Oracle® GoldenGate

Windows and UNIX 管理者ガイド

11*g* リリース 1 パッチ・セット 1 (11.1.1.1)

E47932-01 (原本部品番号 : E21513-01)

2013 年 8 月

Oracle GoldenGate Windows and UNIX 管理者ガイド 11g リリース 1 パッチ・セット 1 (11.1.1.1)

E47932-01 (原本部品番号 : E21513-01)

目次

はじめに

Oracle GoldenGate のドキュメントについて

Oracle GoldenGate のドキュメント・セット全体は、次の各要素で構成されます。

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 Oracle GoldenGate レプリケーション・ソリューションを計画、構成および実装する方法について説明しています。
- 『Oracle GoldenGate *HP NonStop リファレンス・ガイド*』: NonStop プラットフォームを対象とする Oracle GoldenGate のパラメータ、コマンドおよび関数の詳細情報が含まれます。

Windows、UNIX、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るデータベースごとにこのガイドが用意されています。このガイドには、システム要件、インストール前とインストール後の手順、インストール手順など、Oracle GoldenGate レプリケーション・ソリューションをインストールするためのシステム固有の情報が含まれます。
- 『*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 Oracle GoldenGate レプリケーション・ソリューションを計画、構成および実装する方法について説明しています。
- 『*Oracle GoldenGate Windows and UNIX リファレンス・ガイド*』: Windows および UNIX プラットフォームを対象とする Oracle GoldenGate のパラメータ、コマンドおよび関数の詳細情報が含まれます。
- 『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向上するための推奨事項と、一般的な問題に対する解決策を提供しています。

その他の Oracle GoldenGate 製品

- 『*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を行うために、Oracle GoldenGate Director をインストール、実行および管理する方法について説明しています。
- 『*Oracle GoldenGate Veridata 管理者ガイド*』: Oracle GoldenGate Veridata データ比較ソリューションをインストール、実行および管理する方法について説明しています。
- 『*Oracle GoldenGate for Java 管理者ガイド*』: JMS メッセージを Oracle GoldenGate 証跡に取得したり、取得したデータをメッセージ・システムまたはカスタム API に配信するために、Oracle GoldenGate for Java をインストール、構成および実行する方法について説明しています。
- 『*Oracle GoldenGate for Flat File 管理者ガイド*』: ETL アプリケーション、プロプライエタリ・アプリケーションまたはレガシー・アプリケーションに対するバッチ入力として Oracle GoldenGate によって取得されたデータをフォーマットするために、Oracle GoldenGate for Flat File をインストール、構成および実行する方法について説明しています。

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。

- パラメータおよびコマンド引数は、大文字で表記されます。次に例を示します。
  ```
  CHECKPARAMS
  ```
- ファイル名や表名などの名前は、小文字で表記されます (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それらの名前の大 / 小文字が区別されない場合)。次に例を示します。
  ```
  account_tab
  GLOBALS
  ```
- 変数は、< > 文字内に表記されます。次に例を示します。
  ```
  <group name>
  ```
- 複数の相互排他的な引数のうちから 1 つを選択する必要がある場合、その選択肢は、中カッコで囲まれてパイプ文字で区切られます。次に例を示します。
  ```
  VIEW PARAMS {MGR | <group> | <file name>}
  ```
- オプション引数は、大カッコで囲まれます。次に例を示します。
  ```
  CLEANUP EXTRACT <group name> [, SAVE <count>]
  ```
- 非常に多くのオプション引数がある場合、[<option>] などのプレースホルダを使用することがあります。これらのオプションのリストと説明は、別途記載されます。次に例を示します。
  ```
  TRANLOGOPTIONS [<option>]
  ```
- 引数を繰り返し使用できる場合、省略記号 (...) が使用されます。次に例を示します。
  ```
  PARAMS ([<requirement rule>] <param spec> [, <param spec>] [, ...])
  ```
- アンパサンド (&) は、Oracle GoldenGate のパラメータ・ファイル内で継続文字として使用されます。この文字は、複数行にわたるパラメータ文の各行の最後に配置する必要があります。このドキュメントのほとんどの例では、適切な位置にアンパサンドを記載していますが、複数行にわたる文の一部の例では、発行形式によるスペース上の制約に対応するため、アンパサンドを省略しています。

Oracle GoldenGate のヘルプの取得

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

Oracle GoldenGate インタフェースでのヘルプの取得

GGSCI および Oracle GoldenGate Director アプリケーションには、両方ともオンライン・ヘルプが付属します。

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・カテゴリのサマリーを取得するには、オプションなしで 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、コマンドの構文および説明が表示されます。

Oracle GoldenGate Director

Oracle GoldenGate Director クライアントまたは Oracle GoldenGate Director Web のいずれかに関するヘルプを取得するには、そのアプリケーション内で**「Help」**メニューを使用します。

質問および問題のヘルプの取得

トラブルシューティングのヘルプは、『*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。追加情報は、http://support.oracle.com にあるナレッジ・ベースから取得できます。回答が見つからない場合、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 の概要

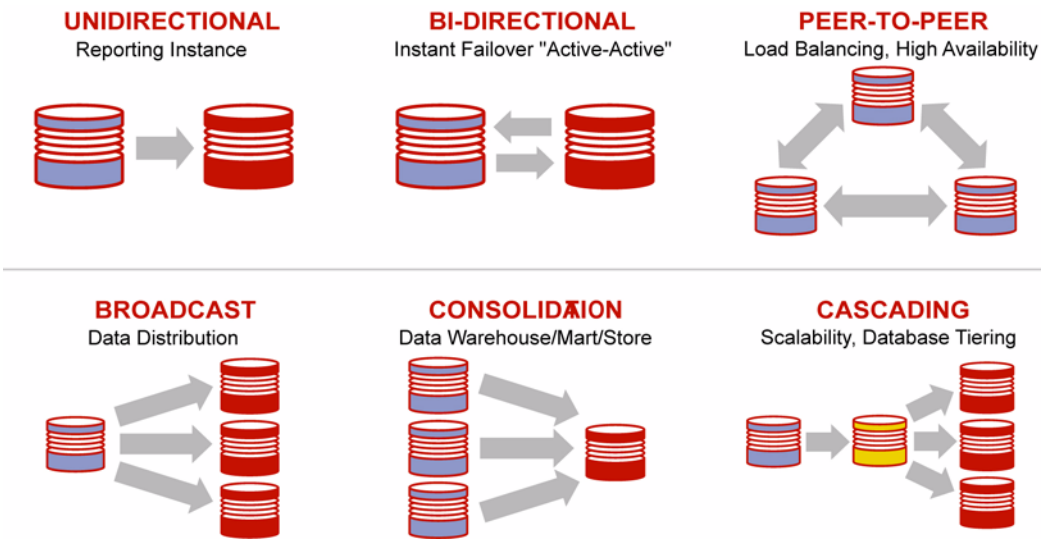
Oracle GoldenGate でサポートされる処理方法およびデータベース

Oracle GoldenGate では、エンタープライズ全体にわたる複数の異機種プラットフォーム間において、トランザクション・レベルでデータを交換および操作できます[1]。そのモジュール型のアーキテクチャによって、選択したデータ・レコード、トランザクション変更および DDL(データ定義言語[2]) 変更を様々なトポロジ間で柔軟に抽出およびレプリケートできます。

この柔軟性と、Oracle GoldenGate のフィルタリング、トランスフォーメーション (変換) およびカスタム処理の各機能により、次のような多くのビジネス要件に対応できます。

- ビジネス継続性および高可用性。
- 初期ロードおよびデータベース移行。
- データ統合。
- 意志決定支援およびデータ・ウェアハウス。

図 1 Oracle GoldenGate でサポートされるトポロジ



1. 異なるデータベース・タイプおよびトポロジ間でのレプリケーションのサポートは、データベース・タイプごとに異なります。サポートされる構成の詳細情報は、使用中のデータベースに対応する Oracle GoldenGate の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。

2. DDL がサポートされないデータベースもあります。

表 1　　サポートされる処理方法[1]

データベース	ログベース抽出（取得）	非ログベース抽出 **（取得）	レプリケーション（配信）
c-tree***	X		X
DB2 for i*			X
DB2 for Linux、UNIX、Windows	X		X
DB2 for z/OS	X		X
Oracle	X		X
MySQL	X		X
SQL/MX	X		X
SQL Server	X		X
Sybase	X		X
Teradata		X	X
TimesTen*			X
汎用 ODBC*			X

[1] 処理方法、サポートされるトポロジと機能、および構成要件の詳細は、使用中のデータベースに対応する Oracle GoldenGate の*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。

* ターゲット・データベースとしてのみサポートされます。Oracle GoldenGate 抽出のソース・データベースには指定できません。

** Oracle GoldenGate API と通信する取得モジュールを使用して、変更データを Oracle GoldenGate に送信します。

*** 同類構成のみがサポートされます。データ操作、フィルタリング、列マッピングはサポートされません。

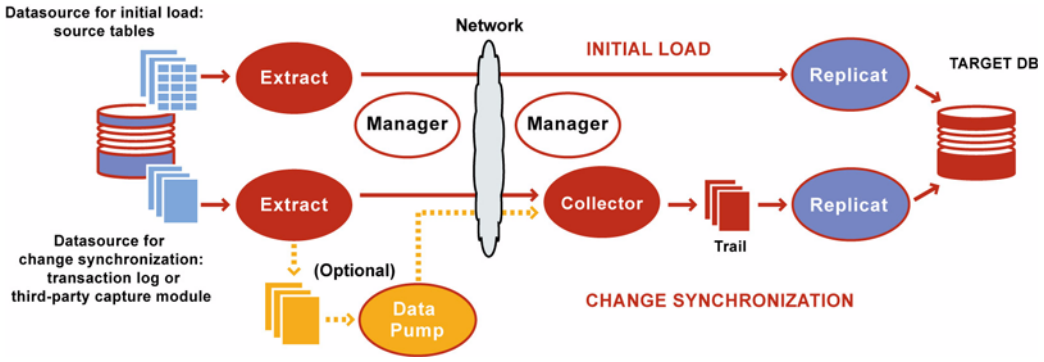
Oracle GoldenGate アーキテクチャの概要

Oracle GoldenGate は、次のコンポーネントで構成されます。

- Extract
- データ・ポンプ
- Replicat
- 証跡または抽出ファイル
- チェックポイント
- Manager
- Collector

図 2 は、初期データ・ロードと実行中のデータベース変更のレプリケーションに対応する Oracle GoldenGate の論理アーキテクチャを示しています。これは基本構成です。このモデルは、ビジネス要件に応じて変更することをお薦めします。

図 2 Oracle GoldenGate の論理アーキテクチャ



Extract の概要

Extract プロセスは、ソース・システム上で実行される、Oracle GoldenGate の抽出 (取得) メカニズムです。Extract は次のいずれかの方法で構成できます。

- **初期ロード :** 初期データ・ロードの場合、Extract は、ソース・オブジェクトから現在のデータセットを直接抽出します。
- **変更同期 :** ソース・データと別のデータセットとの同期を維持するため、Extract は、初期同期の発生後にデータに加えられた変更 (通常はトランザクションによる挿入、更新および削除) を取得します。DDL の変更および順序も、使用中のデータベースのタイプでサポートされていれば抽出されます。

