は、次のフォーマットを持ちます。

```
<index><length><value>[<index><length><value>][...]
```

条件 :

- <index> は、表の列のリストへの 2 バイトの索引 (最初の列はゼロ) です。
- <length> は、表の 2 バイトの長さです。
- <value> は、実際の列値で、適切な場合に次の 2 バイトの NULL インジケータの 1 つが含まれます。`0` は非 NULL です。`-1` は NULL です。

構文

```
#include "usrdecs.h"
short result_code;
compressed_rec_def compressed_rec;
ERCALLBACK (DECOMPRESS_RECORD, &compressed_rec, &result_code);
```

バッファ

```
typedef struct
{
char *compressed_rec;
long compressed_len;
char *decompressed_rec;
long decompressed_len;
short *columns_present;
short source_or_target;
char requesting_before_after_ind;
} compressed_rec_def;
```

入力　次のようになります。

入力	説明
compressed_rec	圧縮フォーマットのレコードへのポインタ。この値を取得するには、GET_RECORD_BUFFER ファンクションを使用します (513 ページを参照してください)。
compressed_len	圧縮レコードの長さ。この値を取得するには、GET_RECORD_BUFFER (513 ページを参照してください) または GET_RECORD_LENGTH (516 ページを参照してください) ファンクションを使用します。
source_or_target	ソースまたはターゲットのどちらのレコードが解凍されるかを示す次の一方。 EXIT_FN_SOURCE_VAL EXIT_FN_TARGET_VAL
requesting_before_after_ind	内部入力として使用されます。設定は不要です。設定しても無視されます。

出力　次のようになります。

出力	説明
decompressed_rec	解凍済フォーマットで返されるレコードへのポインタ。レコードは、Oracle GoldenGate 内部正規フォーマットとみなされます。コール元は、decompressed_rec に割り当てる適切な量のメモリーを確保する必要があります。
decompressed_len	返される解凍済レコードの長さ。

出力	説明
columns_present	圧縮レコード内に存在する列を示す値の配列。たとえば、圧縮レコードに 1 番目、3 番目、6 番目の列が存在し、表内の列数の合計が 7 の場合、配列に次を含む必要があります。 1, 0, 1, 0, 0, 1, 0 マッピング・ファンクションは、この配列によって圧縮列をマップするかどうかとその時期を決定します。

返される値

```
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_OK
EXIT_FN_RET_INVALID_PARAM
```

GET_BEFORE_AFTER_IND

適用対象 Extract および Replicat

GET_BEFORE_AFTER_IND ファンクションでは、レコードがデータベース操作のビフォア・イメージかアフター・イメージかを確認します。挿入の場合はアフター・イメージ、削除の場合はビフォア・イメージ、更新の場合はアフターまたはビフォア・イメージのいずれかになります (Extract および Replicat パラメータの GETUPDATEBEFORES および GETUPDATEAFTERS を参照してください)。更新のビフォア・イメージが抽出される場合は、同一の更新内でビフォア・イメージがアフター・イメージよりも先行します。

構文

```
#include "usrdecs.h"
short result_code;
record_def record;
ERCALLBACK (GET_BEFORE_AFTER_IND, &record, &result_code);
```

バッファ

```
typedef struct
{
char *table_name;
char *buffer;
long length;
char before_after_ind;
short io_type;
short record_type;
short transaction_ind;
int64_t timestamp;
exit_ts_str io_datetime;
short mapped;
short source_or_target;
/* Version 2 CALLBACK_STRUCT_VERSION   */
char requesting_before_after_ind;
} record_def;
```

入力 なし

出力 次のようになります。

出力	説明
before_after_ind	レコードがビフォア・イメージかアフター・イメージかを示す次の一方。 BEFORE_IMAGE_VAL AFTER_IMAGE_VAL

返される値

EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_OK

GET_CATALOG_NAME_ONLY

適用対象

Extract および Replicat

GET_CATALOG_NAME_ONLY ファンクションでは、処理中のレコードに関連付けられているソースまたはターゲット表の、スキーマや名前ではなく、カタログを取得します。完全修飾名を返すには、次を参照してください。

GET_TABLE_NAME

表名の他の部分を返すには、次を参照してください。

GET_TABLE_NAME_ONLY

GET_SCHEMA_NAME_ONLY

データベース・オブジェクト名は、大 / 小文字の区別を含め、ホストしているデータベースで定義されているとおりに、正確に返されます。

構文

```
#include "usrdecs.h"
short result_code;
env_value_def env_value;
ERCALLBACK (GET_TABLE_NAME, &env_value, &result_code);
```

バッファ

```
typedef struct
{
char *buffer;
long max_length;
long actual_length;
short value_truncated;
short index;
short source_or_target;
} env_value_def;
```

入力

次のようになります。

入力	説明
buffer	返されたカタログ名を受け付けるバッファへのポインタ。名前は NULL で終了します。 ユーザー・イグジットの文字セッションが、SET_SESSION_CHARSET を使用して、オペレーティング・システムのデフォルトの文字セット以外の値に設定されている場合、ucharset.h ファイルの ULIB_CS_DEFAULT で定義されているように、カタログ名はセッションの文字セットで解釈されます。
max_length	名前を受け付けるために割り当てたバッファの最大長。これは NULL 終了文字列として返されます。
source_or_target	ソースまたはターゲット表のどちらのカタログを返すかを示す次の一方。 EXIT_FN_SOURCE_VAL EXIT_FN_TARGET_VAL

出力　次のようになります。

出力	説明
buffer	NULL で終了する完全修飾カタログ名。
actual length	返される名前の文字列長。実際の長さに NULL 終了文字は含まれません。
value_truncated	値が切り捨てられたかどうかを示すフラグ (0 または 1)。切捨ては、カタログ名と NULL 終了文字を足した長さが、最大バッファ長を超えるときに行われます。

返される値
```
EXIT_FN_RET_INVALID_COLUMN
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_INVALID_PARAM
EXIT_FN_RET_OK
```

GET_COL_METADATA_FROM_INDEX

適用対象　Extract および Replicat

GET_COL_METADATA_FROM_INDEX ファンクションでは、該当の列の索引を指定して列メタデータを取得します。

データベース・オブジェクト名は、大 / 小文字の区別を含め、ホストしているデータベースで定義されているとおりに、正確に返されます。

構文
```
#include "usrdecs.h"
short result_code;
col_metadata_def column_meta_rec;
ERCALLBACK (GET_COL_METADATA_FROM_INDEX, &column_meta_rec, &result_code);
```

バッファ

```
typedef struct
{
    short column_index;
    char *column_name;
    long max_name_length;
    short native_data_type;
    short gg_data_type;
    short gg_sub_data_type;
    short is_nullable;
    short is_part_of_key;
    short key_column_index;
    short length;
    short precision;
    short scale;
    short source_or_target;
} col_metadata_def;
```

入力

次のようになります。

入力	説明
column_index	返される列値の列索引。
max_name_length	返される列名の最大長。通常、最大長は名前のバッファの長さです。返される名前は NULL で終了しているため、最大長は列名の最大長と同じになるはずです。
source_or_target	ソースまたはターゲットのどちらのレコードが圧縮されているかを示す次のいずれか。 EXIT_FN_SOURCE_VAL EXIT_FN_TARGET_VAL

出力

次のようになります。

出力	説明
column_name	返される列値の列名。
native_data_type	列の (データベースに) ネイティブなデータ型。プロセスに応じて、次のように native_data_type または dd_data_type が返されます。 ◆ Extract がソース列に対してコールバック・リクエストを行っている場合は、native_data_type が返されます。Extract がマップされたターゲット列をリクエストしている場合は、(システム上にターゲット定義があるとみなされ)gg_data_type が返されます。

出力	説明
	◆ Extract データ・ポンプがソース列に対してコールバック・リクエストを行い、ローカル・データベースが存在する場合は、native_data_type が返されます。データベースがない場合は、(システム上にソース定義ファイルがあるとみなされ)gg_data_type が返されます。データ・ポンプがマップされたターゲット列をリクエストしている場合は、(システム上にターゲット定義が存在するとみなされ)gg_data_type が返されます。 ◆ Replicat がソース列に対してコールバック・リクエスを行っている場合は、gg_data_type が返されます (システム上にソース定義が存在するとみなされます)。Replicat がソース列をリクエストし、パラメータ・ファイルで ASSUMETARGETDEFS が使用されている場合は、native_data_type が返されます。Replicat がターゲット列をリクエストしている場合は、native_data_type が返されます。
gg_data_type	列の Oracle GoldenGate データ型。
gg_sub_data_type	列の Oracle GoldenGate サブデータ型。
is_nullable	列が NULL 値を許可するかどうかを示すフラグ (TRUE または FALSE)。
is_part_of_key	列が Oracle GoldenGate によって使用されているキーの一部かどうかを示すフラグ (TRUE または FALSE)。

key_column_index	索引での列の順序を示します。たとえば次の表は、主キーで宣言されている順序とは異なる順序で存在する 2 つのキー列を持ちます。
	`CREATE TABLE ABC`
	`(`
	`cust_code VARCHAR2(4),`
	`name VARCHAR2(30),`
	`city VARCHAR2(20),`
	`state CHAR(2),`
	`PRIMARY KEY (city, cust_code)`
	`USING INDEX`
	`);`
	論理列順に各列に対してコールバック・ファンクションを実行すると、次が返されます。
	列名 — **返されるキー索引値**
	cust_code — 1
	name — -1
	city — 0
	state — -1
	
	◆ 列がキーの一部の場合、返される値はキー内の列の順序です。
	◆ 列がキーの一部ではない場合は、値 -1 が返されます。

出力	説明
length	列の長さを返します。
precision	数値データ型の場合、列の精度を返します。
scale	数値データ型の場合、位取りを返します。

返される値

```
EXIT_FN_RET_INVALID_PARAM
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_EXCEEDED_MAX_LENGTH
EXIT_FN_RET_INVALID_COLUMN
EXIT_FN_RET_OK
```

GET_COL_METADATA_FROM_NAME

適用対象 Extract および Replicat

GET_COL_METADATA_FROM_NAME ファンクションでは、該当の列の名前を指定して列メタデータを取得します。ユーザー・イグジットの文字セッションが、SET_SESSION_CHARSET を使用して、オペレーティング・システムのデフォルトの文字セット以外の値に設定されている場合、ucharset.h ファイルの ULIB_CS_DEFAULT で定義されているように、ユーザー・イグジットとプロセス間で交換される文字データは、セッションの文字セットで解釈されます。

データベースが大 / 小文字を区別する場合、オブジェクト名は、ホストしているデータベースで定義されているのと同じ大 / 小文字の区別で指定する必要があります。それ以外の場合、大 / 小文字の区別はありません。

構文

```
#include "usrdecs.h"
short result_code;
col_metadata_def column_meta_rec;
ERCALLBACK (GET_COL_METADATA_FROM_NAME, &column_meta_rec, &result_code);
```

バッファ

```
typedef struct
{
    short column_index;
    char *column_name;
    long max_name_length;
    short native_data_type;
    short gg_data_type;
    short gg_sub_data_type;
    short is_nullable;
    short is_part_of_key;
    short key_column_index;
    short length;
    short precision;
    short scale;
    short source_or_target;
} col_metadata_def;
```

入力　　次のようになります。

入力	説明
column_name	返される列値の列名。
source_or_target	ソースまたはターゲットのどちらのレコードが圧縮されているかを示す次のいずれか。 EXIT_FN_SOURCE_VAL EXIT_FN_TARGET_VAL

出力　　次のようになります。

出力	説明
column_index	返される列値の列索引。
source_or_target	ソースまたはターゲットのどちらのレコードが圧縮されているかを示す次のいずれか。 EXIT_FN_SOURCE_VAL EXIT_FN_TARGET_VAL
native_data_type	列の (データベースに) ネイティブなデータ型。
gg_data_type	列の Oracle GoldenGate データ型。
gg_sub_data_type	列の Oracle GoldenGate サブデータ型。
is_nullable	列が NULL 値を許可するかどうかを示すフラグ (TRUE または FALSE)。
is_part_of_key	列が Oracle GoldenGate によって使用されているキーの一部かどうかを示すフラグ (TRUE または FALSE)。
key_column_index	索引での列の順序を示します。たとえば、次の表は、表での定義の順序と索引での定義の順序が異なる 2 つのキー列を持ちます。 CREATE TABLE tcustmer (cust_code VARCHAR2(4), name VARCHAR2(30), city VARCHAR2(20), state CHAR(2), PRIMARY KEY (city, cust_code) USING INDEX);