Extract では、データ変更を処理する際、次のいずれかの*データソース*からデータを取得します。

- *データベースのリカバリ・ログまたはトランザクション・ログ* (Oracle REDO ログや SQL/MX 監査証跡など)。ログからデータを取得する実際の方法は、データベース・タイプに応じて異なります。
- *サード・パーティの取得モジュール*。この方法では、データ変更およびメタデータを外部 API から Extract API に渡すための通信レイヤーが提供されます。データベース・ベンダーまたはサード・パーティ・ベンダーによって、データ変更を抽出して Extract に渡すコンポーネントが用意されます。

Extract は、同期対象として構成されているオブジェクトに加えられた変更をすべて取得します。Extract は、変更が含まれるトランザクションのコミット・レコードまたはロールバックを受信するまで、それらの変更を格納します。Extract がロールバックを受信すると、そのトランザクションのデータは破棄されます。Extract がコミットを受信すると、そのトランザクションのデータを証跡に送信して、ターゲット・システムに伝播します。トランザクションのログ・レコードは、すべて順次編成されたトランザクション単位として証跡に書き込まれます。この設計によって、処理速度とデータ整合性の両方が保証されます。

注意 Extract では、Extract 構成に存在しないオブジェクトに対する操作は無視されます。この動作は、Extract 構成に存在するオブジェクトに対する操作が同じトランザクションに含まれている場合でも同様です。

複数の Extract プロセスを異なるオブジェクトに対して同時に動作させることができます。たとえば、あるプロセスでは継続的にトランザクション・データ変更を抽出して意志決定支援データベースに送信し、別のプロセスでは定期レポート作成のためにバッチ抽出を実行できます。または、データベースの規模が大きい場合、ターゲットの待機時間を最小化するために、2 つの Extract プロセスで抽出を行い、(2 つの証跡を使用する) 2 つの Replicat プロセスにパラレルに送信することもできます。異なるプロセスどうしを区別するには、それぞれにグループ名を割り当てます (18 ページの「グループの概要」を参照)。

データ・ポンプの概要

データ・ポンプは、ソース Oracle GoldenGate 構成内のセカンダリ Extract グループです。データ・ポンプを使用しない場合、Extract は、データをターゲットのリモート証跡に送信する必要があります。一方、データ・ポンプが含まれる標準的な構成では、プライマリ Extract グループがソース・システムの証跡に書き込みます。データ・ポンプは、この証跡を読み取り、データをネットワーク経由でターゲットのリモート証跡に送信します。データ・ポンプによって、記憶域の柔軟性が向上すると同時に、プライマリ Extract プロセスが TCP/IP アクティビティから分離されます。

データ・ポンプは、プライマリ Extract グループと同様に、オンライン用またはバッチ処理用として構成できます。データのフィルタリング、マッピングおよび変換を実行できますが、データを操作せずにそのままの状態で単純に転送する*パススルー・モード*で構成することも可能です。パススルー・モードでは、オブジェクト定義を参照するすべての機能が回避されるため、データ・ポンプのスループットが向上します。

ほとんどのビジネス環境で、データ・ポンプを使用する必要があります。データ・ポンプを使用する理由として、次のことがあげられます。

- **ネットワークおよびターゲットの障害に対する保護 :** ターゲット・システムに証跡のみが存在する Oracle GoldenGate の基本構成では、Extract が継続的にメモリーに抽出するデータの格納場所がソース・システム上に存在しません。ネットワークまたはターゲット・システムが使用できなくなると、このような Extract はメモリーを使い果して異常終了する可能性があります。これに対し、ソース・システムに証跡とデータ・ポンプがあれば、取得データをディスクに移動して、プライマリ Extract の異常終了を防ぐことができます。接続が回復されると、データ・ポンプは、ソース証跡からデータを取得して 1 つ以上のターゲット・システムに送信します。
- **データのフィルタリングまたは変換の複数フェーズによる実装。**複雑なフィルタリング構成またはデータ変換構成を使用する場合、データ・ポンプを構成して、最初の変換をソース・システムまたはターゲット・システムのいずれかで (あるいは中間システムで) 実行し、別のデータ・ポンプまたは Replicat グループを使用して 2 番目の変換を実行できます。
- **多くのソースから中央ターゲットへのデータの統合。**複数のソース・データベースと中央ターゲット・データベースとを同期する場合、抽出したデータを各ソース・システムに格納し、それらの各システムでデータ・ポンプを使用してターゲット・システムの証跡にデータを送信できます。記憶域の負荷がソース・システムとターゲット・システムで分割されるため、複数のソースから送信されるデータに対応するためにターゲット・システムに大量の領域を用意する必要がなくなります。
- **1 つのソースと複数のターゲットの同期。**複数のターゲット・システムにデータを送信する場合、ソース・システムで各ターゲット用のデータ・ポンプを構成できます。いずれかのターゲットに対するネットワーク接続が切断されても、他のターゲットにデータを送信できます。

データ・ポンプを使用できない要件がある場合は、データ・ポンプなしでも Oracle GoldenGate を構成できます。Oracle GoldenGate では、様々な構成がサポートされます。このガイドの構成に関する章を参照して、現在の環境に最適な構成を見つけてください。

Replicat の概要

Replicat プロセスは、ターゲット・システムで実行されます。Replicat 構成で指定されている抽出済のデータ変更および DDL 変更 (サポートされる場合) を読み取り、ターゲット・データベースにレプリケートします。Replicat は次のいずれかの方法で構成できます。

- **初期ロード :** 初期データ・ロードの場合、Replicat は、データをターゲット・オブジェクトに適用するか、高速なバルク・ロード・ユーティリティにルーティングします。
- **変更同期 :** 同期状態を維持するため、Replicat は、データベース・タイプに応じてネイティブ・データベース・インタフェースまたは ODBC を使用して、抽出済のデータ変更をターゲット・オブジェクトに適用します。レプリケートされた DDL および順序も、使用中のデータベースのタイプでサポートされていれば適用されます。データ整合性を維持するため、Replicat は、レプリケートされた変更を、それらがソース・データベースにコミットされた順序で適用します。

複数の Replicat プロセスを複数の Extract プロセスとともにパラレルに使用して、スループットを向上できます。プロセスのセットごとに異なるオブジェクトを処理します。プロセスどうしを区別するには、それぞれにグループ名を割り当てます (18 ページの「グループの概要」を参照)。

Replicat は、データをターゲット・データベースに適用する前に一定の時間待機するよう遅延させることができます。遅延が推奨される場合として、たとえば、間違った SQL の伝播を防ぐ場合、異なるタイムゾーンにわたるデータの受信を制御する場合、または他の計画済イベントの発生に備えて時間を考慮する場合があげられます。遅延の長さは、DEFERAPPLYINTERVAL パラメータで制御します。

証跡の概要

データベース変更の継続的な抽出およびレプリケーションをサポートするために、Oracle GoldenGate は、取得した変更をディスク上の*証跡*と呼ばれる一連のファイルに一時的に格納します。証跡は、Oracle GoldenGate の構成方法に応じて、ソース・システム、ターゲット・システム、または中間システムに配置できます。証跡は、ローカル・システムでは*抽出証跡* (または*ローカル証跡*) と呼ばれます。リモート・システムでは、*リモート証跡*と呼ばれます。

Oracle GoldenGate では、記憶域として証跡を使用することで、データの正確性とフォルト・トレランスをサポートします (16 ページの「チェックポイントの概要」を参照)。また、証跡の使用により、抽出アクティビティとレプリケーション・アクティビティを相互に独立して実行できます。これらのプロセスが分離されることで、データを配信する方法の選択肢が広がります。たとえば、変更を継続的に抽出してレプリケートするかわりに、変更を継続的に抽出しながら、ターゲット・アプリケーションの必要に応じて後からいつでもターゲットにレプリケートできるように、それらの変更を証跡に格納することができます。

証跡に書き込むプロセスと証跡を読み取るプロセス

プライマリ Extract プロセスは、証跡に書込みを行います。1 つの Extract プロセスのみが、証跡に書き込むことができます。

証跡を読み取るプロセスには、次のものがあります。

- データ・ポンプ Extract: 後続の処理のために必要に応じてローカル証跡からデータを抽出し、それをターゲット・システムに転送するか、Oracle GoldenGate 構成内の次の Oracle GoldenGate プロセス・ダウンストリームに転送します。
- Replicat: 証跡を読み取って変更データをターゲット・データベースに適用します。

証跡のメンテナンス

証跡ファイルは、処理中に必要に応じて作成され、自動的にエージングされるので、ファイル・メンテナンスのために処理を中断する必要はありません。デフォルトでは、証跡は、Oracle GoldenGate ディレクトリの dirdat サブディレクトリに格納されます。

デフォルトでは、証跡の各ファイルのサイズは 10MB です。証跡のすべてのファイル名は、ユーザーが証跡の作成時に指定した同じ 2 つの文字で始まります。ファイルを作成すると、それぞれの名前に一意の 6 桁 (000000 から 999999) のシリアル (順序) 番号が追加されます (たとえば、\ggs\dirdat\tr000001 のようになります)。証跡の順序番号が 999999 に達すると、再度 000000 から番号付けが始まります。

異なるオブジェクトまたはアプリケーションからデータを分離するために、複数の証跡を作成できます。TABLE または SEQUENCE パラメータで指定したオブジェクトを、Extract パラメータ・ファイルの EXTTRAIL または RMTTRAIL パラメータで指定した証跡にリンクします。古くなった証跡ファイルは、Manager パラメータの PURGEOLDEXTRACTS を使用して消去できます。

プロセスによる証跡への書込み方法

スループットを最大化し、システムの I/O 負荷を最小化するため、抽出データの証跡に対する入出力は、サイズの大きいブロック単位で行われます。トランザクション順序は保持されます。デフォルトでは、Oracle GoldenGate は、証跡にデータを*正規形式* (異機種データベース間で高速かつ正確にデータを交換できる独自仕様の形式) で書き込みます。ただし、データの書込みは、異なるアプリケーションと互換性のある他の形式で行うことも可能です。

デフォルトでは、Extract は*追加モード*で動作します。このモードでは、プロセス障害が発生すると証跡にリカバリ・マーカーが書き込まれ、Extract は、すべての古いデータの履歴をリカバリ目的で保持するために、ファイルにリカバリ・データを追加します。

追加モードでは、Extract の初期化によって、起動時に証跡に書き込まれた最後の完全なトランザクションの ID が判別されます。この情報に基づいて、Extract は、そのトランザクションのコミット・レコードがデータソースで見つかると、リカバリを停止します。その後、Extract は、抽出要件を満たす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、新しいデータの証跡への追加を始めます。データ・ポンプまたは Replicat は、そのリカバリ・ポイントから読取りを再開します。

*上書きモード*は、リリース 10.0 より前のリリースの Oracle GoldenGate で使用されていた別のバージョンの Extract リカバリです。これらのリリースでは、Extract は、新規データを追加するかわりに、最後の書込みチェックポイント位置の後にある証跡の既存のトランザクション・データを上書きします。書き込まれる最初のトランザクションは、データソースの最後の読取りチェックポイント位置の後にある、抽出要件を満たす最初のトランザクションです。

ターゲットの Oracle GoldenGate のリリースがリリース 10 より古い場合、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。この動作は、RECOVERYOPTIONS パラメータを使用して手動で制御できます。

証跡の形式

Oracle GoldenGate リリース 10.0 以上では、証跡の各ファイルの先頭部分に、*ファイル・ヘッダー・レコード*が格納されています。ファイル・ヘッダーには、証跡ファイル自体に関する情報が含まれます。以前のリリースの Oracle GoldenGate には、このヘッダーは含まれません。

証跡ファイルの各データ・レコードには、データ領域の他にヘッダー領域も含まれます。レコード・ヘッダーには、トランザクション環境に関する情報が含まれ、データ領域には、抽出された実際のデータ値が含まれます。証跡レコードの形式の詳細は、付録 1 を参照してください。

ファイルのバージョニング

Oracle GoldenGate プロセスはすべて独立しており、異なる Oracle GoldenGate リリースのプロセスが混在できるため、各証跡ファイルまたは抽出ファイルではファイル・ヘッダーにバージョンが格納されています。デフォルトでは、証跡のバージョンは、そのファイルを作成したプロセスの現行バージョンです。証跡のバージョンを設定するには、EXTTRAIL、EXTFILE、RMTTRAIL、RMTFILE のいずれかのパラメータの FORMAT オプションを使用します。

Oracle GoldenGate の異なるプロセス・バージョン間でファイルの上位互換性または下位互換性を保証するため、標準化されたトークン形式でファイル・ヘッダー・フィールドが書き込まれます。プロセスの新規バージョンによって作成される新しいトークンは、古いバージョンでは無視されるため、下位互換性が保持されます。同様に、Oracle GoldenGate の新しいリリースでは、古いトークンがサポートされます。また、新しいプロセス・バージョンによってあるトークンが非推奨になっても、そのトークンにはデフォルト値が割り当てられるため、古いバージョンも引き続き正しく動作します。ファイル・バージョンを指定するトークンは、COMPATIBILITY です。このトークンは、ログダンプ・ユーティリティで表示することや、@GETENV 関数の GGFILEHEADER オプションを使用して取得することができます。

証跡または抽出ファイルのバージョンは、そのファイルを*読み取る*プロセスのバージョン以下である必要があります。それ以外の場合、プロセスは異常終了します。また、データ・ポンプの出力証跡またはファイルは、Oracle GoldenGate によって強制的に入力証跡またはファイルと同じバージョンに設定されます。再起動時に、Extract は、各ファイルのバージョンがただ 1 つになるように証跡を 1 つの新規ファイルにまとめます (ファイルが空ではない場合)。

抽出ファイルの概要

初期ロードまたはトランザクション変更を同期するバッチ実行など、1 回かぎりの実行を処理する場合 (18 ページを参照)、Oracle GoldenGate は、抽出した変更を証跡ではなく*抽出ファイル*に格納します。通常、抽出ファイルは単一のファイルですが、オペレーティング・システムのファイル・サイズ制限を考慮して複数のファイルにロールオーバーするように構成することもできます。この点で、抽出ファイルは証跡と似ていますが、チェックポイントは記録されません。実行中に 1 つ以上のファイルが自動的に作成されます。証跡に適用されるバージョニング機能と同じ機能が、抽出ファイルにも適用されます。

チェックポイントの概要

*チェックポイント*は、プロセスの現在の読取り位置と書込み位置をリカバリ目的でディスクに格納します。これらのチェックポイントは、同期対象としてマークされたデータ変更が、実際に Extract により抽出されて Replicat によりレプリケートされることを保証し、重複処理を防止します。チェックポイントによって、システム、ネットワークまたは Oracle GoldenGate プロセスを再起動した場合のデータ損失が防止され、フォルト・トレランスが実現します。複雑な同期構成では、チェックポイントにより、複数の Extract プロセスまたは Replicat プロセスを使用して同じ証跡セットから読取りを行うことができます。

チェックポイントは、メッセージがネットワーク内で失われないように、プロセス間の確認応答と連携して動作します。Oracle GoldenGate には、独自仕様のメッセージ配信保証テクノロジがあります。

Extract は、データソースおよび証跡内にその位置を示すチェックポイントを作成します。Replicat は、証跡内にその位置を示すチェックポイントを作成します。

チェックポイント・システムは、継続的に実行される Extract プロセスと Replicat プロセスによって使用されますが、バッチ・モード (18 ページを参照) で実行される Extract プロセスと Replicat プロセスでは必要ありません。バッチ・プロセスは、その開始ポイントから再実行できますが、継続処理では、計画済または計画外の中断に対してチェックポイントで対応する必要があります。

Replicat は、チェックポイントをターゲット・データベース内のチェックポイント表に格納し、トランザクションのコミットと証跡ファイル内の位置とを対応付けます。これにより、Replicat プロセスまたはデータベース・プロセスに障害が発生した場合でもトランザクションは一度しか適用されないことが保証されます。Replicat は、レポート用に、ディスク上の Oracle GoldenGate ディレクトリにある dirchk サブディレクトリにもチェックポイント・ファイルを持っています。オプションで Replicat がこのファイルを唯一のチェックポイント・ストアとして使用し、チェックポイント表はまったく使用しないように構成できます。ただし、このモードでは、Replicat によって適用されたとみなされているトランザクションがエラーによってロールバックまたはロールフォワードされると、ファイルのチェックポイントがデータベースのリカバリ後に適用されたものと一致しない場合があります。チェックポイント表ではリカバリ後の一貫性が保証されます。

Manager の概要

Manager は、Oracle GoldenGate の制御プロセスです。Manager は、Extract または Replicat が起動される前に、Oracle GoldenGate 構成内の各システムで稼働している必要があります。Manager は、リソース管理機能を実行するために、これらのプロセスの実行中は稼働し続ける必要があります。Manager は次の機能を実行します。

- Oracle GoldenGate プロセスの監視および再起動。
- しきい値レポートの発行 (スループットが低下した場合や、同期の待機時間が増加した場合など)。
- 証跡ファイルおよびログの管理。
- データ記憶域の割当て。
- エラーおよびイベントのレポート。
- ユーザー・インタフェースからのリクエストの受信およびルーティング。

1 つの Manager プロセスで、複数の Extract または Replicat プロセスを制御できます。Windows システムでは、Manager はサービスとして実行できます。Manager プロセスの詳細は、第 2 章を参照してください。

Collector の概要

Collector は、ターゲット・システム上でバックグラウンドで実行されるプロセスです。TCP/IP ネットワークを通じて送信された抽出済のデータベース変更を受信し、証跡または抽出ファイルに書き込みます。通常、Manager は、ネットワーク接続が必要な場合は自動的に Collector を起動します。Manager が Collector を起動する場合、そのプロセスは*動的* Collector と呼ばれ、通常は Oracle GoldenGate ユーザーと通信することはありません。ただし、Collector は手動で実行できます。これは*静的* Collector とも呼ばれます。Oracle GoldenGate 構成によっては、Collector プロセスが使用されないこともあります。

動的 Collector は 1 つの Extract プロセスからしか情報を受信できないので、これを使用する場合は Extract ごとに動的 Collector が必要になります。静的 Collector を使用する場合は、複数の Extract プロセスで 1 つの Collector を共有できます。ただし、1 対 1 の比率が最適です。Collector プロセスは、関連する Extract プロセスが終了すると終了します。

デフォルトでは、Extract がソース・システムからターゲットの Collector に対して TCP/IP 接続を開始しますが、Collector がターゲットから接続を開始するように Oracle GoldenGate を構成することもできます。ターゲットから接続を開始する必要があるのは、たとえば、ターゲットが信頼できるネットワーク・ゾーンにある一方で、ソースのネットワーク・ゾーンの信頼度がそれより低い場合です。

処理方法の概要

Oracle GoldenGate は、次の目的で構成できます。

- あるデータベースから選択したデータ・レコードの静的抽出を行い、別のデータベースにそのレコードをロードする場合。
- ソースとターゲットのデータ一貫性を維持するために、選択したトランザクション・データ変更と DDL 変更 (サポートされるデータベースの場合) をオンラインまたはバッチ処理で抽出およびレプリケートする場合。
- データベースから抽出を行い、データベース以外の場所にあるファイルにレプリケートする場合。

これらの目的に対応するため、Oracle GoldenGate では、次の処理モードがサポートされます。

- *オンライン・プロセス*は、ユーザーが停止するまで実行されます。オンライン・プロセスは、証跡にリカバリ・チェックポイントを保持するため、処理を中断しても再開することができます。オンライン・プロセスを使用して、トランザクション変更および DDL 変更 (サポートされる場合) を継続的に抽出してレプリケートできます。
- *バッチ実行* (*特別実行*) プロセスは、既知の開始ポイントと終了ポイントの範囲内で生成されたデータベース変更を抽出またはレプリケートします。特別実行では、Oracle GoldenGate でチェックポイントが保持されません。プロセスに障害が発生した場合は、同じ開始ポイントと終了ポイントを使用してジョブを最初から開始できます。特別実行を使用して、データベース変更をバッチ処理したり (継続的にではなく 1 日に 1 回ソースとターゲットのオブジェクトを同期する場合など)、初期データ・ロードを実行したりできます。
- *タスク*は、バッチ実行プロセスの特別なタイプであり、一部の初期ロード方法で使用されます。タスクは、Extract が TCP/IP 経由で直接 Replicat と通信する構成です。Collector プロセスも、証跡またはファイルの一時ディスク記憶域も使用されません。

グループの概要

システムの複数の Extract プロセスまたは Replicat プロセスを区別するには、処理*グループ*を定義します。たとえば、異なるデータセットをパラレルにレプリケートするには、2 つの Replicat グループを作成します。

処理グループは、プロセス (Extract または Replicat)、パラメータ・ファイル、チェックポイント・ファイル、およびそのプロセスに関連する他のファイルで構成されます。Replicat の場合、グループには関連するチェックポイント表も含まれます。

グループを定義するには、Oracle GoldenGate のコマンド・インタフェースである GGSCI で、ADD EXTRACT コマンドおよび ADD REPLICAT コマンドを使用します。グループ名は次のように指定できます。

表 2　使用可能なグループ名

- 最大8つのASCII文字を使用できます(アンダースコア(_)などの英数字以外の文字にも対応)。任意のASCII 文字を使用できます (ただし、オペレーティング・システムでファイル名への使用が許可されている文字のみ)。この理由は、グループがその関連するチェックポイント・ファイルで識別されるためです。
- 次の ASCII 文字はファイル名には使用できません。

  { \ / : * ? " < > | }
- HP UX、Linux および Solaris では、コロン (:) またはアスタリスク (*) を使用してファイル名を指定できますが、推奨はされません。
- 一般的に、Oracle GoldenGate 内では、グループ名の大 / 小文字は区別されません。たとえば、finance、Finance および FINANCE は、すべて同じであるとみなされます。ただし、Linux では、グループ名 (および ADD コマンドで明示的に定義される場合はそのパラメータ・ファイル名) はすべて大文字またはすべて小文字である必要があります。グループ名およびパラメータ・ファイル名に大文字と小文字が混在していると、プロセスの起動時にエラーが発生します。
- 単一の語を使用してください。
- グループ名に port という語は使用しないでください。ただし、グループ名の一部として port という文字列を使用することは可能です。
- グループ名の末尾に数値を使用しないでください (fin1 や fin10 など)。グループ名の先頭であれば、数値を使用しても構いません (1_fin や 1fin など)。