出力	説明
	返される値は次のようになります。 **列名** / **返されるキー索引値** cust_code / 1 name / -1 city / 0 state / -1 ◆ 列がキーの一部の場合は、索引での順序が整数で返されます。 ◆ 列がキーの一部ではない場合は、値 -1 が返されます。
`length`	列の長さを返します。
`precision`	数値データ型の場合、列の精度を返します。
`scale`	数値データ型の場合、位取りを返します。

返される値

```
EXIT_FN_RET_INVALID_PARAM
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_EXCEEDED_MAX_LENGTH
EXIT_FN_RET_INVALID_COLUMN
EXIT_FN_RET_OK
```

GET_COLUMN_INDEX_FROM_NAME

適用対象 Extract および Replicat

GET_COLUMN_INDEX_FROM_NAME ファンクションでは、特定の列名に関連付けられている列索引を確認します。ユーザー・イグジットの文字セッションが、SET_SESSION_CHARSET を使用して、オペレーティング・システムのデフォルトの文字セット以外の値に設定されている場合、ucharset.h ファイルの ULIB_CS_DEFAULT で定義されているように、ユーザー・イグジットとプロセス間で交換される文字データは、セッションの文字セットで解釈されます。

データベースが大 / 小文字を区別する場合、オブジェクト名は、ホストしているデータベースで定義されているのと同じ大 / 小文字の区別で指定する必要があります。それ以外の場合、大 / 小文字の区別はありません。

構文

```
#include "usrdecs.h"
short result_code;
env_value_def env_value;
ERCALLBACK (GET_COLUMN_INDEX_FROM_NAME, &env_value, &result_code);
```

バッファ

```
typedef struct
{
char *buffer;
long max_length;
long actual_length;
short value_truncated;
short index;
short source_or_target;
} env_value_def;
```

入力

次のようになります。

入力	説明
buffer	列名へのポインタ。
actual_length	バッファ内の列名の長さ。
source_or_target	列名情報を参照するためにソース表とターゲット表のどちらを使用するかを示す次の一方。 EXIT_FN_SOURCE_VAL EXIT_FN_TARGET_VAL

出力

次のようになります。

出力	説明
index	指定した列名に対して返される列索引。

返される値

```
EXIT_FN_RET_INVALID_COLUMN
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_INVALID_PARAM
EXIT_FN_RET_OK
```

GET_COLUMN_NAME_FROM_INDEX

適用対象

Extract および Replicat

GET_COLUMN_NAME_FROM_INDEX ファンクションでは、特定の列索引に関連付けられている列名を確認します。ユーザー・イグジットの文字セッションが、SET_SESSION_CHARSET を使用して、オペレーティング・システムのデフォルトの文字セット以外の値に設定されている場合、ucharset.h ファイルの ULIB_CS_DEFAULT で定義されているように、ユーザー・イグジットとプロセス間で交換される文字データは、セッションの文字セットで解釈されます。

データベース・オブジェクト名は、大 / 小文字の区別を含め、ホストしているデータベースで定義されているとおりに、正確に返されます。

構文

```
#include "usrdecs.h"
short result_code;
env_value_def env_value;
ERCALLBACK (GET_COLUMN_NAME_FROM_INDEX, &env_value, &result_code);
```

バッファ

```
typedef struct
{
char *buffer;
long max_length;
long actual_length;
short value_truncated;
short index;
short source_or_target;
} env_value_def;
```

入力　次のようになります。

入力	説明
buffer	返される列名を受け付けるバッファへのポインタ。列名は NULL で終了します。
max_length	結果の列名を受け付けるために割り当てた buffer の最大長。NULL 終了文字列として返されます。
index	返される列名の列索引。
source_or_target	列名情報を参照するためにソース表とターゲット表のどちらを使用するかを示す次の一方。 EXIT_FN_SOURCE_VAL EXIT_FN_TARGET_VAL

出力　次のようになります。

出力	説明
buffer	NULL で終了している列名。
actual length	返される列名の文字列長。実際の長さに NULL 終了文字は含まれません。
value_truncated	値が切り捨てられたかどうかを示すフラグ (0 または 1)。切捨ては、列名と NULL 終了文字を足した長さが、最大バッファ長を超えるときに行われます。

返される値

```
EXIT_FN_RET_INVALID_COLUMN
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_INVALID_PARAM
EXIT_FN_RET_OK
```

GET_COLUMN_VALUE_FROM_INDEX

適用対象　Extract および Replicat

GET_COLUMN_VALUE_FROM_INDEX ファンクションでは、指定する列索引を使用して、データ・レコードから列値を取得します。列値は、ユーザー・イグジット内のほとんどのロジックの基本です。データ・レコード内の各列の値は、複雑なロジックの基準にできます。返される値の文字形式を指定できます。

ユーザー・イグジットの文字セッションが、SET_SESSION_CHARSET を使用して、オペレーティング・システムのデフォルトの文字セット以外の値に設定されている場合、ucharset.h ファイルの ULIB_CS_DEFAULT で定義されているように、ユーザー・イグジットとプロセス間で交換される文字データは、セッションの文字セットで解釈されます。

次に該当する場合、列値はセッションの文字セットにのみ設定されます。

- 列値が SQL 文字型 (CHAR/VARCHAR2/CLOB、NCHAR/NVARCHAR2/NCLOB)、SQL 日付 / タイムスタンプ / 間隔 / 数値型である。
- column_value_mode インジケータが EXIT_FN_CNVTED_SESS_CHAR_FORMAT に設定されている。

構文

```
#include "usrdecs.h"
short result_code;
column_def column;
ERCALLBACK (GET_COLUMN_VALUE_FROM_INDEX, &column, &result_code);
```

バッファ

```
typedef struct
{
char *column_value;
unsigned short max_value_length;
unsigned short actual_value_length;
short null_value;
short remove_column;
short value_truncated;
short column_index;
char *column_name;
/* Version 3 CALLBACK_STRUCT_VERSION */
short column_value_mode;
short source_or_target;
/* Version 2 CALLBACK_STRUCT_VERSION   */
char requesting_before_after_ind;
char more_lob_data;
/* Version 3 CALLBACK_STRUCT_VERSION */
ULibCharSet column_charset;
} column_def;
```

入力

次のようになります。

入力	説明
column_value	返された列値を受け付けるバッファへのポインタ。
max_value_length	返される列値の最大長。通常、最大長は列値バッファの長さです。column_value_mode で ASCII フォーマットが指定される場合、列値は NULL で終了し、最大長は列値の最大長と同じになるはずです。

入力	説明
column_index	返される列値の列索引。
column_value_mode	列値のフォーマットを示します。 EXIT_FN_CHAR_FORMAT EXIT_FN_RAW_FORMAT EXIT_FN_CNVTED_SESS_CHAR_FORMAT **EXIT_FN_CHAR_FORMAT** ASCII フォーマット : 値は、NULL で終了している ASCII (または EBCDIC) 文字列です (既知の例外としてサブデータ型 UTF16_BE があり、これは UTF8 に変換されます)。 **注意** : 列値は、ASCII 文字列として解釈され、NULL で終了する必要があるため、ユーザー・イグジットに示される際に切り捨てられる可能性があります。最初の 0 の値は、文字列の終了文字になります。 ◆ 日付は CCYY-MM-DD HH:MI:SS.FFFFFF のフォーマットで、時間の端数はデータベースに依存します。 ◆ 数値は文字列フォーマットです。たとえば、123.45 は "123.45" として表されます。 ◆ 出力不可能な文字またはバイナリ値は、16 進表記法に変換されます。 ◆ 浮動小数点型は、有効桁数が最初の 14 桁までの NULL 終了文字列として出力されます。 **EXIT_FN_RAW_FORMAT** 内部 Oracle GoldenGate 正規フォーマット : このフォーマットには、適切な場合に、2 バイトの NULL インジケータおよび 2 バイトの変数データ長が含まれます。文字データ型の場合、Oracle GoldenGate ではこのフォーマットへの文字セットの変換は実行されません。 **EXIT_FN_CNVTED_SESS_CHAR_FORMAT** ユーザー・イグジットの文字セット : これが適用されるのは、列のデータ型が次の場合のみです。 ◆ シングルバイトまたはマルチバイトの文字ベースの型 ◆ 文字列で表現する数値型 このフォーマットは NULL で終了しません。
source_or_target	列値の取得にソースまたはターゲット・データ・レコードのどちらを使用するかを示す次の一方。 EXIT_FN_SOURCE_VAL EXIT_FN_TARGET_VAL

入力	説明
requesting_before_after_ind	アフター・イメージ・レコードの処理中に、更新または主キー更新のビフォア・イメージ列値を必要とする場合に設定します。 主キー更新または通常の (非キー) 更新レコードの " アフター・イメージ " を処理中に、列の " ビフォア " 値を取得するには、requesting_before_after_ind フラグを BEFORE_IMAGE_VAL に設定します。 ◆ 主キー更新のキー列のビフォア・イメージにアクセスする場合は、他の設定は不要です。 ◆ 主キー更新の非キー列または通常の更新の任意の列にアクセスするには、ビフォア・イメージが使用可能である必要があります。 requesting_before_after_ind の入力が明示的に指定されていない場合、デフォルト設定は AFTER_IMAGE_VAL(列のアフター・イメージの取得) です。 ビフォア・イメージを使用可能にするには、GETUPDATEBEFORES パラメータを使用するか、CUSEREXIT パラメータ文内の INCLUDEUPDATEBEFORES オプションを使用します。 次の点に注意してください。 ◆ GETUPDATEBEFORES を使用すると、Extract プロセスはトレイルにビフォア・イメージを書き込み、ビフォア・イメージを使用してユーザー・イグジットへの EXIT_CALL_PROCESS_RECORD コールも行います。 ◆ INCLUDEUPDATEBEFORES を使用すると、ユーザー・イグジットへの EXIT_CALL_PROCESS_RECORD コールは行われず、Extract の場合はトレイルへのビフォア・イメージの書き込みも行われません。

出力　次のようになります。

出力	説明
column_value	返される列値へのポインタ。column_value_mode が EXIT_FN_CHAR_FORMAT として指定されている場合、列値は NULL で終了する ASCII 文字列として返され、それ以外の場合は Oracle GoldenGate 内部正規フォーマットで返されます。ASCII フォーマット場合、日付は次のフォーマットで返されます。 YYYY-MM-DD HH:MI:SS.FFFFFF 秒の端数が含まれるかどうかは、データベースに依存します。
actual_value_length	返される列名の文字列長 (バイト)。column_value_mode が EXIT_FN_CHAR_FORMAT として指定されている場合、実際の長さに NULL 終了文字は含まれません。

出力	説明
`null_value`	列値が NULL かどうかを示すフラグ (0 または 1)。null_value フラグが 1 の場合、列値バッファは NULL バイトで埋められます。
`value_truncated`	値が切り捨てられたかどうかを示すフラグ (0 または 1)。切捨ては、列値の長さが最大バッファ長を超えるときに行われます。column_value_mode が EXIT_FN_CHAR_FORMAT として指定されている場合は、列長に NULL 終了文字が含まれます。
`char more_lob_data;`	ベース・レコードに保持可能な初期の 4K の制限よりも大きい LOB データが存在するかどうかを示すフラグ。LOB は、4K の上限を越えている場合、LOB フラグメントに保持されます。 返される値を保持するために適切な量のメモリーを割り当てる必要があります。Oracle GoldenGate は、通常 LOB 列のデータに 8K までアクセスし、ユーザー・イグジットによって割り当てられた量までバッファに格納します。割り当てられた量よりも LOB が大きい場合は、すべてのデータがユーザー・イグジットに送信されるまで、後続のコールバックを使用してすべての列データを取得する必要があります。 データの終端を確認するには、more_lob_data を検証します。ユーザー・イグジットは、新しい列にアクセスする前に、このフラグを CHAR_NO_VAL または CHAR_YES_VAL に設定します。最初のコールバック後にこのフラグがまだ初期化状態で、CHAR_YES_VAL または CAR_NO_VAL にも設定されていない場合は、次のいずれかに該当します。 ◆ LOB の処理に十分なメモリーが割り当てられていた。 ◆ LOB でなかった。 ◆ ベース・トレイル・レコード・サイズの 4K の制限を越えていなかった。 列が LOB かどうかを確認するために、ソース表メタデータを取得することをお薦めします。

返される値

```
EXIT_FN_RET_BAD_COLUMN_DATA
EXIT_FN_RET_COLUMN_NOT_FOUND
EXIT_FN_RET_INVALID_COLUMN
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_INVALID_PARAM
EXIT_FN_RET_OK
```

GET_COLUMN_VALUE_FROM_NAME

適用対象 Extract および Replicat

GET_COLUMN_VALUE_FROM_NAME ファンクションでは、指定する列名を使用して、データ・レコードから列値を取得します。列値は、ユーザー・イグジット内のほとんどのロジックの基本です。データ・レコード内の各列の値は、複雑なロジックの基準にできます。

ユーザー・イグジットの文字セッションが、SET_SESSION_CHARSET を使用して、オペレーティング・システムのデフォルトの文字セット以外の値に設定されている場合、ucharset.h ファイルの ULIB_CS_DEFAULT で定義されているように、ユーザー・イグジットとプロセス間で交換される文字データは、セッションの文字セットで解釈されます。

次に該当する場合、列値はセッションの文字セットにのみ設定されます。

- 列値がSQL文字型(CHAR/VARCHAR2/CLOB、NCHAR/NVARCHAR2/NCLOB)、SQL日付/タイムスタンプ/間隔/数値型である。
- column_value_mode インジケータが EXIT_FN_CNVTED_SESS_CHAR_FORMAT に設定されている。

データベースが大 / 小文字を区別する場合、オブジェクト名は、ホストしているデータベースで定義されているのと同じ大 / 小文字の区別で指定する必要があります。それ以外の場合、大 / 小文字の区別はありません。

構文

```
#include "usrdecs.h"
short result_code;
column_def column;
ERCALLBACK (GET_COLUMN_VALUE_FROM_NAME, &column, &result_code);
```

バッファ

```
typedef struct
{
char *column_value;
unsigned short max_value_length;
unsigned short actual_value_length;
short null_value;
short remove_column;
short value_truncated;
short column_index;
char *column_name;
/* Version 3 CALLBACK_STRUCT_VERSION */
short column_value_mode;
short source_or_target;
/* Version 2 CALLBACK_STRUCT_VERSION   */
char requesting_before_after_ind;
char more_lob_data;
/* Version 3 CALLBACK_STRUCT_VERSION */
ULibCharSet column_charset;
} column_def;
```

入力

次のようになります。

入力	説明
column_value	返された列値を受け付けるバッファへのポインタ。
max_value_length	返される列値の最大長。通常、最大長は列値バッファの長さです。ASCII フォーマットが指定された場合 (column_value_mode を参照)、列値は NULL で終了し、最大長は列値の最大長と同じになります。
column_name	返される列値の列名。
column_value_mode	列値の文字セットを示します。 EXIT_FN_CHAR_FORMAT EXIT_FN_RAW_FORMAT EXIT_FN_CNVTED_SESS_CHAR_FORMAT

入力	説明
	EXIT_FN_CHAR_FORMAT ASCII フォーマット : 値は、NULL で終了している ASCII (または EBCDIC) 文字列です (既知の例外としてサブデータ型 UTF16_BE があり、これは UTF8 に変換されます)。 **注意** : 列値は、ASCII 文字列として解釈され、NULL で終了する必要があるため、ユーザー・イグジットに示される際に切り捨てられる可能性があります。最初の 0 の値は、文字列の終了文字になります。 ◆ 日付は CCYY-MM-DD HH:MI:SS.FFFFFF のフォーマットで、時間の端数はデータベースに依存します。 ◆ 数値は文字列フォーマットです。たとえば、123.45 は "123.45" として表されます。 ◆ 出力不可能な文字またはバイナリ値は、16 進表記法に変換されます。 ◆ 浮動小数点型は、有効桁数が最初の 14 桁までの NULL 終了文字列として出力されます。 **EXIT_FN_RAW_FORMAT** 内部 Oracle GoldenGate 正規フォーマット : このフォーマットには、適切な場合に、2 バイトの NULL インジケータおよび 2 バイトの変数データ長が含まれます。文字データ型の場合、Oracle GoldenGate ではこのフォーマットへの文字セットの変換は実行されません。 **EXIT_FN_CNVTED_SESS_CHAR_FORMAT** ユーザー・イグジットの文字セット : これが適用されるのは、列のデータ型が次の場合のみです。 ◆ シングルバイトまたはマルチバイトの文字ベースの型 ◆ 文字列で表現する数値型 このフォーマットは NULL で終了しません。

`source_or_target`	列値の取得にソースまたはターゲット・データ・レコードのどちらを使用するかを示す次の一方。
	`EXIT_FN_SOURCE_VAL`
	`EXIT_FN_TARGET_VAL`

入力	説明
requesting_before_after_ind	アフター・イメージ・レコードの処理中で、更新または主キー更新のビフォア列を必要とする場合に設定します。 主キー更新または通常の (非キー) 更新レコードの " アフター・イメージ " を処理中に、列の " ビフォア " 値を取得するには、requesting_before_after_ind フラグを BEFORE_IMAGE_VAL に設定します。 ◆ 主キー更新のキー列のビフォア・イメージにアクセスする場合は、他の設定は不要です。 ◆ 主キー更新の非キー列または通常の更新の任意の列にアクセスするには、ビフォア・イメージが使用可能である必要があります。 requesting_before_after_ind の入力が明示的に指定されていない場合、デフォルト設定は AFTER_IMAGE_VAL(列のアフター・イメージの取得) です。 ビフォア・イメージを使用可能にするには、GETUPDATEBEFORES パラメータを使用するか、CUSEREXIT パラメータ文内の INCLUDEUPDATEBEFORES オプションを使用します。 次の点に注意してください。 ◆ GETUPDATEBEFORES を使用すると、Extract プロセスはトレイルにビフォア・イメージを書き込み、ビフォア・イメージを使用してユーザー・イグジットへの EXIT_CALL_PROCESS_RECORD コールも行います。 ◆ INCLUDEUPDATEBEFORES を使用すると、ユーザー・イグジットへの EXIT_CALL_PROCESS_RECORD コールは行われず、Extract の場合はトレイルへのビフォア・イメージの書き込みも行われません。

出力　次のようになります。

出力	説明
column_value	返される列値へのポインタ。column_value_mode が EXIT_FN_CHAR_FORMAT として指定されている場合、列値は NULL で終了する ASCII 文字列として返され、それ以外の場合は Oracle GoldenGate 内部正規フォーマットで返されます。ASCII フォーマット場合、日付は次のフォーマットで返されます。 CCYY-MM-DD HH:MI:SS.FFFFFF 秒の端数が含まれるかどうかは、データベースに依存します。
actual length	返される列名の文字列長。column_value_mode が EXIT_FN_CHAR_FORMAT として指定されている場合、実際の長さに NULL 終了文字は含まれません。

出力	説明
`null_value`	列値が NULL かどうかを示すフラグ (0 または 1)。null_value フラグが 1 の場合、列値バッファは NULL バイトで埋められます。
`value_truncated`	値が切り捨てられたかどうかを示すフラグ (0 または 1)。切捨ては、列値の長さが最大バッファ長を超えるときに行われます。column_value_mode が EXIT_FN_CHAR_FORMAT として指定されている場合は、列長に NULL 終了文字が含まれます。
`char more_lob_data;`	ベース・レコードに保持可能な初期の 4K の制限よりも大きい LOB データが存在するかどうかを示すフラグ。LOB は、4K の上限を越えている場合、LOB フラグメントに保持されます。 返される値を保持するために適切な量のメモリーを割り当てる必要があります。Oracle GoldenGate は、通常 LOB 列のデータに 8K までアクセスし、ユーザー・イグジットによって割り当てられた量までバッファに格納します。割り当てられた量よりも LOB が大きい場合は、すべてのデータがユーザー・イグジットに送信されるまで、後続のコールバックを使用してすべての列データを取得する必要があります。 データの終端を確認するには、more_lob_data を検証します。ユーザー・イグジットは、新しい列にアクセスする前に、このフラグを CAR_NO_VAL または CHAR_YES_VAL に設定します。最初のコールバック後にこのフラグがまだ初期化状態で、CHAR_YES_VAL または CAR_NO_VAL にも設定されていない場合は、次のいずれかに該当します。 ◆ LOB の処理に十分なメモリーが割り当てられていた。 ◆ LOB でなかった。 ◆ ベース・トレイル・レコード・サイズの 4K の制限を越えていなかった。 列が LOB かどうかを確認するために、ソース表メタデータを取得することをお薦めします。

返される値

```
EXIT_FN_RET_BAD_COLUMN_DATA
EXIT_FN_RET_COLUMN_NOT_FOUND
EXIT_FN_RET_INVALID_COLUMN
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_INVALID_PARAM
EXIT_FN_RET_OK
```

例

```
memset (&col_meta, 0, sizeof(col_meta));
if (record.mapped)
col_meta.source_or_target = EXIT_FN_TARGET_VAL;
else
col_meta.source_or_target = EXIT_FN_SOURCE_VAL;
col_meta.source_or_target = EXIT_FN_SOURCE_VAL;
col_meta.column_name = (char *)malloc(100);
col_meta.max_name_length = 100;
col_meta.column_index = 1;

call_callback (GET_COL_METADATA_FROM_NAME, &col_meta, &result_code);
```

GET_DATABASE_METADATA

適用対象 Extract および Replicat

GET_DATABASE_METADATA ファンクションでは、レコードに関連付けられているデータベースのメタデータを返します。

構文

バッファ

```
typedef struct
{
char*    dbName;
long      dbName_max_length;
long      dbName_actual_length;
unsigned char    dbNameMetadata[MAXDBOBJTYPE];
char*   locale;
long     locale_max_length;
long     locale_actual_length;
} database_def;
typedef struct
{
    database_def  source_db_def;
    database_def  target_db_def;
} database_defs;
```

入力 次のようになります。

入力	説明
dbname	データベース名を受け付けるバッファへのポインタ。
dbname_max_length	名前を保持するバッファの最大長。
dbname_actual_length	データベース名の実際の長さ。
dbNameMetadata	大 / 小文字を区別する名前メタデータで、Extract によってデータ・ポンプからトレイルに書き込まれる値と同じになります。オブジェクト名タイプが指定された場合、データベース・オブジェクト名のメタデータをチェックするユーザー・イグジットが使用できるマクロのリストは、『Oracle GoldenGate *管理ガイド*』を参照してください。
locale	データベースのロケールを指定する NULL 終了文字列。これは、次を結合したものとして返されます。 ◆ ISO-639 の 2 文字の言語コード ◆ ISO-3166 の 2 文字の国コード ◆ セパレータに '_' U+005F を使用したバリアント・コード 例 : "en_US"、"ja_Japen"
locale_max_length	ロケールを受け付けるバッファの最大長。

入力	説明
locale_actual_length	ロケールの実際の長さ。
database_def source_db_def	ソース・データベースのメタデータを返すようにプロセスに指示します。
database_def target_db_def	ターゲット・データベースのメタデータを返すようにプロセスに指示します。

出力

返される値

GET_DDL_RECORD_PROPERTIES

適用対象 Extract および Replicat(DDL レプリケーションがサポートされているデータベース)

GET_DDL_RECORD_PROPERTIES ファンクションでは、DDL が実行されたオブジェクトに関する情報や DDL 文自体のテキストも含む DDL 操作の情報を返します Extract プロセスは、ソース表レイアウトのみを取得できます。Replicat プロセスは、ソースまたはターゲット・レイアウトを取得できます。

ユーザー・イグジットの文字セッションが、SET_SESSION_CHARSET を使用して、オペレーティング・システムのデフォルトの文字セット以外の値に設定されている場合、ucharset.h ファイルの ULIB_CS_DEFAULT で定義されているように、ユーザー・イグジットとプロセス間で交換される文字データは、セッションの文字セットで解釈されます。これには、DDL タイプ、オブジェクト・タイプ、オブジェクト名、所有者名および DDL テキスト自体が含まれます。