グループに関連するすべてのファイルおよびチェックポイントでは、グループ自体に割り当てられた名前を共有します。処理を制御または表示するコマンドを発行する場合、常に単一のグループ名または (ワイルドカードを使用して) 複数のグループ名を指定します。

コミット順序番号 (CSN) の概要

Oracle GoldenGate で作業する場合、状況に応じて*コミット順序番号* (CSN) を参照する必要があります。CSN は、トランザクション・ログで Extract の位置を指定する場合、証跡で Replicat の位置を再指定する場合、またはその他の目的で必要になることがあります。CSN は、複数の変換関数によって戻され、レポートおよび一部の GGSCI 出力に含まれます。

CSN は、トランザクション一貫性とデータ整合性を維持する目的でトランザクションを識別するために Oracle GoldenGate が作成する識別子です。CSN によって、トランザクションがデータベースにコミットされた特定の時点が一意に識別されます。

各種のデータベース管理システムでは、各トランザクションの完了時になんらかの一意のシリアル番号が独自に生成され、そのトランザクションが一意に識別されます。CSN は、この同じ識別情報を取得して、それを内部的に一連のバイト列として表現します。ただし、CSN は、プラットフォームに依存しない方法で処理されます。それぞれが同じログ・ストリームのトランザクション・コミット・レコードにバインドされた 2 つの任意の CSN 番号を比較することで、2 つのトランザクションが完了した順序が正確に示されます。

CSN 値は、トランザクションの開始を識別する証跡レコードにトークンとして格納されます。この値は、@GETENV 列変換関数を使用して取得することや、ログダンプ・ユーティリティを使用して表示することができます。

Extract は、証跡ファイルやチェックポイント・ファイルなどの外部記憶域に正規形の CSN を書き込みます。そこでは、CSN はバイト列の 16 進文字列として表現されます。正規形では、最初の 2 バイトがデータベース・プラットフォームを表し、残りの文字列が実際の一意識別子を表します。

CSN は、レポート出力、エラー・メッセージおよびコマンド入出力 (適切な場合) にも、ネイティブ文字エンコーディングを使用した判読可能な表示形式で出力されます。データベース・タイプはこの形式に含まれていませんが、識別子とは別に指定できます。

表 3　データベースごとの Oracle GoldenGate の CSN 値[1]

データベース	CSN 値

c-tree	`<log number>.<byte offset>`
	変数 :
	◆ <log number> は、先行する 0 (ゼロ) が埋め込まれた、10 桁の 10 進 c-tree ログ・ファイル番号です。
	◆ <byte offset> は、先行する 0 (ゼロ) が埋め込まれた、ファイル (0 ベース) の先頭部分から 10 桁の 10 進相対バイト位置です。
	例 :
	`0000000068.0000004682`

DB2 for i	DB2 for i には CSN はありません。このデータベースでは、Oracle GoldenGate による抽出 (取得) がサポートされないためです。

DB2 LUW	`<LSN>`
	変数 :
	◆ <LSN> は、可変長の 10 進ベースの DB2 ログ順序番号です。
	例 :
	`1234567890`

DB2 z/OS	`<RBA>`
	変数 :
	◆ <RBA> は、トランザクション・ログ内のコミット・レコードにおける 6 バイトの相対バイト・アドレスです。
	例 :
	`1274565892`

MySQL	`<LogNum>:<LogPosition>`
	変数 :
	◆ <LogNum> は、識別されるトランザクションの START TRANSACTION レコードが含まれるログ・ファイルの名前です。
	◆ <LogPosition> は、そのレコードのイベント・オフセット値です。イベント・オフセット値は、ログ・レコードのレコード・ヘッダー・セクションに格納されます。
	たとえば、ログ番号が 12 で、ログ位置が 121 の場合、CSN は次のようになります。
	`000012:000000000000121`

表 3　データベースごとの Oracle GoldenGate の CSN 値[1] (続き)

データベース	CSN 値

Oracle	`<system change number>`
	変数 :
	◆ <system change number> は、Oracle の SCN 値です。
	例 :
	`6488359`

SQL/MX	`<sequence number>.<RBA>`
	変数 :
	◆ <sequence number> は、先行する 0 (ゼロ) が埋め込まれた、6 桁の 10 進 NonStop TMF 監査証跡順序番号です。
	◆ <RBA> は、先行する 0 (ゼロ) が埋め込まれた、そのファイル内の 10 桁の 10 進相対バイト・アドレスです。
	これらの組合せにより、TMF マスター監査証跡 (MAT) 内の位置が指定されます。
	例 :
	`000042.0000068242`

SQL Server	データベースが値を戻す方法に応じて、次のいずれかになります。
	◆ 先行する 0 (ゼロ) が埋め込まれた、0X 接頭辞付きのコロン区切りの 16 進文字列 (8:8:4)
	◆ 先行する 0 (ゼロ) が埋め込まれた、コロン区切りの 10 進文字列 (10:10:5)
	◆ 先行する 0 (ゼロ) のない、0X 接頭辞付きのコロン区切りの 16 進文字列
	◆ 先行する 0 (ゼロ) のない、コロン区切りの 10 進文字列
	◆ 10 進文字列
	変数 :
	◆ 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。
	例 :
	`0X00000d7e:0000036b:01bd`
	`0000003454:0000000875:00445`
	`0Xd7e:36b:1bd`
	`3454:875:445`
	`3454000000087500445`

表 3　データベースごとの Oracle GoldenGate の CSN 値[1] (続き)

データベース	CSN 値

Sybase	`<time_high>.<time_low>.<page>.<row>`
	変数 :
	◆ <time_high> および <time_low> は、ログ・ページのインスタンス ID を表します。これは、各データベース・ログ・ページのヘッダーに格納されます。<time_high> は 2 バイトで、<time_low> は 4 バイトです (それぞれ先行する 0 (ゼロ) が埋め込まれます)。
	◆ <page> は、0 (ゼロ) が埋め込まれたデータベース論理ページ番号です。
	◆ <row> は、0 (ゼロ) が埋め込まれた行番号です。
	これらの構成要素が組み合されて、ログ・ストリーム内で一意の位置を表します。timestamp-high の 2 バイト整数の有効範囲は、0 から 65535 です。timestamp-low の 4 バイト整数の場合、0 から 4294967295 です。
	例 :
	`00001.0000067330.0000013478.00026`

Teradata	`<sequence ID>`
	変数 :
	◆ <sequence ID> は、固定長の出力可能な汎用順序 ID です。
	例 :
	`0x08000000000000000D700000021`

TimesTen	TimesTen には CSN はありません。このデータベースでは、Oracle GoldenGate による抽出 (取得) がサポートされないためです。

[1] Oracle、DB2 LUW および DB2 z/OS 以外のすべてのデータベース・プラットフォームは、固定長の CSN を持ちます。この CSN は、固定長にするために必要に応じて先行する 0 (ゼロ) が埋め込まれます。複数のフィールドを含む CSN は、各フィールド内で埋込みが行われることがあります (Sybase CSN など)。

第 2 章

Manager プロセスの構成

Manager プロセスの概要

Oracle GoldenGate を構成して実行するには、ソース・システムとターゲット・システムで Manager プロセスが稼働している必要があります。Manager プロセスは次の機能を実行します。

- Oracle GoldenGate プロセスの起動
- 動的プロセスの起動
- 証跡管理の実行
- イベント、エラーおよびしきい値レポートの作成

Oracle GoldenGate のインストール環境ごとに 1 つの Manager があります。1 つの Manager で、Oracle GoldenGate の複数の抽出プロセスおよびレプリケーション・プロセスに対応できます。

Manager の構成

Manager を構成するには、次の手順に従ってパラメータ・ファイルを作成します。UNIX クラスタに Oracle GoldenGate をインストールしている場合、ベンダーのドキュメントに従ってクラスタ・アプリケーション内で Oracle GoldenGate Manager プロセスを構成し、Oracle GoldenGate が他のアプリケーションに適切にフェイルオーバーされるようにします。

Manager を構成する手順

1. Oracle GoldenGate ディレクトリから ggsci プログラムを実行し、Oracle GoldenGate ソフトウェア・コマンド・インタフェース (一般に GGSCI と呼ばれる) を起動します。
2. GGSCI で、次のコマンドを発行して Manager のパラメータ・ファイルを編集します。

```
EDIT PARAMS MGR
```

3. 次のパラメータを追加して Manager のポート番号を指定します。

```
PORT <port_number>
```

PORT では、ローカル・システム上で Manager が実行されるポート番号を定義します。次のガイドラインに従ってください。

- デフォルトのポート番号は 7809 です。
- デフォルトのポート番号または別のポート番号を指定する必要があります。
- システム上の Manager インスタンスごとに異なるポート番号を使用する必要があります。
- これは、予約されていない制限なしのポートである必要があります。GGSCI では、このポートを使用して、Manager にプロセスを起動するようリクエストします。Extract プロセスでは、このポー

トを使用して、Manager にリモート Collector プロセスまたは初期ロード Replicat プロセスを起動するようリクエストします。

- Manager の必須パラメータは PORT のみです。

  注意 **ポート番号は、Extract パラメータの RMTHOST の MGRPORT 引数でも指定する必要があります。**

4. 『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』に記載されているオプションの Manager パラメータを任意に入力し、ファイルを保存して閉じます。

推奨パラメータ

次のパラメータは Manager プロセスではオプションですが、使用することをお薦めします。詳細と構文は、『*Windows and UNIX リファレンス・ガイド*』を参照してください。

- USERID: Oracle GoldenGate の DDL サポートを使用する場合は必須です。これを使用して Manager のユーザーとパスワードを指定します。
- DYNAMICPORTLIST: ソース・システムとターゲット・システム間の動的 TCP/IP 通信用に、予約されていない制限なしのポート (最大 256 個) を指定する場合に使用します。Collector、Replicat および GGSCI の各プロセスでは、これらのポートが使用されます。DYNAMICPORTLIST が指定されない場合、Manager は、Collector をポート 7840 で起動しようと試みます。7840 を使用できない場合、Manager は、使用できるポートが見つかるまで 1 ずつ増分していきます。
- DYNAMICPORTREASSIGNDELAY: Manager が以前に割り当てたポートを割り当てるまでに待機する期間を制御します。
- AUTOSTART: Manager の起動時に Extract プロセスおよび Replicat プロセスを起動します。これは、たとえば、Manager が起動ルーチンの一部である場合に、システムの起動とともに即座に Oracle GoldenGate アクティビティを開始する場合に便利です。同じパラメータ・ファイルで複数の AUTOSTART 文を使用できます。
- AUTOSTART: 異常終了の後に Extract プロセスおよび Replicat プロセスを再起動します。
- PURGEOLDEXTRACTS: Oracle GoldenGate による処理が終了した証跡ファイルを消去します。PURGEOLDEXTRACTS を使用しない場合、消去は実行されないので、証跡ファイルによって大量のディスク領域が消費される可能性があります。Extract または Replicat 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。

  注意 **PURGEOLDEXTRACTS を使用する場合、Oracle GoldenGate 以外のユーザーまたはプログラムによる証跡ファイルの削除を許可しないでください。PURGEOLDEXTRACTS が正しく動作しない可能性があります。**

Manager の起動

Manager は、他の Oracle GoldenGate プロセスが起動する前に実行されている必要があります。Manager は次の方法を使用して起動できます。