構文

```
#include "usrdecs.h"
short result_code;
ddl_record_def ddl_rec;
ERCALLBACK (GET_DDL_RECORD_PROPERTIES, &ddl_rec, &result_code);
```

バッファ

```
typedef struct
{
char *ddl_type;
long ddl_type_max_length; /* Maximum Description length PASSED IN BY USER */
long ddl_type_length; /* Actual length */

char *object_type;
long object_type_max_length; /* Maximum Description length PASSED IN BY USER */
long object_type_length; /* Actual length */

char *object_name;
long object_max_length; /* Maximum Description length PASSED IN BY USER */
long object_length; /* Actual length */

char *owner_name;
long owner_max_length; /* Maximum Description length PASSED IN BY USER */
long owner_length; /* Actual length */

char *ddl_text;
long ddl_text_max_length; /* Maximum Description length PASSED IN BY USER */
long ddl_text_length; /* Actual length */

short ddl_text_truncated; /* Was value truncated? */
short source_or_target; /* Source or target value? */
} ddl_record_def;
```

入力

次のようになります。

入力	説明
ddl_type_length object_type_length object_length owner_length ddl_text_length	返される列値を受け付ける各アイテム用の 1 つのバッファへのポインタ。アイテムは次のとおりです。 ◆ ddl_type_length には、DDL 操作タイプの長さ (CREATE または ALTER など) を含めます。 ◆ object_type_length には、DDL 操作によって影響を受けるオブジェクトのタイプの長さ (TABLE または INDEX など) を含めます。 ◆ object_length には、オブジェクト名の長さが含めます。 ◆ owner_length には、オブジェクトの所有者 (スキーマまたはデータベース) の長さを含めます。 ◆ ddl_text_length には、実際の DDL 文テキストの長さを含めます。
ddl_type_max_length	*ddl_type によって返される DDL 操作タイプの最大長。DDL タイプは、データベースに対して有効な任意の DDL コマンド (ALTER など) です。
object_type_max_length	*object_type によって返されるオブジェクト・タイプの最大長。オブジェクト・タイプは、データベースに有効な任意のオブジェクト (TABLE、INDEX、TRIGGER など) です。

入力	説明
object_max_length	*object_name によって返されるオブジェクト名の最大長。
owner_max_length	*owner_name によって返される所有者名の最大長。
ddl_text_max_length	*ddl_text によって返される DDL 文のテキストの最大長。
source_or_target	ソースまたはターゲット・データ・レコードのどちらの操作タイプを返すかを示す次の一方。 EXIT_FN_SOURCE_VAL EXIT_FN_TARGET_VAL

出力　次のようになります。

出力	説明
ddl_type_length object_type_length object_length owner_length ddl_text_length	これらのフィールドはすべて、リクエストされた値の実際の長さを返します。(説明は入力のセクションを参照してください。)
ddl_text_truncated	DDL テキストが切り捨てられたかどうかを示すフラグ (0 または 1)。切捨ては、DDL テキストと NULL 終了文字を足した長さが、最大バッファ長を超えるときに行われます。

返される値
EXIT_FN_RET_OK
EXIT_FN_RET_NOT_SUPPORTED
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_INCOMPLETE_DDL_REC

GET_ENV_VALUE

適用対象　Extract および Replicat

GET_ENV_VALUE ファンクションでは、Oracle GoldenGate 環境に関する情報を返します。指定される情報は、@GETENV 列変換ファンクションと同じで、同一の入力値を使用して指定されます。有効な情報タイプ、環境変数、および戻り値の詳細は、441 ページの @GETENV の項を参照してください。

ユーザー・イグジットの文字セッションが、SET_SESSION_CHARSET を使用して、オペレーティング・システムのデフォルトの文字セット以外の値に設定されている場合、ucharset.h ファイルの ULIB_CS_DEFAULT で定義されているように、ユーザー・イグジットとプロセス間で交換される文字データは、セッションの文字セットで解釈されます。

構文

```
#include "usrdecs.h"
short result_code;
getenv_value_def env_ptr;
ERCALLBACK (GET_ENV_VALUE, &env_ptr, &result_code);
```

バッファ

```
typedef struct
{
char *information_type;
char *env_value_name;
char *return_value;
long max_return_length;
long actual_length;
short value_truncated;
} getenv_value_def;
```

入力　次のようになります。

入力	説明
`information_type`	返される情報タイプ ("GGENVIRONMENT" や "GGHEADER" など)。情報タイプは、二重引用符で囲って指定する必要があります。情報タイプのリストと詳細な説明は、441 ページを参照してください。
`env_value_name`	情報タイプから取得する環境値。環境値は、二重引用符で囲って指定する必要があります。有効な値については、441 ページから開始されている、使用する情報タイプの**環境値**のリストを参照してください。たとえば、"GGENVIRONMENT" 情報タイプを使用している場合は、"GROUPNAME" が有効な環境値の 1 つです。
`max_return_length`	このデータのバッファの最大長。

出力　次のようになります。

出力	説明
`return_value`	指定する環境変数の説明のセクションで、この環境値に対してリストされている有効な**戻り値**。
`actual_length`	このバッファのデータの実際の長さ。
`value_truncated`	値が切り捨てられたかどうかを示すフラグ (0 または 1)。切捨ては、値と NULL 終了文字を足した長さが、最大バッファ長を超えるときに行われます。

返される値

```
EXIT_FN_RET_OK
EXIT_FN_RET_ENV_NOT_FOUND
EXIT_FN_RET_INVALID_PARAM
```

GET_ERROR_INFO

適用対象 Extract および Replicat

GET_ERROR_INFO ファンクションでは、破棄レコードに関連付けられているエラー情報を取得します。ユーザー・イグジットは、この情報をカスタム・エラー処理ロジックで使用できます。たとえば、ユーザー・イグジットは詳細なエラー情報を含む電子メール・メッセージを送信できます。

ユーザー・イグジットの文字セッションが、SET_SESSION_CHARSET を使用して、オペレーティング・システムのデフォルトの文字セット以外の値に設定されている場合、ucharset.h ファイルの ULIB_CS_DEFAULT で定義されているように、ユーザー・イグジットとプロセス間で交換されるメッセージ・データは、セッションの文字セットで解釈されます。

構文

```
#include "usrdecs.h"
short result_code;
error_info_def error_info;
ERCALLBACK (GET_ERROR_INFO, &error_info, &result_code);
```

バッファ

```
typedef struct
{
long error_num;
char *error_msg;
long max_length;
long actual_length;
short msg_truncated;
} error_info_def;
```

入力 次のようになります。

入力	説明
error_msg	返されるエラー・メッセージを受け付けるバッファへのポインタ。
max_length	結果のエラー・メッセージを受け付けるために割り当てた error_msg バッファの最大長。これは NULL 終了文字列として返されます。

出力 次のようになります。

出力	説明
error_num	破棄ファイルに関連付けられている SQL またはシステム・エラー番号。
error_msg	破棄レコードに関連付けられている NULL で終了するエラー・メッセージ文字列へのポインタ。
actual_length	NULL 終了文字を含まないエラー・メッセージ長。
msg_truncated	エラー・メッセージが切り捨てられたかどうかを示すフラグ (0 または 1)。切捨ては、エラー・メッセージと NULL 終了文字を足した長さが、最大バッファ長を超えるときに行われます。

返される値

```
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_OK
```

GET_GMT_TIMESTAMP

適用対象 Extract および Replicat

GET_GMT_TIMESTAMP ファンクションでは、操作コミット・タイムスタンプを GMT フォーマットで取得します。このファンクションは、usrdecs.h バージョン 2 以降を使用してコンパイルする必要があります。

構文

```
#include "usrdecs.h"
short result_code;
record_def record;
ERCALLBACK (GET_GMT_TIMESTAMP, &record, &result_code);
```

バッファ

```
typedef struct
{
char *table_name;
char *buffer;
long length;
char before_after_ind;
short io_type;
short record_type;
short transaction_ind;
int64_t timestamp;
exit_ts_str io_datetime;
short mapped;
short source_or_target;
/* Version 2 CALLBACK_STRUCT_VERSION   */
char requesting_before_after_ind;
} record_def;
```

入力 なし

出力 次のようになります。

出力	説明
timestamp	返される GMT フォーマットの 64 ビット I/O タイムスタンプ。
io_datetime	ローカル I/O 日付および時刻を含む、次の NULL 終了文字列。 YYYY-MM-DD HH:MI:SS.FFFFFF 日時文字列のフォーマットは、セッションの文字セットにあります。

返される値

```
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_OK
```

GET_MARKER_INFO

適用対象 Extract(データ・ポンプのみ) および Replicat

GET_MARKER_INFO ファンクションでは、Replicat がデータを適用しているときに NonStop ソース・シ

ステムから送信されたマーカー情報を取得します。マーカーは、ユーザー・イグジット内でカスタム処理をトリガーするために使用します。

ユーザー・イグジットの文字セッションが、SET_SESSION_CHARSET を使用して、オペレーティング・システムのデフォルトの文字セット以外の値に設定されている場合、ucharset.h ファイルの ULIB_CS_DEFAULT で定義されているように、返されるすべてのマーカー・データはセッションの文字セットで解釈されます。

構文

```
#include "usrdecs.h"
short result_code;
marker_info_def marker_info;
ERCALLBACK (GET_MARKER_INFO, &marker_info, &result_code);
```

バッファ

```
typedef struct
{
char *processed;
char *added;
char *text;
char *group;
char *program;
char *node;
} marker_info_def;
```

入力

次のようになります。

出力	説明
processed	processed 戻り値を受け付けるバッファへのポインタ。
added	added 戻り値を受け付けるバッファへのポインタ。
text	text 戻り値を受け付けるバッファへのポインタ。
group	group 戻り値を受け付けるバッファへのポインタ。
program	program 戻り値を受け付けるバッファへのポインタ。
node	node 戻り値を受け付けるバッファへのポインタ。

出力

次のようになります。

出力	説明
processed	マーカーが処理されたローカルの日付と時刻を示す YYYY-MM-DD HH:MI:SS フォーマットの NULL 終了文字列。
added	マーカーが追加されたローカルの日付と時刻を示す YYYY-MM-DD HH:MI:SS フォーマットの NULL 終了文字列。
text	マーカーに関連付けられたテキストを含む NULL 終了文字列。
group	マーカーを処理した Replicat グループを示す NULL 終了文字列。

出力	説明
program	マーカーを処理したプログラムを示す NULL 終了文字列。
node	マーカーが作成された Himalaya ノードを表す NULL 終了文字列。

返される値

```
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_OK
```

GET_OPERATION_TYPE

適用対象

Extract および Replicat

GET_OPERATION_TYPE ファンクションでは、レコードに関連付けられている操作のタイプを確認します。ユーザー・イグジットで操作のタイプを把握できることは有益です。たとえばユーザー・イグジットは、削除操作を検出するたびに複雑な検証を実行できます。また、ユーザー・イグジットが完全なデータ・レコードを操作する場合は、圧縮レコードがいつ処理されるかを把握することも重要です。

かわりの方法として、GET_RECORD_BUFFER ファンクションでも操作タイプを確認できます (513 ページを参照してください)。

構文

```
#include "usrdecs.h"
short result_code;
record_def record;
ERCALLBACK (GET_OPERATION_TYPE, &record, &result_code);
```

バッファ

```
typedef struct
{
char *table_name;
char *buffer;
long length;
char before_after_ind;
short io_type;
short record_type;
short transaction_ind;
int64_t timestamp;
exit_ts_str io_datetime;
short mapped;
short source_or_target;
/* Version 2 CALLBACK_STRUCT_VERSION   */
char requesting_before_after_ind;
} record_def;
```

入力

次のようになります。

入力	説明
source_or_target	ソースまたはターゲット・データ・レコードのどちらの操作タイプを返すかを示す次の一方。 EXIT_FN_SOURCE_VAL EXIT_FN_TARGET_VAL

出力　次のようになります。

出力	説明
io_type	次のいずれかが返されます。 **DDL タイプ** SQL_DDL_VAL **DML タイプ** DELETE_VAL INSERT_VAL UPDATE_VAL **圧縮 Enscribe 更新** UPDATE_COMP_ENSCRIBE_VAL **圧縮 SQL 更新** UPDATE_COMP_SQL_VAL UPDATE_COMP_PK_SQL_VAL **その他** TRUNCATE_TABLE_VAL

返される値
```
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_INVALID_PARAM
EXIT_FN_RET_OK
```

GET_POSITION

適用対象　Extract(データ・ポンプのみ) および Replicat

GET_POSITION ファンクションでは、Oracle GoldenGate トレイル内の Extract データ・ポンプまたは Replicat の読取り位置を取得します。

構文
```
#include "usrdecs.h"
short result_code;
ERCALLBACK (GET_POSITION &position_def, &result_code);
```

バッファ
```
typedef struct
{
char *position;
long position_len;
short position_type;
short ascii_or_internal;
} position_def;
```

入力　次のようになります。

入力	説明
position_len	位置の長さへの割当ての長さ。
position_type	次のいずれかになります。 ◆ STARTUP_CHECKPOINT トレイルの開始位置。 ◆ CURRENT_CHECKPOINT トレイルの最後の読取り位置。
column_value_mode	列値が渡されたフォーマットのインジケータ。現在はデフォルトの Oracle GoldenGate 正規フォーマットのみサポートされており、次のように表されます。 EXIT_FN_RAW_FORMAT

出力　次のようになります。

出力	説明
*position	位置の値を表すバッファへのポインタ。このバッファは、position_def で、char フィールドの 8 バイトの seqnorba として、2 つのバイナリ値 (符号なし int32t および int32t) として宣言されます。ユーザー・イグジットは、このデータを正しいデータ型に移行する必要があります。リトル・エンディアン・プラットフォームでこのファンクションを使用すると、プロセスは 2 つのそれぞれのフィールドで " バイトを反転 " させます。

返される値
```
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_NOT_SUPPORTED
EXIT_FN_RET_OK
```

GET_RECORD_BUFFER

適用対象　Extract および Replicat

GET_RECORD_BUFFER ファンクションでは、カスタム列変換に関する情報を取得します。MAP または TABLE パラメータの COLMAP オプションでは十分に対応できないときは、ユーザー・イグジットを使用して異なるソースおよびターゲット・レコード間でデータをマッピングできます。たとえば、ユーザー・イグジットを使用して Enscribe データベースの固有の日付フィールド (たとえば YYDDD) をターゲット・レコードで標準の SQL 日付に変換し、他の列は COLMAP オプションを使用して Extract プロセスでマップできます。

SET_RECORD_BUFFER ファンクション (538 ページを参照してください) では、GET_RECORD_BUFFER で取得したデータを変更できます。ただし、これには内部 Oracle GoldenGate 正規フォーマットで書き込まれたデータ・レコードに対する理解が必要です。かわりの方法として、SET_COLUMN_VALUE_BY_INDEX ファンクション (531 ページを参照してください) または SET_COLUMN_VALUE_BY_NAME ファンクション (534 ページを参照してください) を使用して、データ・レコードの列値を設定できます。

削除、挿入、および更新は、バッファに完全なレコード・イメージで保持されます。

圧縮 SQL 更新は、次のフォーマットを持ちます。

```
<index><length><value>[<index><length><value>][...]
```

条件 :

- <index> は、表の列のリストへの 2 バイトの索引 (最初の列はゼロ) です。
- <length> は、表の 2 バイトの長さです。
- <value> は、実際の列値で、適切な場合に次の 2 バイトの NULL インジケータの 1 つが含まれます。`0` は非 NULL です。`-1` は NULL です。

SQL レコードの場合は、DECOMPRESS_RECORD ファンクション (483 ページを参照してください) を使用して操作のためにレコードを解凍した後、COMPRESS_RECORD ファンクション (482 ページを参照してください) を使用して、プロセスに予期されるとおりにもう一度圧縮できます。

圧縮 Enscribe 更新は、次のフォーマットを持ちます。

```
<offset><length><value>[<offset><length><value>][...]
```

条件 :

- <offset> は変更された Enscribe レコードのデータ・フラグメントのオフセットです。
- <length> はフラグメント長です。
- <value> はデータです。フラグメントはフィールド境界にまたがることがあるため、(圧縮が無効かまたは FETCHCOMPS が使用されている場合を除き) 完全なフィールドが常に取得されるわけではありません。

構文

```
#include "usrdecs.h"
short result_code;
record_def record;
ERCALLBACK (GET_RECORD_BUFFER, &record, &result_code);
```

バッファ

```
typedef struct
{
char *table_name;
char *buffer;
long length;
char before_after_ind;
short io_type;
short record_type;
short transaction_ind;
int64_t timestamp;
exit_ts_str io_datetime;
short mapped;
short source_or_target;
/* Version 2 CALLBACK_STRUCT_VERSION   */
char requesting_before_after_ind;
} record_def;
```

入力

次のようになります。

入力	説明
source_or_target	ソースまたはターゲット・データ・レコードのどちらのレコード・バッファを返すかを示す次の一方。 EXIT_FN_SOURCE_VAL EXIT_FN_TARGET_VAL
requesting_before_after_ind	オプションです。io_type が UPDATE_COMP_PK_SQL_VAL(主キー更新) のレコードのレコード・バッファをリクエストしているときに設定します。次のいずれかを使用して、主キー更新のどの部分にアクセスするかを指定します。デフォルトは AFTER_IMAGE_VAL です。 BEFORE_IMAGE_VAL AFTER_IMAGE_VAL

出力　次のようになります。

出力	説明
buffer	レコード・バッファへのポインタ。通常 buffer は、exit_rec_buf_def タイプのバッファへのポインタです。exit_rec_buf_def バッファには、Extract または Replicat にまもなく処理される実際のレコードが含まれます。このバッファは、コール・タイプが EXIT_CALL_DISCARD_RECORD のときに指定されます。イグジット・ルーチンは、たとえばカスタム・マッピング・ファンクションを実行するために、このバッファの内容を変えることがあります。 レコード・バッファの内容は、ユーザー・イグジットの文字セットとの変換が行われません。これはそのままで渡されます。
length	返されるレコード・バッファの長さ。
io_type	次のいずれかが返されます。 **DDL タイプ** SQL_DDL_VAL **DML タイプ** DELETE_VAL INSERT_VAL UPDATE_VAL **圧縮 Enscribe 更新** UPDATE_COMP_ENSCRIBE_VAL **圧縮 SQL 更新** UPDATE_COMP_SQL_VAL UPDATE_COMP_PK_SQL_VAL **その他** TRUNCATE_TABLE_VAL
mapped	これがマップされたレコード・バッファかどうかを示すフラグ (0 または 1)。

出力	説明
before_after_ind	レコードがビフォア・イメージかアフター・イメージかを示す次の一方。 BEFORE_IMAGE_VAL AFTER_IMAGE_VAL

返される値

EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_INVALID_PARAM
EXIT_FN_RET_OK

GET_RECORD_LENGTH

適用対象

Extract および Replicat

GET_RECORD_LENGTH ファンクションでは、データ・レコードの長さを取得します。かわりの方法として、GET_RECORD_BUFFER ファンクションでもデータ・レコードの長さを取得できます。

構文

```
#include "usrdecs.h"
short result_code;
record_def record;
ERCALLBACK (GET_RECORD_LENGTH, &record, &result_code);
```

バッファ

```
typedef struct
{
char *table_name;
char *buffer;
long length;
char before_after_ind;
short io_type;
short record_type;
short transaction_ind;
int64_t timestamp;
exit_ts_str io_datetime;
short mapped;
short source_or_target;
/* Version 2 CALLBACK_STRUCT_VERSION   */
char requesting_before_after_ind;
} record_def;
```

入力

次のようになります。

入力	説明
source_or_target	ソースまたはターゲット・データ・レコードのどちらのレコード長を返すかを示す次の一方。 EXIT_FN_SOURCE_VAL EXIT_FN_TARGET_VAL

出力

次のようになります。

出力	説明
length	返されるデータ・レコードの長さ。

返される値

```
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_INVALID_PARAM
EXIT_FN_RET_OK
```

GET_RECORD_TYPE

適用対象 Extract および Replicat

GET_RECORD_TYPE ファンクションでは、処理されているレコードのタイプを取得します。レコードは、SQL または Enscribe レコードです。各レコード・タイプはフォーマットが異なるため、レコード・バッファを操作するときにレコード・タイプは重要です。

構文

```
#include "usrdecs.h"
short result_code;
record_def record;
ERCALLBACK (GET_RECORD_TYPE, &record, &result_code);
```

バッファ

```
typedef struct
{
char *table_name;
char *buffer;
long length;
char before_after_ind;
short io_type;
short record_type;
short transaction_ind;
int64_t timestamp;
exit_ts_str io_datetime;
short mapped;
short source_or_target;
/* Version 2 CALLBACK_STRUCT_VERSION   */
char requesting_before_after_ind;
} record_def;
```

入力 次のようになります。

入力	説明
source_or_target	ソースまたはターゲット・データ・レコードのどちらのレコード・タイプを返すかを示す次の一方。 EXIT_FN_SOURCE_VAL EXIT_FN_TARGET_VAL

出力 次のようになります。

出力	説明
record_type	返されるレコード・タイプ。次のいずれかになります。 SQL レコードの場合 : EXIT_REC_TYPE_SQL Enscribe レコードの場合 : EXIT_REC_TYPE_ENSCRIBE

返される値

EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_INVALID_PARAM
EXIT_FN_RET_OK

GET_SCHEMA_NAME_ONLY

適用対象

Extract および Replicat

GET_SCHEMA_NAME_ONLY ファンクションでは、処理中のレコードに関連付けられているソースまたはターゲット表の、カタログや名前ではなく、スキーマを取得します。完全修飾表名を返すには、次を参照してください。

GET_TABLE_NAME

表名の他の部分を返すには、次を参照してください。

GET_TABLE_NAME_ONLY

GET_CATALOG_NAME_ONLY

データベース・オブジェクト名は、大 / 小文字の区別を含め、ホストしているデータベースで定義されているとおりに、正確に返されます。

構文

```
#include "usrdecs.h"
short result_code;
env_value_def env_value;
ERCALLBACK (GET_TABLE_NAME, &env_value, &result_code);
```

バッファ

```
typedef struct
{
char *buffer;
long max_length;
long actual_length;
short value_truncated;
short index;
short source_or_target;
} env_value_def;
```

入力

次のようになります。

入力	説明
buffer	返されるスキーマ名を受け付けるバッファへのポインタ。名前は NULL で終了します。
max_length	スキーマ名を受け付けるために割り当てたバッファの最大長。これは NULL 終了文字列として返されます。
source_or_target	ソースまたはターゲット・スキーマのどちらの名前を返すかを示す次の一方。 EXIT_FN_SOURCE_VAL EXIT_FN_TARGET_VAL

出力　次のようになります。

出力	説明
buffer	NULL で終了する完全修飾スキーマ名。 ユーザー・イグジットの文字セッションが、SET_SESSION_CHARSET を使用して、オペレーティング・システムのデフォルトの文字セット以外の値に設定されている場合、ucharset.h ファイルの ULIB_CS_DEFAULT で定義されているように、スキーマ名はセッションの文字セットで解釈されます。
actual length	返される名前の文字列長。実際の長さに NULL 終了文字は含まれません。
value_truncated	値が切り捨てられたかどうかを示すフラグ (0 または 1)。切捨ては、スキーマ名と NULL 終了文字を足した長さが、最大バッファ長を超えるときに行われます。

返される値
```
EXIT_FN_RET_INVALID_COLUMN
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_INVALID_PARAM
EXIT_FN_RET_OK
```

GET_SESSION_CHARSET

適用対象　Extract および Replicat

GET_SESSION_CHARSET では、現在のユーザー・イグジット・セッションの文字セットを取得します。この文字セットは、コールバック・ファンクション SET_SESSION_CHARSET を使用して設定できます。ユーザー・イグジット・セッションの文字セットは、次のような (ただしこれらに限定されません) メタデータなど、ユーザー・イグジットとコール元プロセス (Extract、データ・ポンプ、Replicat) 間で使用される文字ベースのコールバック構造メンバーのエンコーディングを示します。

- データベースの名前およびロケール
- 表名および列名
- DDL テキスト

- エラー・メッセージ
- CHAR および NCHAR などの文字型の列
- 文字列形式で表される日時および数値の列

セッションの文字セットの有効な値は、ヘッダー・ファイル ucharset.h で定義されます。このファンクションは、ユーザー・イグジットが制御しているときはいつでも呼び出すことができます。