- サポートされる任意のオペレーティング・システムのコマンドライン。
- GGSCI
- Windows システムの「サービス」アプレット (Manager をサービスとしてインストールした場合)。Windows のドキュメントを参照するか、システム管理者に連絡してください。
- クラスタ・アドミニストレータ・ツール (システムが Windows クラスタの一部である場合)。この方法は、Manager リソースをオンラインにする場合に推奨されます。クラスタのドキュメントを参照するか、システム管理者に連絡してください。

- UNIX または Linux クラスタのクラスタ・ソフトウェア。クラスタ・ベンダーから提供されているドキュメントを参照して、Manager をクラスタから起動するか、GGSCI またはオペレーティング・システムのコマンドラインを使用して起動するかを確認してください。

コマンドラインから Manager を起動する手順

オペレーティング・システムのコマンド・シェルから Manager を実行するには、次の構文を使用します。

```
mgr paramfile <param file> [reportfile <report file>]
```

reportfile 引数はオプションです。この引数は、Oracle GoldenGate のインストール場所にあるデフォルトの dirrpt ディレクトリ以外の場所に Manager プロセス・レポートを格納する場合に使用できます。

Manager を GGSCI から起動する手順

1. Oracle GoldenGate ディレクトリから GGSCI を実行します。
2. GGSCI で、次のコマンドを発行します。

```
START MANAGER
```

Windows システムでは、BOOTDELAYMINUTES パラメータを使用して、システム起動後に Manager が処理アクティビティを開始するまでの遅延時間を指定できます。

注意 Windows Server 2008 でユーザー・アカウント制御を有効にした状態でコマンドラインまたは GGSCI から Manager を起動すると、プログラム実行の許可または拒否を求める UAC プロンプトが表示されます。

Manager の停止

Manager は、ユーザーが停止するまで無制限に実行されます。通常、同期アクティビティの実行中は、常に Manager が実行されている必要があります。Manager は、重要な監視機能およびメンテナンス機能を実行するため、Manager が稼働していないとプロセスを起動できません。

Manager を停止する手順

- UNIX、Linux、および USS を使用する z/OS では、GGSCI の STOP MANAGER コマンドを使用して Manager を停止する必要があります。

```
STOP MANAGER [!]
```

  条件： ！は、ユーザー確認なしで Manager を停止します。

- Windows では、「サービス」アプレットから Manager を停止できます (Manager をサービスとしてインストールしている場合)。Windows のドキュメントを参照するか、システム管理者に連絡してください。
- Windows クラスタで Manager を停止するには、必ずクラスタ・アドミニストレータを使用して Manager リソースをオフラインにする必要があります。GGSCI インタフェースから Manager サービスを停止しようとすると、クラスタ・モニターがそれをリソース障害と解釈し、リソースを再度オンラインにしようとします。停止リクエストが GGSCI インタフェースから繰り返し発行され、Manager クラスタ・リソースの再起動しきい値を超えると、クラスタ・モニターが Manager リソースを障害状態としてマークします。
- UNIX または Linux クラスタの場合、クラスタ・ベンダーから提供されているドキュメントを参照して、Manager をクラスタから停止する必要があるか、GGSCI を使用して停止する必要があるかを確認してください。

第 3 章

Oracle GoldenGate のスタート・ガイド

ユーザー・インタフェースの実行

処理を制御および監視するには、次のいずれかの方法を使用します。

- GGSCI(Oracle GoldenGate ソフトウェア・コマンド・インタフェース)
- Oracle GoldenGate Director
- バッチ・スクリプトおよびシェル・スクリプト

GGSCI コマンドライン・インタフェースの使用

GGSCI は、Oracle GoldenGate のコマンドライン・インタフェースです。GGSCI を使用して、Oracle GoldenGate を構成、制御および監視する全種類のコマンドを発行できます。

GGSCI を起動する手順

1. Oracle GoldenGate がインストールされているディレクトリに移動します。
2. ggsci 実行可能ファイルを実行します。

Oracle GoldenGate コマンドの詳細は、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』を参照してください。

コマンド引数でのワイルドカードの使用

一部の Oracle GoldenGate コマンドでワイルドカードを使用して、複数の Extract グループおよび Replicat グループを 1 つの単位として制御できます。Oracle GoldenGate でサポートされるワイルドカード記号は、アスタリスク (*) です。アスタリスクは、任意の数の文字を表します。たとえば、名前に文字 X を含むすべての Extract グループを起動するには、次のコマンドを発行します。

```
START EXTRACT *X*
```

コマンド履歴の使用

次のツールを使用すると、複数のコマンドを簡単に実行できます。

- 以前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。
- 以前のコマンドを編集せずに再実行するには、! コマンドを使用します。
- 以前のコマンドを編集してから再実行するには、FC コマンドを使用します。

よく使用するコマンド・シーケンスの保存

よく使用する一連のコマンドは、OBEY ファイルおよび OBEY コマンドを使用して自動化できます。

OBEY を使用する手順

1. 1 行に 1 つのコマンドを含むテキスト・ファイルを作成して保存します。これがユーザーの OBEY ファイルになります。任意の名前を付けてください。OBEY ファイル内に他の OBEY ファイルをネストできます。
2. GGSCI を実行します。
3. (オプション) ネストした OBEY ファイルが含まれる OBEY ファイルを使用する場合、次のコマンドを発行します。このコマンドによって、GGSCI の現在のセッションでネストした OBEY ファイルを使用できるようになります。ネストした OBEY ファイルを使用する場合、常にこのコマンドが必要です。

   ```
   ALLOWNESTED
   ```

4. GGSCI で、次の構文を使用して OBEY ファイルを呼び出します。

   ```
   OBEY <file name>
   ```

   条件 : <file name> は、OBEY ファイルの相対名または完全修飾名です。

図 3 は、OBEY コマンドと組み合せて使用する OBEY コマンド・ファイルを示しています。この操作によって、Extract グループおよび Replicat グループが作成されて起動され、処理情報が取得されます。

図 3 OBEY コマンド・ファイル

```
ADD EXTRACT myext, TRANLOG, BEGIN now
START EXTRACT myext

ADD REPLICAT myrep, EXTTRAIL /ggs/dirdat/aa
START REPLICAT myrep

INFO EXTRACT myext, DETAIL
INFO REPLICAT myrep, DETAIL
```

UNIX バッチ・スクリプトおよびシェル・スクリプトの使用

UNIX システムでは、GGSCI を実行して入力ファイルを呼び出すことで、起動スクリプト、停止スクリプト、フェイルオーバー・スクリプトなどのスクリプトから Oracle GoldenGate コマンドを発行できます。次の構文を使用します。

```
ggsci < <input_file>
```

条件 : <input_file> は、コマンドが発行順に記載されたテキスト・ファイルであり、< 文字は、ファイルを GGSCI プログラムにパイプします。

注意 バッチ・ファイルから Manager プロセスを停止するには、必ず STOP MANAGER コマンドの最後に ! 引数を追加してください。そうしないと、GGSCI によってレスポンスを要求するプロンプトが発行され、処理がループに陥ります。

Oracle GoldenGate のパラメータ・ファイルの使用

ほとんどの Oracle GoldenGate 機能は、パラメータ・ファイルに指定されたパラメータによって制御されます。パラメータ・ファイルは、関連するプロセスによって読み取られる ASCII ファイルです。Oracle GoldenGate では、GLOBALS ファイルとランタイム・パラメータ・ファイルという 2 種類のパラメータ・ファイルが使用されます。

GLOBALS ファイルの概要

GLOBALS ファイルには、Oracle GoldenGate インスタンス全体に関連するパラメータが格納されます。これは、Extract などの特定のプロセスと関連付けられているランタイム・パラメータとは対照的です。GLOBALS ファイルのパラメータは、Oracle GoldenGate インスタンスのすべてのプロセスに適用されますが、特定のプロセスのパラメータによって上書きされることがあります。GLOBALS パラメータは、設定後に変更することはほとんどなく、その数もランタイム・パラメータに比べてかなり少なくなっています。

GLOBALS パラメータ・ファイルは、一部の環境でのみ必要とされますが、使用する場合は、GGSCI などの Oracle GoldenGate プロセスを起動する前にコマンド・シェルから作成しておく必要があります。GGSCI プログラムは、GLOBALS ファイルを読み取って、各パラメータを適切なプロセスに渡します。

GLOBALS ファイルを作成する手順

1. Oracle GoldenGate のインストール場所から、GGSCI を実行して次のコマンドを入力するか、ASCII テキスト・エディタでファイルを開きます。

   ```
   EDIT PARAMS ./GLOBALS
   ```

   注意 GLOBALS ファイルは Oracle GoldenGate インストール・ファイルのルートに位置している必要があるため、このコマンドの ./ 部分を必ず使用してください。

2. ファイルに GLOBALS パラメータを 1 行に 1 つずつ入力します。
3. ファイルを保存します。テキスト・エディタを使用した場合、Oracle GoldenGate のインストール・ディレクトリのルートに GLOBALS(大文字、ファイル拡張子なし) という名前でファイルを保存します。GGSCI で正しく作成すると、ファイルは自動的にこの形式で保存されます。このファイルは移動しないでください。
4. GGSCI を終了します。コマンドを発行する前に、または GLOBALS ファイルを参照するプロセスを起動する前に、新規 GGSCI セッションを開始する必要があります。

ランタイム・パラメータの概要

ランタイム・パラメータによって、次のような Oracle GoldenGate の同期の様々な機能を制御します。

- データの選択、マッピング、変換およびレプリケーション
- DDL および順序の選択、マッピングおよびレプリケーション (サポートされる場合)
- エラーの解決
- ロギング
- ステータスおよびエラーのレポート
- システム・リソースの使用方法
- 起動時および実行時の動作

1 つの Manager プロセスまたは 1 つの Extract(あるいは Replicat) グループに対してアクティブなパラメータ・ファイルを 1 つのみ指定できます。ただし、OBEY パラメータを使用することで、他のファイルのパラメータを使用できます。33 ページの「パラメータ・ファイルの作成の簡略化」を参照してください。

パラメータには、次のようにグローバル・パラメータ (GLOBALS パラメータとは異なります) とオブジェクト固有パラメータの 2 種類があります。

- グローバル・パラメータは、パラメータ・ファイルに指定されているすべてのデータベース・オブジェクトに適用されます。たとえば、プロセスの動作に影響するパラメータや、メモリー使用率などの機能に影響するパラメータがあります。図 4 および図 5 の USERID は、グローバル・パラメータの例です。ほとんどの場合、グローバル・パラメータは、ファイル内でデータベース・オブジェクトを指定するパラメータ (たとえば、図 4 および図 5 の TABLE 文や MAP 文) より前の任意の場所に指定できます。グローバル・パラメータは、ファイル内で 1 回のみリストされる必要があります。複数回リストされると、*最後*のインスタンスのみがアクティブになり、他のすべてのインスタンスは無視されます。
- オブジェクト固有パラメータでは、データベース・オブジェクトの異なるセットに対して異なる処理ルールを適用できます。図 5 の GETINSERTS および IGNOREINSERTS は、オブジェクト固有パラメータの例です。どちらのパラメータも、影響を受けるオブジェクトを指定する MAP 文の前にあります。オブジェクト固有パラメータは、ファイルにリストされている順序で有効になります。

次に、Extract および Replicat の基本パラメータ・ファイルの例を示します。

図 4 Extract のサンプル・パラメータ・ファイル

```
EXTRACT capt
USERID ggs, PASSWORD *********
DISCARDFILE /ggs/capt.dsc, PURGE
RMTHOST sysb, MGRPORT 7809
RMTTRAIL /ggs/dirdat/aa
TABLE fin.*;
TABLE sales.*;
```

図 5 Replicat のサンプル・パラメータ・ファイル

```
REPLICAT deliv
USERID ggs, PASSWORD ****SOURCEDEFS /ggs/dirdef/defs
DISCARDFILE /ggs/deliv.dsc, PURGE

GETINSERTS
MAP fin.account, TARGET fin.acctab,
COLMAP (account = acct,
balance = bal,
branch = branch);

MAP fin.teller, TARGET fin.telltab,
WHERE (branch = "NY");

IGNOREINSERTS
MAP fin.teller, TARGET fin.telltab,
WHERE (branch = "LA");
```

パラメータ・ファイルの作成

パラメータ・ファイルを作成するには、GGSCI ユーザー・インタフェース内で EDIT PARAMS コマンドを使用するか (推奨)、テキスト・エディタを直接使用します。GGSCI を使用する場合、標準のテキスト・エディタを使用しますが、パラメータ・ファイルは正しいファイル名で適切なディレクトリに自動的に保存されます。

EDIT PARAMS コマンドによって、GGSCI インタフェース内で次のテキスト・エディタが起動されます。

- Microsoft Windows システムの場合、メモ帳
- UNIX および Linux システムの場合、vi エディタ

> **注意** GGSCI インタフェースを通じてデフォルト・エディタを変更するには、SET EDITOR コマンドを使用します。

GGSCI でパラメータ・ファイルを作成する手順

1. Oracle GoldenGate がインストールされているディレクトリから GGSCI を実行します。
2. GGSCI で、次のコマンドを発行してデフォルトのテキスト・エディタを起動します。

   ```
   EDIT PARAMS <group name>
   ```

   条件 : <group name> は、mgr(Manager プロセスの場合) か、ファイルを作成する Extract グループまたは Replicat グループの名前です。Extract または Replicat のパラメータ・ファイルの名前は、プロセス・グループの名前と同じである必要があります。

   次のコマンドでは、extora という名前の Extract グループのパラメータ・ファイルを作成または編集します。

   ```
   EDIT PARAMS extora
   ```

   次のコマンドでは、Manager プロセスのパラメータ・ファイルを作成または編集します。

   ```
   EDIT PARAMS MGR
   ```

   > **注意** Linux では、グループおよびパラメータ・ファイルの名前はすべて大文字か小文字で指定する必要があります。ファイル名に大 / 小文字が混在している場合、プロセスの起動時にエラーが発生します。

3. エディタの編集機能を使用して、このファイルを説明するコメント行を必要な数だけ入力します (各コメント行の先頭には必ず 2 つのハイフン (--) を挿入してください)。
4. コメント以外の行に、Oracle GoldenGate のパラメータを入力します (パラメータ文ごとに新しい行を開始します)。

   Oracle GoldenGate のパラメータの構文は次のとおりです。

   ```
   <PARAMETER> <argument> [, <option>] [&]
   ```

   条件 :

   - <PARAMETER> は、パラメータの名前です。

- <argument> は、パラメータの必須引数です。一部のパラメータには引数がありますが、その他のパラメータにはありません。引数は、すべてカンマで区切ってください。次に例を示します。

  ```
  USERID ggs, PASSWORD AACAAAAAAAAAAAIALCKDZIRHOJBHOJUH, &
  ENCRYPTKEY superx128
  RMTHOST sysb, MGRPORT 8040
  RMTTRAIL /home/ggs/dirdat/c1, PURGE
  ```

- [, <option>] は、オプション引数です。
- [&] は、前述の例の USERID パラメータ文のように、複数行にわたるパラメータ文の各行の最後に必要です。

5. ファイルを保存して閉じます。

テキスト・エディタでパラメータ・ファイルを作成する手順

1. テキスト・エディタで新規ファイルを作成します。
2. エディタの編集機能を使用して、このファイルを説明するコメント行を必要な数だけ入力します (各コメント行の先頭には必ず 2 つのハイフン (--) を挿入してください)。
3. コメント以外の行に、Oracle GoldenGate のパラメータを入力します (パラメータ文ごとに新しい行を開始します)。

   Oracle GoldenGate のパラメータの構文は次のとおりです。

   ```
   <PARAMETER> <argument> [, <option>] [&]
   ```

   条件 :

   - <PARAMETER> は、パラメータの名前です。
   - <argument> は、パラメータの必須引数です。一部のパラメータには引数がありますが、その他のパラメータにはありません。引数は、すべてカンマで区切ってください。次に例を示します。

     ```
     USERID ggs, PASSWORD AACAAAAAAAAAAAIALCKDZIRHOJBHOJUH, &
     ENCRYPTKEY superx128
     RMTHOST sysb, MGRPORT 8040
     RMTTRAIL /home/ggs/dirdat/c1, PURGE
     ```

   - [, <option>] は、オプション引数です。
   - [&] は、前述の例の USERID パラメータ文のように、複数行にわたるパラメータ文の各行の最後に必要です。

4. ファイルを保存して閉じます。GGSCI 外でパラメータ・ファイルを作成する場合、次の点を確認してください。

   - パラメータ・ファイルは、そのファイルを所有する Extract グループまたは Replicat グループの名前で保存するか、そのファイルを Manager プロセスが所有する場合は mgr という名前で保存します。
   - パラメータ・ファイルは、Oracle GoldenGate のインストール・ディレクトリの dirprm ディレクトリに保存します。

パラメータ・ファイルの保存

GGSCI の EDIT PARAMS でパラメータ・ファイルを作成すると、そのファイルは 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。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にはフルパス名を指定しますが、プロセス・グループの作成時にも ADD EXTRACT または ADD REPLICAT コマンドの PARAMS オプションでそのフルパス名を指定する必要があります。

パラメータ・ファイルは、Extract グループまたは Replicat グループに一度関連付けたら、処理の開始後に Oracle GoldenGate を適切に動作させるため、元の場所から移動しないでください。

パラメータ・ファイルの検証

Extract または Replicat のパラメータ・ファイルに含まれるパラメータの構文の正確性をチェックできます。この機能は、他の Oracle GoldenGate プロセスでは使用できません。

パラメータの構文を検証する手順

1. パラメータ・ファイルに CHECKPARAMS パラメータを含めます。
2. GGSCI で START EXTRACT または START REPLICAT コマンドを発行して、関連するプロセスを起動します。

   ```
   START {EXTRACT | REPLICAT} <group name>
   ```

   Oracle GoldenGate によって構文が検査され、結果がレポート・ファイルまたは画面に出力されます。その後、プロセスは停止します。レポート・ファイルの詳細は、第 19 章を参照してください。

3. 次のいずれかの操作を実行します。
   - 構文が正しい場合、プロセスを起動してデータを処理する前に CHECKPARAMS パラメータを削除します。
   - 構文が間違っている場合、レポートの結果に基づいて該当箇所を修正します。必要に応じて、もう 1 回テストを実行して変更内容を検証できます。プロセスを起動してデータを処理する前に、CHECKPARAMS を削除します。

パラメータ・ファイルの表示

パラメータ・ファイルは、オペレーティング・システムのコマンド・シェルから直接表示するか、GGSCI ユーザー・インタフェースから表示することができます。GGSCI からファイルを表示するには、VIEW PARAMS コマンドを使用します。

```
VIEW PARAMS <group name>
```

条件 : <group name> は、mgr(Manager の場合) か、パラメータ・ファイルに関連付けられた Extract グループまたは Replicat グループの名前です。

パラメータ・ファイルが Oracle GoldenGate ディレクトリの dirprm サブディレクトリ以外の場所に作成されている場合、次の例のようにフルパス名を指定します。

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

パラメータ・ファイルの変更

Oracle GoldenGate プロセスは、パラメータ・ファイルの編集前に停止して、パラメータ・ファイルの保存後に再起動する必要があります。プロセスの実行中にパラメータ設定を変更すると、特に表を追加する場合やマッピング・ルールまたはフィルタリング・ルールを変更する場合に予期しない悪影響が発生する可能性があります。

パラメータを変更する手順

1. GGSCI で次のコマンドを使用してプロセスを停止します。ただし、Windows クラスタで Manager を停止する場合は、クラスタ・アドミニストレータを使用して Manager を停止する必要があります。

   ```
   STOP {EXTRACT | REPLICAT | MANAGER} <group name>
   ```

2. テキスト・エディタまたは GGSCI の EDIT PARAMS コマンドを使用してパラメータ・ファイルを開きます。

   ```
   EDIT PARAMS mgr
   ```

3. ファイルを編集して保存します。
4. プロセスを起動します (Windows クラスタで Manager を起動する場合は、クラスタ・アドミニストレータを使用します)。

   ```
   START {EXTRACT | REPLICAT | MANAGER} <group name>
   ```

パラメータ・ファイルの作成の簡略化

Oracle GoldenGate には、パラメータを指定する回数を減らすツールが用意されています。

- ワイルドカード
- OBEY パラメータ
- マクロ
- パラメータ置換

ワイルドカードの使用

オブジェクト名を使用するパラメータの場合、アスタリスク・ワイルドカード (*) を使用して、任意の数の文字を照合できます。所有者名が使用されている場合、その名前はワイルドカードで指定できません。ワイルドカードを使用することで、多数のオブジェクト名または特定のスキーマ内のすべてのオブジェクトを指定する必要がなくなります。

OBEY の使用

よく使用するパラメータ設定が含まれるテキスト・ファイルのライブラリを作成し、その後、OBEY パラメータを使用して、アクティブなパラメータ・ファイルからそれらのファイルを呼び出すことができます。OBEY の構文は次のとおりです。

```
OBEY <file name>
```

条件 : <file name> は、ファイルの相対名またはフルパス名です。

Oracle GoldenGate は、アクティブなパラメータ・ファイル内で OBEY パラメータを検出すると、その参照先ファイルのパラメータを処理してから、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。

マクロの使用

マクロを使用して、パラメータ文を自動的に複数回使用できます。詳細は、272 ページの「Oracle GoldenGate マクロを使用した作業の簡略化および自動化」を参照してください。

パラメータ置換の使用

パラメータ置換を使用すると、パラメータ・ファイルの作成時に静的な値を割り当てるかわりに、実行時に自動的に Oracle GoldenGate のパラメータに値を割り当てることができます。この方法であれば、実行ごとに値が変化する場合に、パラメータ・ファイルを編集したり、異なる設定を含む複数のファイルを管理する必要がなくなります。必要な値は、実行時に簡単にエクスポートできます。パラメータ置換は、任意の Oracle GoldenGate プロセスで使用できます。

パラメータ置換を使用する手順

1. 置換を実行する各パラメータに対して、値のかわりにランタイム・パラメータを宣言します。次の例のように、ランタイム・パラメータ名の前に疑問符 (?) を付けます。

```
SOURCEISFILE
EXTFILE ?EXTFILE
MAP ?TABNAME, TARGET account_targ;
```

2. Oracle GoldenGate プロセスを起動する前に、オペレーティング・システムのシェルを使用して、環境変数を通じてランタイム値を渡します (図 6 および図 7 を参照)。

図 6 Windows でのパラメータ置換

```
C:\GGS> set EXTFILE=C:\ggs\extfile
C:\GGS> set TABNAME=prod.accounts
C:\GGS> replicat paramfile c:\ggs\dirprm\parmfl
```