構文

```
#include    usrdecs.h
short result_code;
session_def session_charset_def;
ERCALLBACK (GET_SESSION_CHARSET, &session_charset_def, &result_code);
```

バッファ

```
typedef struct
{
ULibCharSet  session_charset;
} session_def;
```

入力

なし

出力

```
session_charset_def.session_charset
```

返される値

```
EXIT_FN_RET_OK
```

GET_STATISTICS

適用対象

Extract および Replicat

GET_STATISTICS ファンクションでは、Extract または Replicat プロセスの現在の処理統計を取得します。たとえばユーザー・イグジットは、Extract または Replicat 処理中に致命的なエラーが発生した場合に、カスタム・レポートに統計を出力できます。

統計は、データをリクエストしたプロセスのタイプに基づいて自動的に処理されます。

- Extract プロセスは、常にリクエストをソース表として扱い、出力回数にかかわらずこの表を 1 回カウントします。
- Replicat プロセスは、常にリクエストをターゲット表のセットとして扱います。このセットには、ソース表の数にかかわらず、ターゲットへのすべてのカウントが含まれます。

データベースが大 / 小文字を区別する場合、オブジェクト名は、ホストしているデータベースで定義されているのと同じ大 / 小文字の区別で指定する必要があります。それ以外の場合、大 / 小文字の区別はありません。

構文

```
#include "usrdecs.h"
short result_code;
statistics_def statistics;
ERCALLBACK (GET_STATISTICS, &statistics, &result_code);
```

バッファ

```
typedef struct
{
char *table_name;
short group;
exit_timestamp_string start_datetime;
long num_inserts;
long num_updates;
long num_befores;
long num_deletes;
long num_discards;
long num_ignores;
long total_db_operations;
long total_operations;
/* Version 2 CALLBACK_STRUCT_VERSION   */
long num_truncates;
} statistics_def;
```

入力

次のようになります。

入力	説明	
table_name	ソース表名を指定する NULL 終了文字列。統計は常にソース・レコードに対して記録されます。ユーザー・イグジットの文字セッションが、SET_SESSION_CHARSET を使用して、オペレーティング・システムのデフォルトの文字セット以外の値に設定されている場合、ucharset.h ファイルの ULIB_CS_DEFAULT で定義されているように、表名および日付はセッションの文字セットで解釈されます。	
group	次のいずれかになります。	
	EXIT_STAT_GROUP_STARTUP	Oracle GoldenGate プロセスが最後に起動したときからの統計を取得します。
	EXIT_STAT_GROUP_DAILY	現在の日付の開始からの統計を取得します。
	EXIT_STAT_GROUP_HOURLY	現在の時間の開始からの統計を取得します。
	EXIT_STAT_GROUP_RECENT	GGSCI を使用して統計をリセットしてからの統計を取得します。
	EXIT_STAT_GROUP_REPORT	最後のレポートが生成されてからの統計を取得します。
	EXIT_STAT_GROUP_USEREXIT	ユーザー・イグジットが RESET_USEREXIT_STATS を使用して最後に統計をリセットしてからの統計を取得します。

出力

次のようになります。

出力	説明
start_datetime	指定のグループに対して統計の記録が開始されたローカルの日付と時刻を示す YYYY-MM-DD HH:MI:SS フォーマットの NULL 終了文字列。
num_inserts	返される Extract または Replicat が処理した挿入数。
num_updates	返される Extract または Replicat が処理した更新数。
num_befores	返される Extract または Replicat が処理した更新のビフォア・イメージ数。
num_deletes	返される Extract または Replicat が処理した削除数。
num_discards	返される Extract または Replicat が破棄したレコード数。
num_ignores	返される Extract または Replicat が無視したレコード数。
total_db_operations	返される Extract または Replicat が処理したデータベース操作合計数。
total_operations	返される Extract または Replicat が処理した合計操作数 (破棄および無視を含む)。
num_truncates	返される Extract または Replicat が処理した切捨て数。

返される値

```
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_INVALID_PARAM
EXIT_FN_RET_TABLE_NOT_FOUND
EXIT_FN_RET_OK
```

GET_TABLE_COLUMN_COUNT

適用対象 Extract および Replicat

GET_TABLE_COLUMN_COUNT ファンクションでは、表内の列の合計数 (キー列数を含む) を取得します。

構文

```
#include "usrdecs.h"
short result_code;
table_def table;
ERCALLBACK (GET_TABLE_COLUMN_COUNT, &table, &result_code);
```

バッファ

```
typedef struct
{
short num_columns;
short source_or_target;
/* Version 2 CALLBACK_STRUCT_VERSION   */
short num_key_columns;
} table_def;
```

入力

次のようになります。

入力	説明
source_or_target	ソースまたはターゲット表のどちらの列合計数を返すかを示す次の一方。 EXIT_FN_SOURCE_VAL EXIT_FN_TARGET_VAL

出力

次のようになります。

出力	説明
num_columns	返される指定の表の列の合計数。
num_key_columns	指定の表のキーとして Oracle GoldenGate に使用されている列の返される合計数。

返される値

```
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_INVALID_PARAM
EXIT_FN_RET_OK
```

GET_TABLE_METADATA

適用対象

Extract および Replicat

GET_TABLE_METADATA ファンクションでは、処理中のレコードに関連付けられている表のメタデータを取得します。

構文

```
#include "usrdecs.h"
short result_code;
table_metadata_def tbl_meta_rec;
ERCALLBACK (GET_TABLE_METADATA, &tbl_meta_rec, &result_code);
```

バッファ

```
typedef struct
{
char *table_name;
short value_truncated;
long max_name_length;
long actual_name_length;
short num_columns;
short num_key_columns;
short *key_columns;
short num_keys_returned;
BOOL using_pseudo_key;
short source_or_target;
} table_metadata_def;
```

入力　次のようになります。

入力	説明
table_name	table_name 戻り値を受け付けるバッファへのポインタ。
key_columns	key_columns 索引の配列へのポインタ。
max_name_length	返される表名の最大長。通常、最大長は表名のバッファの長さです。返される表名は NULL で終了しているため、最大長は表名の最大長と同じになるはずです。
source_or_target	ソースまたはターゲット表のどちらの名前を返すかを示す次の一方。 EXIT_FN_SOURCE_VAL EXIT_FN_TARGET_VAL

出力　次のようになります。

出力	説明
table_name	処理中のレコードに関連付けられている表の名前。ユーザー・イグジットの文字セッションが、SET_SESSION_CHARSET を使用して、オペレーティング・システムのデフォルトの文字セット以外の値に設定されている場合、ucharset.h ファイルの ULIB_CS_DEFAULT で定義されているように、表名はセッションの文字セットで解釈されます。
value_truncated	値が切り捨てられたかどうかを示すフラグ (0 または 1)。切捨ては、表名と NULL 終了文字を足した長さが、最大バッファ長を超えるときに行われます。
actual_name_length	返される表名の文字列長。実際の長さに NULL 終了文字は含まれません。
num_columns	表内の列数。

出力	説明
num_key_columns	Oracle GoldenGate に使用されているキー内の列数。
key_columns	キー列の値。キー数に列の長さを掛けた値を把握し、適切な量のバッファを割当てる必要があります。
num_keys_returned	リクエストされているキー列の数。
using_pseudo_key	KEYCOLS で指定されている列がキーとして使用されているかどうかを示すフラグ。TRUE または FALSE を返します。

返される値

```
EXIT_FN_RET_INVALID_PARAM
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_EXCEEDED_MAX_LENGTH
EXIT_FN_RET_OK
```

GET_TABLE_NAME

適用対象

Extract および Replicat

GET_TABLE_NAME ファンクションでは、処理中のレコードに関連付けられているソースまたはターゲット表の完全修飾名を取得します。完全修飾名の一部のみを返すには、次も参照してください、

GET_TABLE_NAME_ONLY

GET_SCHEMA_NAME_ONLY

GET_CATALOG_NAME_ONLY

データベース・オブジェクト名は、大 / 小文字の区別を含め、ホストしているデータベースで定義されているとおりに、正確に返されます。

構文

```
#include "usrdecs.h"
short result_code;
env_value_def env_value;
ERCALLBACK (GET_TABLE_NAME, &env_value, &result_code);
```

バッファ

```
typedef struct
{
char *buffer;
long max_length;
long actual_length;
short value_truncated;
short index;
short source_or_target;
} env_value_def;
```

入力

次のようになります。

入力	説明
buffer	返された表名を受け付けるバッファへのポインタ。表名は NULL で終了します。
max_length	表名を受け付けるために割り当てたバッファの最大長。これは NULL 終了文字列として返されます。
source_or_target	ソースまたはターゲット表のどちらの名前を返すかを示す次の一方。 EXIT_FN_SOURCE_VAL EXIT_FN_TARGET_VAL

出力　次のようになります。

出力	説明
buffer	schema.table や catalog.schema.table など、データベース・プラットフォームに応じた、NULL で終了する完全修飾表名。 ユーザー・イグジットの文字セッションが、SET_SESSION_CHARSET を使用して、オペレーティング・システムのデフォルトの文字セット以外の値に設定されている場合、ucharset.h ファイルの ULIB_CS_DEFAULT で定義されているように、表名はセッションの文字セットで解釈されます。
actual length	返される表名の文字列長。実際の長さに NULL 終了文字は含まれません。
value_truncated	値が切り捨てられたかどうかを示すフラグ (0 または 1)。切捨ては、表名と NULL 終了文字を足した長さが、最大バッファ長を超えるときに行われます。

返される値
```
EXIT_FN_RET_INVALID_COLUMN
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_INVALID_PARAM
EXIT_FN_RET_OK
```

GET_TABLE_NAME_ONLY

適用対象　Extract および Replicat

GET_TABLE_NAME_ONLY ファンクションでは、処理中のレコードに関連付けられているソースまたはターゲット表の、カタログやスキーマではなく、名前を取得します。完全修飾名を返すには、次を参照してください。

GET_TABLE_NAME

表名の他の部分を返すには、次を参照してください。

GET_SCHEMA_NAME_ONLY

GET_CATALOG_NAME_ONLY

データベース・オブジェクト名は、大 / 小文字の区別を含め、ホストしているデータベースで定義されているとおりに、正確に返されます。

構文

```
#include "usrdecs.h"
short result_code;
env_value_def env_value;
ERCALLBACK (GET_TABLE_NAME, &env_value, &result_code);
```

バッファ

```
typedef struct
{
char *buffer;
long max_length;
long actual_length;
short value_truncated;
short index;
short source_or_target;
} env_value_def;
```

入力

次のようになります。

入力	説明
`buffer`	返された表名を受け付けるバッファへのポインタ。表名は NULL で終了します。
`max_length`	表名を受け付けるために割り当てたバッファの最大長。これは NULL 終了文字列として返されます。

`source_or_target`	ソースまたはターゲット表のどちらの名前を返すかを示す次の一方。
	`EXIT_FN_SOURCE_VAL`
	`EXIT_FN_TARGET_VAL`

出力

次のようになります。

出力	説明
`buffer`	schema.table や catalog.schema.table など、データベース・プラットフォームに応じた、NULL で終了する完全修飾表名。 ユーザー・イグジットの文字セッションが、SET_SESSION_CHARSET を使用して、オペレーティング・システムのデフォルトの文字セット以外の値に設定されている場合、ucharset.h ファイルの ULIB_CS_DEFAULT で定義されているように、表名はセッションの文字セットで解釈されます。
`actual length`	返される表名の文字列長。実際の長さに NULL 終了文字は含まれません。
`value_truncated`	値が切り捨てられたかどうかを示すフラグ (0 または 1)。切捨ては、表名と NULL 終了文字を足した長さが、最大バッファ長を超えるときに行われます。

返される値

```
EXIT_FN_RET_INVALID_COLUMN
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_INVALID_PARAM
EXIT_FN_RET_OK
```

GET_TIMESTAMP

適用対象 Extract および Replicat

GET_TIMESTAMP ファンクションでは、ソース・データ・レコードに関連付けられている I/O タイムスタンプを ASCII 日付時刻フォーマットで取得します。タイムスタンプは、その後ローカル時間に変換され、元のデータベース操作の時間に概算されます。

注意 ASCII コミット・タイムスタンプは、地域によって異なる夏時間の使用によって変化する可能性があります。ユーザー・イグジット・コールバックは、この差異を回避するために、GMT 時間として ASCII 日付時刻を返す必要があります。Oracle GoldenGate トレイルは、GMT フォーマットを使用します。GET_GMT_TIMESTAMP を参照してください。

構文

```
#include "usrdecs.h"
short result_code;
record_def record;
ERCALLBACK (GET_TIMESTAMP, &record, &result_code);
```

バッファ

```
typedef struct
{
char *table_name;
char *buffer;
long length;
char before_after_ind;
short io_type;
short record_type;
short transaction_ind;
int64_t timestamp;
exit_ts_str io_datetime;
short mapped;
short source_or_target;
/* Version 2 CALLBACK_STRUCT_VERSION   */
char requesting_before_after_ind;
} record_def;
```

入力 なし

出力 次のようになります。

出力	説明
timestamp	返される ASCII フォーマットの 64 ビット I/O タイムスタンプ。
io_datetime	ローカル I/O 日付および時刻を含む、次のフォーマットの NULL 終了文字列。 YYYY-MM-DD HH:MI:SS.FFFFFF

返される値

```
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_OK
```

GET_TRANSACTION_IND

適用対象

Extract および Replicat

GET_TRANSACTION_IND ファンクションでは、データ・レコードがトランザクションの最初、最後、または中間の操作かを確認します。これは、たとえばユーザー・イグジットが各トランザクションの詳細を作成し、特別の概要レコードを出力するときに役立ちます。

構文

```
#include "usrdecs.h"
short result_code;
record_def record;
ERCALLBACK (GET_TRANSACTION_IND, &record, &result_code);
```

バッファ

```
typedef struct
{
char *table_name;
char *buffer;
long length;
char before_after_ind;
short io_type;
short record_type;
short transaction_ind;
int64_t timestamp;
exit_ts_str io_datetime;
short mapped;
short source_or_target;
/* Version 2 CALLBACK_STRUCT_VERSION   */
char requesting_before_after_ind;
} record_def;
```

入力

なし

出力

次のようになります。

出力	説明	
transaction_ind	返されるトランザクション・インジケータで、次のいずれかで表されます。	
	BEGIN_TRANS_VAL	レコードはトランザクションの最初です。
	MIDDLE_TRANS_VAL	レコードはトランザクションの中間です。
	END_TRANS_VAL	レコードはトランザクションの最後です。
	WHOLE_TRANS_VAL	レコードはトランザクションの唯一のレコードです。

返される値

```
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_OK
```

GET_USER_TOKEN_VALUE

適用対象 Extract および Replicat

GET_USER_TOKEN_VALUE ファンクションでは、トレイル・レコードからユーザー・トークン値を取得します。トークン値では、文字セットの変換は実行されません。

構文

```
#include "usrdecs.h"
```

バッファ

```
typedef struct
{
char *token_name;
char *token_value;
long max_length;
long actual_length;
short value_truncated;
} token_value_def;
```

入力 次のいずれかになります。

入力	説明
token_name	トークン名を表すバッファへのポインタ。トークン名は、トークンが構成される Extract の TABLE 文をホストするオペレーティング・システムのデフォルトの文字セットでエンコードされているとみなされます。ユーザー・イグジットは、SET_SESSION_CHARSET で指定された文字セットでトークン名を準備しますが、一致するトークン値を取得する前に、オペレーティング・システムの文字セットに戻します。
max_length	結果のトークン値を受け付けるために割り当てた token_name バッファの最大長。これは NULL 終了文字列として返されます。

出力 次のようになります。

入力	説明
token_value	(ある場合) トークンの戻り値を表すバッファへのポインタ。トークン値は、文字セットの変換を行わずに、ユーザー・イグジットにそのまま戻されます。
actual_length	返されるトークン値の実際の長さ。トークンが見つかっても値が存在しないときは、値 0 が返されます。
value_truncated	トークン値が切り捨てられたかどうかを示すフラグ (0 または 1)。切捨ては、表名と NULL 終了文字を足した長さが、最大バッファ長を超えるときに行われます。

返される値

```
EXIT_FN_RET_INVALID_PARAM
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_TOKEN_NOT_FOUND
EXIT_FN_RET_OK
```

OUTPUT_MESSAGE_TO_REPORT

適用対象 Extract および Replicat

OUTPUT_MESSAGE_TO_REPORT ファンクションでは、レポート・ファイルにメッセージを出力します。ユーザー・イグジットの文字セッションが SET_SESSION_CHARSET を使用して設定されている場合、メッセージはセッションの文字セットで解釈されますが、レポート・ファイルに書き込まれる前に、オペレーティング・システムのデフォルトの文字セットに変換されます。

構文

```
#include "usrdecs.h"
short result_code;
char  message[500];
ERCALLBACK (OUTPUT_MESSAGE_TO_REPORT, message, &result_code);
```

バッファ なし

入力 次のようになります。

入力	説明
message	NULL 終了文字列。

出力 なし

返される値 EXIT_FN_RET_OK

RESET_USEREXIT_STATS

適用対象 Extract および Replicat

RESET_USEREXIT_STATS ファンクションでは、最後の GET_STATISTICS へのコールが処理された後の Oracle GoldenGate プロセスの EXIT_STAT_GROUP_USEREXIT 統計をリセットします。このファンクションにより、ユーザー・イグジットは GET_STATISTICS ファンクションによって返されるグループ統計をいつリセットするかを制御できますが、他の統計のリセットは許可されません。

構文

```
#include "usrdecs.h"
short result_code;
call_callback (RESET_USEREXIT_STATS, NULL, &result_code);
```

入力 なし

出力 なし

返される値 なし

SET_COLUMN_VALUE_BY_INDEX

適用対象 Extract および Replicat

SET_COLUMN_VALUE_BY_INDEX または SET_COLUMN_VALUE_BY_NAME ファンクションでは、データ・レコード全体を操作せず、単一の列値のみを変更します。ユーザー・イグジットの文字セッションが、SET_SESSION_CHARSET を使用して、オペレーティング・システムのデフォルトの文字セット以外の値に設定されている場合、ucharset.h ファイルの ULIB_CS_DEFAULT で定義されているように、ユーザー・イグ

ジットとプロセス間で交換される文字データは、セッションの文字セットで解釈されます。

次に該当する場合、列値はセッションの文字セットにのみ設定されます。

- 列値がSQL文字型(CHAR/VARCHAR2/CLOB、NCHAR/NVARCHAR2/NCLOB)、SQL日付/タイムスタンプ/間隔/数値型である。
- column_value_mode インジケータが EXIT_FN_CNVTED_SESS_CHAR_FORMAT に設定されている。

構文

```
#include "usrdecs.h"
short result_code;
column_def column;
ERCALLBACK (SET_COLUMN_VALUE_BY_INDEX, &column, &result_code);
```

バッファ

```
typedef struct
{
char *column_value;
unsigned short max_value_length;
unsigned short actual_value_length;
short null_value;
short remove_column;
short value_truncated;
short column_index;
char *column_name;
/* Version 3 CALLBACK_STRUCT_VERSION */
short column_value_mode;
short source_or_target;
/* Version 2 CALLBACK_STRUCT_VERSION   */
char requesting_before_after_ind;
char more_lob_data;
/* Version 3 CALLBACK_STRUCT_VERSION */
ULibCharSet column_charset;
} column_def;
```

入力

次のようになります。

入力	説明
column_value	新しい列値を表すバッファへのポインタ。
actual_value_length	新しい列値の長さ(バイト)。新しい列値がASCIIフォーマットの場合、実際の長さにNULL終了文字は含めません。
null_value	新しい列値がNULLかどうかを示すフラグ(0または1)。null_valueフラグが1に設定されている場合、データ・レコードの列値はNULLに設定されます。
remove_column	存在する場合に、圧縮更新の列を削除するかどうかを示すフラグ(0または1)。このフラグは、レコードの操作タイプがUPDATE_COMP_SQL_VAL、PK_UPDATE_SQL_VAL、またはUPDATE_COMP_ENSCRIBE_VALの場合にのみ設定する必要があります。
column_index	データ・レコード・バッファにコピーする新しい列値の列索引。列索引はゼロから開始されます。

入力	説明
column_value_mode	列値のフォーマットを示します。 EXIT_FN_CHAR_FORMAT EXIT_FN_RAW_FORMAT EXIT_FN_CNVTED_SESS_CHAR_FORMAT **EXIT_FN_CHAR_FORMAT** ASCII フォーマット : 値は、NULL で終了している ASCII (または EBCDIC) 文字列です (既知の例外としてサブデータ型 UTF16_BE があり、これは UTF8 に変換されます)。 **注意** : 列値は、ASCII 文字列として解釈され、NULL で終了する必要があるため、ユーザー・イグジットに示される際に切り捨てられる可能性があります。最初の 0 の値は、文字列の終了文字になります。 ◆ 日付は CCYY-MM-DD HH:MI:SS.FFFFFF のフォーマットで、時間の端数はデータベースに依存します。 ◆ 数値は文字列フォーマットです。たとえば、123.45 は "123.45" として表されます。 ◆ 出力不可能な文字またはバイナリ値は、16 進表記法に変換されます。 ◆ 浮動小数点型は、有効桁数が最初の 14 桁までの NULL 終了文字列として出力されます。 **EXIT_FN_RAW_FORMAT** 内部 Oracle GoldenGate 正規フォーマット : このフォーマットには、適切な場合に、2 バイトの NULL インジケータおよび 2 バイトの変数データ長が含まれます。文字データ型の場合、Oracle GoldenGate ではこのフォーマットへの文字セットの変換は実行されません。 **EXIT_FN_CNVTED_SESS_CHAR_FORMAT** ユーザー・イグジットの文字セット : これが適用されるのは、列のデータ型が次の場合のみです。 ◆ シングルバイトまたはマルチバイトの文字ベースの型 ◆ 文字列で表現する数値型 このフォーマットは NULL で終了しません。
source_or_target	ソースまたはターゲット・レコードが変更されるかを示す次の一方。 EXIT_FN_SOURCE_VAL EXIT_FN_TARGET_VAL

入力	説明
requesting_before_after_ind	io_type が UPDATE_COMP_PK_SQL_VAL(主キー更新) のレコードの列値を設定するときに設定します。次のいずれかを使用して、主キー更新のどの部分にアクセスするかを指定します。デフォルトは AFTER_IMAGE_VAL です。 ◆ BEFORE_IMAGE_VAL ◆ AFTER_IMAGE_VAL

出力　なし

返される値

```
EXIT_FN_RET_BAD_COLUMN_DATA
EXIT_FN_RET_INVALID_COLUMN
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_INVALID_PARAM
EXIT_FN_RET_OK
EXIT_FN_RET_NOT_SUPPORTED
EXIT_FN_RET_INVALID_COLUMN_TYPE
```

SET_COLUMN_VALUE_BY_NAME

適用対象　Extract および Replicat

SET_COLUMN_VALUE_BY_NAME または SET_COLUMN_VALUE_BY_INDEX ファンクションでは、データ・レコード全体を操作せず、単一の列値のみを変更します。

ユーザー・イグジットの文字セッションが、SET_SESSION_CHARSET を使用して、オペレーティング・システムのデフォルトの文字セット以外の値に設定されている場合、ucharset.h ファイルの ULIB_CS_DEFAULT で定義されているように、ユーザー・イグジットとプロセス間で交換される文字データは、セッションの文字セットで解釈されます。

次に該当する場合、列値はセッションの文字セットにのみ設定されます。

- 列値が SQL 文字型 (CHAR/VARCHAR2/CLOB、NCHAR/NVARCHAR2/NCLOB)、SQL 日付 / タイムスタンプ / 間隔 / 数値型である。
- column_value_mode インジケータが EXIT_FN_CNVTED_SESS_CHAR_FORMAT に設定されている。

データベースが大 / 小文字を区別する場合、オブジェクト名は、ホストしているデータベースで定義されているのと同じ大 / 小文字の区別で指定する必要があります。それ以外の場合、大 / 小文字の区別はありません。

構文

```
#include "usrdecs.h"
short result_code;
column_def column;
ERCALLBACK (SET_COLUMN_VALUE_BY_NAME, &column, &result_code);
```