図 7 UNIX(Korn シェル) でのパラメータ置換

```
$ EXTFILE=/ggs/extfile
$ export EXTFILE
$ TABNAME=prod.accounts
$ export TABNAME
$ replicat paramfile c:\ggs\dirprm\parmfl
```

UNIX では、大 / 小文字が区別されるため、大 / 小文字の使用については、パラメータ・ファイルのパラメータ宣言とシェルの変数割当てで同じである必要があります。

Oracle GoldenGate のパラメータに関する情報の取得

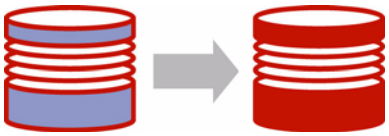
Oracle GoldenGate のパラメータの詳細は、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』を参照してください。

第 4 章

ライブ・レポートのための Oracle GoldenGate の使用

レポート構成の概要

最も基本的な Oracle GoldenGate 構成は、ソース・データベースからレポートや分析などのデータ取得目的のみに使用されるターゲット・データベースへの一方向レプリケートを行う **1 対 1 構成**です。Oracle GoldenGate では、構成内のいずれのシステムにおいても、フィルタリング機能と変換機能を備えたデータの同類転送または異機種転送がサポートされます (サポート内容はデータベース・プラットフォームごとに異なります)。



レポート・トポロジ

Oracle GoldenGate では、レポート用の多くのトポロジが用意されており、スケーラビリティ、可用性およびパフォーマンスに対するユーザー要件に基づいて各モジュールをカスタム構成できます。

単一のターゲット

- 37 ページの「標準レポート構成の作成」
- 39 ページの「ソース・システムでデータ・ポンプを使用するレポート構成の作成」
- 42 ページの「中間システムでデータ・ポンプを使用するレポート構成の作成」
- 46 ページの「カスケード・レポート構成の作成」

複数のターゲット

複数のレポート・ターゲットにデータを送信できます。53 ページの「リアルタイム・データ分散のための Oracle GoldenGate の使用」を参照してください。

レポート構成を選択する場合の考慮事項

データ量

次の場合は標準構成で問題ありません。

- トランザクション負荷が一貫しており、レプリケートされるすべてのオブジェクト全体でほぼ均一に分散される程度の適度な量である場合。

  および

- 長時間実行トランザクションの影響を受ける表、大量の列が変更される表、または Oracle GoldenGate によってデータベースからフェッチする必要のある列 (通常は、LOB が含まれる列、Oracle GoldenGate によって実行される SQL プロシージャからの影響を受ける列、およびトランザクション・ログに記録されない列) が含まれる表が存在しない場合。

現在の環境がこれらの条件を満たしていない場合、パラレル・プロセスの 1 つ以上のセットを追加することを検討してください。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。

フィルタリングおよび変換

データ・フィルタリングおよびデータ変換では、どちらもオーバーヘッドが発生します。これらのアクティビティは、構成エラーにつながるおそれがあります。Oracle GoldenGate で大量のフィルタリングおよび変換を実行する必要がある場合は、1 つ以上のデータ・ポンプを使用してその作業を処理することを検討してください。この目的に Replicat を使用することもできますが、その場合データがフィルタされないため、ネットワーク全体により多くのデータが送信されます。フィルタリングおよび変換は、データ・ポンプと Replicat に分割して、2 つのシステムで分けることが可能です。

データをフィルタする場合、次の処理を使用できます。

- TABLE 文 (Extract) または MAP 文 (Replicat) の FILTER 句または WHERE 句。
- SQL 問合せまたはプロシージャ
- ユーザー・イグジット

データを変換する場合、次の処理を使用できます。

- ネイティブの Oracle GoldenGate 変換関数
- 外部変換ソリューションのルールを適用して操作済データを Oracle GoldenGate に戻す Extract または Replicat プロセスからのユーザー・イグジット。
- ETL ソリューションまたは他の変換エンジンに直接データを配信する Replicat。

Oracle GoldenGate のフィルタリングおよび変換サポートの詳細は、次の項を参照してください。

- 241 ページの「データのマッピングおよび操作」
- 265 ページの「Oracle GoldenGate 処理のカスタマイズ」

読取り専用と高可用性の比較

Oracle GoldenGate のライブ・レポート構成では、読取り専用ターゲットがサポートされます。この構成のターゲットを、高可用性をサポートするトランザクション・アクティビティにも使用する場合、79 ページの「アクティブ / アクティブ型高可用性のための Oracle GoldenGate の使用」を参照してください。

追加情報

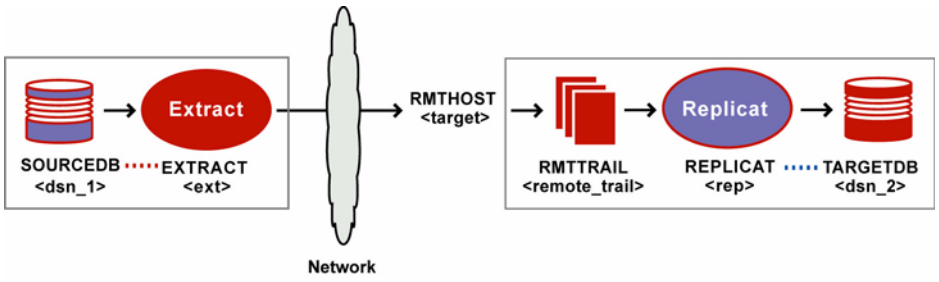
- システムおよびデータベースの構成の追加要件は、使用中のデータベースのタイプに対応する Oracle GoldenGate のインストレーションおよびセットアップ・ガイドを参照してください。
- Teradata の Extract 構成の追加要件は、『Oracle GoldenGate *Teradata* インストレーションおよびセットアップ・ガイド』を参照してください。
- Oracle GoldenGate の変更取得および配信グループの構成の詳細は、120 ページの「オンライン変更同期の構成」を参照してください。
- Oracle GoldenGate のコマンドとパラメータの構文の詳細および説明は、『Oracle GoldenGate *Windows and UNIX* リファレンス・ガイド』を参照してください。
- この構成の調整方法の詳細は、『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。

標準レポート構成の作成

標準の Oracle GoldenGate 構成では、取得データが 1 つの Extract グループによって TCP/IP を通じてターゲット・システムの証跡に送信され、1 つの Replicat グループによって処理されるまで格納されます。

作成するオブジェクトのビジュアル表現は、図 8 を参照してください。

図 8 単一ターゲットに対するレプリケーションの構成要素



ソース・システム

Manager プロセスを構成する手順

1. ソースで、第 2 章の指示に従って Manager プロセスを構成します。

Extract グループを構成する手順

2. ソースで、ADD EXTRACT コマンドを使用して Extract グループを作成します。説明上、このグループを *ext* と呼びます。

```
ADD EXTRACT <ext>, TRANLOG, BEGIN <time> [, THREADS <n>]
```

   - TRANLOG をデータソース・オプションとして使用します。Z/OS 上の DB2 では、TRANLOG に続けてブートストラップ・データセット (BSDS) 名を指定します。

3. ソースで、ADD RMTTRAIL コマンドを使用して、ターゲット・システムに作成するリモート証跡を指定します。

```
ADD RMTTRAIL <remote_trail>, EXTRACT <ext>
```

   - EXTRACT 引数を使用して、この証跡を Extract グループにリンクします。

4. ソースで、EDIT PARAMS コマンドを使用して Extract グループのパラメータ・ファイルを作成します。次のパラメータと、データベース環境に適用する他のパラメータを含めます。

```
-- Identify the Extract group:
EXTRACT <ext>
-- Specify database login information as needed for the database:
[SOURCEDB <dsn_1>,][USERID <user>[, PASSWORD <pw>]]
-- Specify the name or IP address of the target system:
RMTHOST <target>, MGRPORT <portnumber>
-- Specify the remote trail on the target system:
RMTTRAIL <remote_trail>
-- Specify tables to be captured:
TABLE <owner>.<table>;
```

ターゲット・システム

Manager プロセスを構成する手順

5. ターゲットで、第 2 章の指示に従って Manager プロセスを構成します。
6. Manager のパラメータ・ファイルで、PURGEOLDEXTRACTS パラメータを使用して、ローカル証跡からのファイルの消去を制御します。

Replicat グループを構成する手順

7. ターゲットで、Replicat のチェックポイント表を作成します。手順については、121 ページの「チェックポイント表の作成」を参照してください。すべての Replicat グループで同じチェックポイント表を使用できます。
8. ターゲットで、ADD REPLICAT コマンドを使用して Replicat グループを作成します。説明上、このグループを *rep* と呼びます。

```
ADD REPLICAT <rep>, EXTTRAIL <remote_trail>, BEGIN <time>
```

   - EXTTRAIL 引数を使用して、Replicat グループをリモート証跡にリンクします。

9. ターゲットで、EDIT PARAMS コマンドを使用して Replicat グループのパラメータ・ファイルを作成します。次のパラメータと、データベース環境に適用する他のパラメータを含めます。

```
-- Identify the Replicat group:
REPLICAT <rep>
-- State whether or not source and target definitions are identical:
SOURCEDEFS <full_pathname> | ASSUMETARGETDEFS
-- Specify database login information as needed for the database:
[TARGETDB <dsn_2>,] [USERID <user id>[, PASSWORD <pw>]]
-- Specify error handling rules:
REPERROR (<error>, <response>)
-- Specify tables for delivery:
MAP <owner>.<table>, TARGET <owner>.<table>[, DEF <template name>];
```

ソース・システムでデータ・ポンプを使用するレポート構成の作成

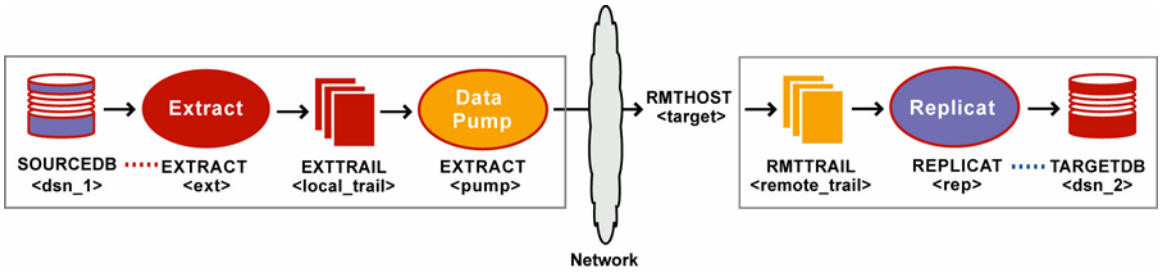
ソース・システムにデータ・ポンプを追加して、プライマリ Extract を TCP/IP 機能から分離することで、記憶域の柔軟性を向上し、プライマリ Extract からフィルタリングおよび変換処理のオーバーヘッドを削減できます。

この構成の場合、プライマリ Extract は、ソース・システムのローカル証跡に書込みを行います。ローカル・データ・ポンプは、その証跡を読み取り、データをターゲット・システムのリモート証跡に移動します (このデータは Replicat によって読み取られます)。