バッファ

```
typedef struct
{
char *column_value;
unsigned short max_value_length;
unsigned short actual_value_length;
short null_value;
short remove_column;
```

```
short value_truncated;
short column_index;
char *column_name;
/* Version 3 CALLBACK_STRUCT_VERSION */
short column_value_mode;
short source_or_target;
/* Version 2 CALLBACK_STRUCT_VERSION   */
char requesting_before_after_ind;
char more_lob_data;
/* Version 3 CALLBACK_STRUCT_VERSION */
ULibCharSet column_charset;
} column_def;
```

入力　次のようになります。

入力	説明
column_value	新しい列値を表すバッファへのポインタ。
actual_value_length	新しい列値の長さ (バイト)。新しい列値が ASCII フォーマットの場合、実際の長さに NULL 終了文字は含めません。
null_value	新しい列値が NULL かどうかを示すフラグ (0 または 1)。null_value フラグが 1 に設定されている場合、データ・レコードの列値は NULL に設定されます。
remove_column	存在する場合に、圧縮更新の列を削除するかどうかを示すフラグ (0 または 1)。このフラグは、レコードの操作タイプが UPDATE_COMP_SQL_VAL、PK_UPDATE_SQL_VAL、または UPDATE_COMP_ENSCRIBE_VAL の場合にのみ設定する必要があります。
column_name	データ・レコード・バッファにコピーする新しい列値に対応する列名。
column_value_mode	列値のフォーマットを示します。 EXIT_FN_CHAR_FORMAT EXIT_FN_RAW_FORMAT EXIT_FN_CNVTED_SESS_CHAR_FORMAT **EXIT_FN_CHAR_FORMAT** ASCII フォーマット : 値は、NULL で終了している ASCII (または EBCDIC) 文字列です (既知の例外としてサブデータ型 UTF16_BE があり、これは UTF8 に変換されます)。

入力	説明
	注意 : 列値は、ASCII 文字列として解釈され、NULL で終了する必要があるため、ユーザー・イグジットに示される際に切り捨てられる可能性があります。最初の 0 の値は、文字列の終了文字になります。 ◆ 日付は CCYY-MM-DD HH:MI:SS.FFFFFF のフォーマットで、時間の端数はデータベースに依存します。 ◆ 数値は文字列フォーマットです。たとえば、123.45 は "123.45" として表されます。 ◆ 出力不可能な文字またはバイナリ値は、16 進表記法に変換されます。 ◆ 浮動小数点型は、有効桁数が最初の 14 桁までの NULL 終了文字列として出力されます。 **EXIT_FN_RAW_FORMAT** 内部 Oracle GoldenGate 正規フォーマット : このフォーマットには、適切な場合に、2 バイトの NULL インジケータおよび 2 バイトの変数データ長が含まれます。文字データ型の場合、Oracle GoldenGate ではこのフォーマットへの文字セットの変換は実行されません。 **EXIT_FN_CNVTED_SESS_CHAR_FORMAT** ユーザー・イグジットの文字セット : これが適用されるのは、列のデータ型が次の場合のみです。 ◆ シングルバイトまたはマルチバイトの文字ベースの型 ◆ 文字列で表現する数値型 このフォーマットは NULL で終了しません。

`source_or_target`	ソースまたはターゲット・データ・レコードが変更されるかを示す次の一方。
	`EXIT_FN_SOURCE_VAL`
	`EXIT_FN_TARGET_VAL`

`requesting_before_after_ind`	io_type が UPDATE_COMP_PK_SQL_VAL(主キー更新) のレコードの列値を設定するときに設定します。次のいずれかを使用して、主キー更新のどの部分にアクセスするかを指定します。デフォルトは AFTER_IMAGE_VAL です。 ◆ BEFORE_IMAGE_VAL ◆ AFTER_IMAGE_VAL

出力 なし

返される値

```
EXIT_FN_RET_BAD_COLUMN_DATA
EXIT_FN_RET_INVALID_COLUMN
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_INVALID_PARAM
EXIT_FN_RET_OK
EXIT_FN_RET_NOT_SUPPORTED
EXIT_FN_RET_INVALID_COLUMN_TYPE
```

SET_OPERATION_TYPE

適用対象 Extract および Replicat

SET_OPERATION_TYPE ファンクションでは、レコードに関連付けられている操作のタイプを変更します。たとえば、特定の表での削除を別の表で挿入に変更できます。レコード・ヘッダーのビフォア / アフター・インジケータは、挿入および削除操作に対して適切に変更されます。

構文

```
#include "usrdecs.h"
short result_code;
record_def record;
ERCALLBACK (SET_OPERATION_TYPE, &record, &result_code);
```

バッファ

```
typedef struct
{
char *table_name;
char *buffer;
long length;
char before_after_ind;
short io_type;
short record_type;
short transaction_ind;
int64_t timestamp;
exit_ts_str io_datetime;
short mapped;
short source_or_target;
/* Version 2 CALLBACK_STRUCT_VERSION   */
char requesting_before_after_ind;
} record_def;
```

入力 次のようになります。

入力	説明
io_type	削除、挿入、および更新に、次の 1 つがそれぞれ返されます。 DELETE_VAL INSERT_VAL UPDATE_VAL 圧縮 Enscribe 更新の場合は、次が返されます。 UPDATE_COMP_ENSCRIBE_VAL

入力	説明
	圧縮 SQL 更新の場合は、次が返されます。 UPDATE_COMP_SQL_VAL 新しい操作タイプが挿入または削除の場合、レコードのビフォア / アフター・インジケータは次のいずれかに設定されます。 **挿入 :** AFTER_IMAGE_VAL(アフター・イメージ) **削除 :** BEFORE_IMAGE_VAL(ビフォア・イメージ)
source_or_target	ソースまたはターゲット・データ・レコードのどちらの操作タイプを設定するかを示す次の一方。 EXIT_FN_SOURCE_VAL EXIT_FN_TARGET_VAL

出力 なし

返される値
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_INVALID_PARAM
EXIT_FN_RET_OK

SET_RECORD_BUFFER

適用対象 Extract および Replicat

SET_RECORD_BUFFER ファンクションは、ユーザー・イグジットとの互換性の維持、および複雑なデータ・レコード操作のために使用します。このファンクションでは、レコード全体を操作します。データ・レコード・バッファを直接正確に変更するには、Oracle GoldenGate 内部レコード・フォーマットを把握する必要があるため、変更を行うときは、レコード全体ではなく個々の列値を変更する方法が最適です。列値を変更するには、SET_COLUMN_VALUE_BY_INDEX および SET_COLUMN_VALUE_BY_NAME ファンクションを使用します。ユーザー・イグジット内のこれらのファンクションで、ほとんどのカスタム・マッピングを十分に処理できます。

構文

```
#include "usrdecs.h"
short result_code;
record_def record;
ERCALLBACK (SET_RECORD_BUFFER, &record_def, &result_code);
```

バッファ

```
typedef struct
{
char *table_name;
char *buffer;
long length;
char before_after_ind;
short io_type;
short record_type;
short transaction_ind;
int64_t timestamp;
exit_ts_str io_datetime;
short mapped;
short source_or_target;
/* Version 2 CALLBACK_STRUCT_VERSION   */
char requesting_before_after_ind;
} record_def;
```

入力

次のようになります。

入力	説明
buffer	新しいレコード・バッファへのポインタ。通常、buffer はタイプ exit_rec_buf_def のバッファへのポインタです。exit_rec_buf_def バッファには、Extract または Replicat にまもなく処理される実際のレコードが含まれます。このバッファは、コール・タイプが EXIT_CALL_DISCARD_RECORD のときに指定されます。イグジット・ルーチンは、たとえばカスタム・マッピング・ファンクションを実行するために、このバッファの内容を変えることがあります。 レコード・バッファの内容は、ユーザー・イグジットの文字セットとの変換が行われません。これはそのままで渡されます。
length	新しいレコード・バッファの長さ。

出力

なし

返される値

```
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_INVALID_PARAM
EXIT_FN_RET_OK
EXIT_FN_RET_NOT_SUPPORTED
```

SET_SESSION_CHARSET

適用対象

Extract および Replicat

SET_SESSION_CHARSET ファンクションでは、ユーザー・イグジットの文字セットを設定します。ユーザー・イグジット・セッションの文字セットは、次のような (ただしこれらに限定されません) メタデータなど、ユーザー・イグジットとコール元プロセス (Extract、データ・ポンプ、Replicat) 間で使用される文字ベースのコールバック構造メンバーのエンコーディングを示します。

- データベースの名前およびロケール
- 表名および列名
- DDL テキスト
- エラー・メッセージ

- CHAR および NCHAR などの文字型の列
- 文字列形式で表される日時および数値の列

このファンクションは、ユーザー・イグジットが制御しているときはいつでも呼び出すことができます。ユーザー・イグジットがセッションの文字セットを設定すると、すぐに有効になり、すべての文字値は指定したセットへの変換を開始します。このファンクションを呼び出すには、コール・タイプ EXIT_CALL_START を使用することをお薦めします。

注意 SET_SESSION_CHARSET はスレッドセーフではありません。

SET_SESSION_CHARSET が呼び出されない場合、セッションは、ucharset.h ファイルの ULIB_CS_DEFAULT の事前定義された列挙型の値であるオペレーティング・システムのデフォルトの文字セットに設定されます。セッションの文字セットが ULIB_CS_DEFAULT からのデフォルトである場合、ユーザー・イグジットとコール元プロセス間で交換される文字型の値に対して、Oracle GoldenGate では変換は実行されません。さらに、データベースのオブジェクト名のメタデータは、オペレーティング・システムのデフォルトの文字セットであるとみなされます。デフォルトは適切ではない場合があることに注意してください。

ユーザー・イグジットがロードされ、SET_SESSION_CHARSET が呼び出されると、ユーザー・イグジットの文字セットはレポート・ファイルに出力されます。セッションの文字セットが ULIB_CS_DEFAULT である場合、列データの文字セットの変換が実行されないことを示すメッセージが表示されます。

グローバリゼーション・サポートの詳細は、『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』を参照してください。

構文

```
#include    usrdecs.h
short result_code;
session_def session_charset_def;
ERCALLBACK (SET_SESSION_CHARSET, &session_charset_def, &result_code);
```

バッファ

```
typedef struct
{
ULibCharSet  session_charset;
} session_def;
```

入力 次のようになります。

入力	説明
session_charset	セッションの文字セットの有効な値は、ヘッダー・ファイル ucharset.h で定義されます。

出力 なし

返される値 EXIT_FN_RET_OK

SET_TABLE_NAME

適用対象 Extract およびデータ・ポンプ

SET_TABLE_NAME ファンクションでは、レコードに関連付けられている表名を変更します。たとえば、特定の表での削除を履歴表への挿入に変更できます。表名は、Extract 処理中にのみ変更できます。

データベースが大 / 小文字を区別する場合、オブジェクト名は、ホストしているデータベースで定義されているのと同じ大 / 小文字の区別で指定する必要があります。それ以外の場合、大 / 小文字の区別はありません。

構文

```
#include "usrdecs.h"
short result_code;
record_def record;
ERCALLBACK (SET_TABLE_NAME, &record_def, &result_code);
```

バッファ

```
typedef struct
{
char *table_name;
char *buffer;
long length;
char before_after_ind;
short io_type;
short record_type;
short transaction_ind;
int64_t timestamp;
exit_ts_str io_datetime;
short mapped;
short source_or_target;
/* Version 2 CALLBACK_STRUCT_VERSION   */
char requesting_before_after_ind;
} record_def;
```

入力

次のようになります。

入力	説明
table_name	データ・レコードに関連付ける新しい表名を指定する NULL 終了文字列。 ユーザー・イグジットの文字セッションが、SET_SESSION_CHARSET を使用して、オペレーティング・システムのデフォルトの文字セット以外の値に設定されている場合、ucharset.h ファイルの ULIB_CS_DEFAULT で定義されているように、表名はセッションの文字セットで解釈されます。

出力

なし

返される値

```
EXIT_FN_RET_INVALID_CONTEXT
EXIT_FN_RET_INVALID_PARAM
EXIT_FN_RET_OK
```

索引

記号

数字

A

B

C

D

E

F

G

H

I

K

L

M

N

Q

R

S

T

U

V

W

X

ア

イ

ウ

エ

オ

カ

キ

ク

シ

ス

セ

ソ

タ

チ

ツ

テ

ト

ナ

ニ

ネ

ハ

ヒ

フ

ヘ

ロ

ワ