必須ではありませんが、データ・ポンプを使用して、Oracle GoldenGate のパフォーマンスおよびフォルト・トレランスを改善できます。

作成するオブジェクトのビジュアル表現は、図 9 を参照してください。

図 9　データ・ポンプを使用した単一ターゲットに対するレプリケーションの構成要素



ソース・システム

Manager プロセスを構成する手順

1. ソースで、第 2 章の指示に従って Manager プロセスを構成します。
2. Manager のパラメータ・ファイルで、PURGEOLDEXTRACTS パラメータを使用して、ローカル証跡からのファイルの消去を制御します。

プライマリ Extract グループを構成する手順

3. ソースで、ADD EXTRACT コマンドを使用してプライマリ Extract グループを作成します。説明上、このグループを *ext* と呼びます。

```
ADD EXTRACT <ext>, TRANLOG, BEGIN <time> [, THREADS <n>]
```

   - TRANLOG をデータソース・オプションとして使用します。Z/OS 上の DB2 では、TRANLOG に続けてブートストラップ・データセット (BSDS) 名を指定します。

4. ソースで、ADD EXTTRAIL コマンドを使用してローカル証跡を作成します。プライマリ Extract がこの証跡に書込みを行い、データ・ポンプ Extract がそのデータを読み取ります。

```
ADD EXTTRAIL <local_trail>, EXTRACT <ext>
```

   - EXTRACT 引数を使用して、この証跡をプライマリ Extract グループにリンクします。プライマリ Extract グループがこの証跡に書込みを行い、データ・ポンプ・グループがそのデータを読み取ります。

5. ソースで、EDIT PARAMS コマンドを使用してプライマリ Extract グループのパラメータ・ファイルを作成します。次のパラメータと、データベース環境に適用する他のパラメータを含めます。

```
-- Identify the Extract group:
EXTRACT <ext>
-- Specify database login information as needed for the database:
[SOURCEDB <dsn_1>,][USERID <user>[, PASSWORD <pw>]]
-- Specify the local trail that this Extract writes to:
EXTTRAIL <local_trail>
-- Specify tables to be captured:
TABLE <owner>.<table>;
```

データ・ポンプ Extract グループを構成する手順

6. ソースで、ADD EXTRACT コマンドを使用してデータ・ポンプ・グループを作成します。説明上、このグループを *pump* と呼びます。

```
ADD EXTRACT <pump>, EXTTRAILSOURCE <local_trail>, BEGIN <time>
```

   - データソース・オプションとして EXTTRAILSOURCE を使用し、ローカル証跡の名前を指定します。

7. ソースで、ADD RMTTRAIL コマンドを使用して、ターゲット・システムに作成するリモート証跡を指定します。

```
ADD RMTTRAIL <remote_trail>, EXTRACT <pump>
```

   - EXTRACT 引数を使用して、リモート証跡をデータ・ポンプ・グループにリンクします。リンクされたデータ・ポンプは、この証跡に書込みを行います。

8. ソースで、EDIT PARAMS コマンドを使用してデータ・ポンプのパラメータ・ファイルを作成します。次のパラメータと、データベース環境に適用する他のパラメータを含めます。

```
-- Identify the data pump group:
EXTRACT <pump>
-- Specify database login information as needed for the database:
[SOURCEDB <dsn_1>,][USERID <user>[, PASSWORD <pw>]]
-- Specify the name or IP address of the target system:
RMTHOST <target>, MGRPORT <portnumber>
-- Specify the remote trail on the target system:
RMTTRAIL <remote_trail>
-- Allow mapping, filtering, conversion or pass data through as-is:
[PASSTHRU | NOPASSTHRU]
-- Specify tables to be captured:
TABLE <owner>.<table>;
```

注意 PASSTHRU モードを使用するには、ソース・オブジェクト名とターゲット・オブジェクト名が同一である必要があります。列マッピング、フィルタリング、SQLEXEC 機能、変換など、データ操作を必要とする機能は、パラメータ・ファイルに指定できません。通常の処理とパススルー処理を結合するには、PASSTHRU および NOPASSTHRU を異なる TABLE 文と組み合せます。

ターゲット・システム

Manager プロセスを構成する手順

9. ターゲットで、第 2 章の指示に従って Manager プロセスを構成します。

10. Manager のパラメータ・ファイルで、PURGEOLDEXTRACTS パラメータを使用して、ローカル証跡からのファイルの消去を制御します。

Replicat グループを構成する手順

11. ターゲットで、Replicat のチェックポイント表を作成します。手順については、121 ページの「チェックポイント表の作成」を参照してください。

12. ターゲットで、ADD REPLICAT コマンドを使用して Replicat グループを作成します。説明上、このグループを *rep* と呼びます。

```
ADD REPLICAT <rep>, EXTTRAIL <remote_trail>, BEGIN <time>
```

- EXTTRAIL 引数を使用して、Replicat グループをリモート証跡にリンクします。

13. ターゲットで、EDIT PARAMS コマンドを使用して Replicat グループのパラメータ・ファイルを作成します。次のパラメータと、データベース環境に適用する他のパラメータを含めます。

```
-- Identify the Replicat group:
REPLICAT <rep>
-- State whether or not source and target definitions are identical:
SOURCEDEFS <full_pathname> | ASSUMETARGETDEFS
-- Specify database login information as needed for the database:
[TARGETDB <dsn_2>,] [USERID <user id>[, PASSWORD <pw>]]
-- Specify error handling rules:
REPERROR (<error>, <response>)
-- Specify tables for delivery:
MAP <owner>.<table>, TARGET <owner>.<table>[, DEF <template name>];
```

中間システムでデータ・ポンプを使用するレポート構成の作成

中間システムにデータ・ポンプを追加して、記憶域の柔軟性を向上し、ソース・システムからフィルタリングおよび変換処理のオーバーヘッドを削減できます。この構成は、ソース・システムとターゲット・システムが異なるネットワーク内に存在し、それらが直接接続されていない場合にも使用できます。両方のシステムに接続することが可能な中間システムを通じてデータを転送できます。中間システムにデータベースを配置する必要はありません。

この構成では、プライマリ Extract がローカル・データ・ポンプおよび証跡に書込みを行ってから、データ・ポンプがそのデータを中間システムのリモート証跡に送信します。中間システムのデータ・ポンプは、その証跡を読み取り、データをターゲットのリモート証跡に移動し、それを Replicat グループが読み取ります。

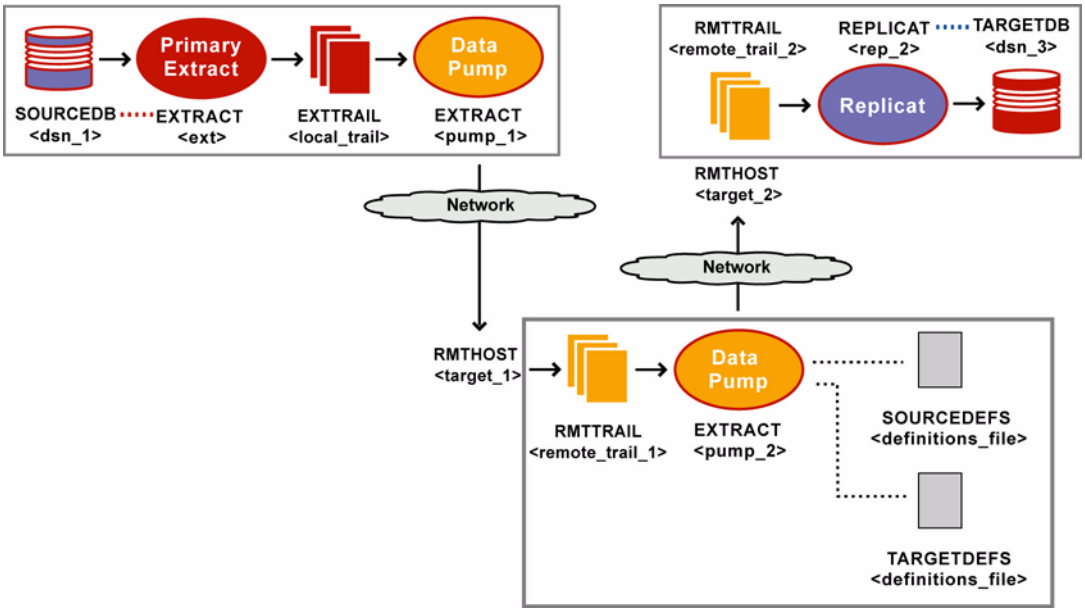
注意　ソース・システムのデータ・ポンプはオプションですが、ネットワークが停止した場合のデータ損失を防止するのに役立ちます。

これは、カスケード・レプリケーションの一種です。ただし、この構成では、データは中間システム上のデータベースに適用されません。Oracle GoldenGate 構成に中間システムのデータベースを含める方法は、58 ページの「カスケード・レポート構成の作成」を参照してください。

中間システムでデータ・ポンプを使用して変換およびトランスフォーメーションを実行するには、DEFGEN ユーティリティを使用してソース定義ファイルおよびターゲット定義ファイルを作成してから、両方のファイルを中間システムに転送する必要があります。この詳細は、第 11 章を参照してください。

作成するオブジェクトのビジュアル表現は、図 10 を参照してください。

図 10 中間システムを通じたレプリケーションの構成要素



ソース・システム

Manager プロセスを構成する手順

1. ソースで、第 2 章の指示に従って Manager プロセスを構成します。
2. Manager のパラメータ・ファイルで、PURGEOLDEXTRACTS パラメータを使用して、証跡からのファイルの消去を制御します。

ソースのプライマリ Extract グループを構成する手順

3. ソースで、ADD EXTRACT コマンドを使用してプライマリ Extract グループを作成します。説明上、このグループを *ext* と呼びます。

   ```
   ADD EXTRACT <ext>, TRANLOG, BEGIN <time> [, THREADS <n>]
   ```

   - TRANLOG をデータソース・オプションとして使用します。Z/OS 上の DB2 では、TRANLOG に続けてブートストラップ・データセット (BSDS) 名を指定します。

4. ソースで、ADD EXTTRAIL コマンドを使用してローカル証跡を作成します。プライマリ Extract がこの証跡に書込みを行い、データ・ポンプ Extract がそのデータを読み取ります。

   ```
   ADD EXTTRAIL <local_trail>, EXTRACT <ext>
   ```

   - EXTRACT 引数を使用して、この証跡をプライマリ Extract グループにリンクします。プライマリ Extract グループがこの証跡に書込みを行い、データ・ポンプ・グループがそのデータを読み取ります。

5. ソースで、EDIT PARAMS コマンドを使用してプライマリ Extract グループのパラメータ・ファイルを作成します。次のパラメータと、データベース環境に適用する他のパラメータを含めます。

```
-- Identify the Extract group:
EXTRACT <ext>
-- Specify database login information as needed for the database:
[SOURCEDB <dsn_1>,][USERID <user>[, PASSWORD <pw>]]
-- Specify the local trail that this Extract writes to:
EXTTRAIL <local_trail>
-- Specify tables to be captured:
TABLE <owner>.<table>;
```

ソースのデータ・ポンプを構成する手順

6. ソースで、ADD EXTRACT コマンドを使用してデータ・ポンプ・グループを作成します。説明上、このグループを *pump_1* と呼びます。

```
ADD EXTRACT <pump_1>, EXTTRAILSOURCE <local_trail>, BEGIN <time>
```

   - データソース・オプションとして EXTTRAILSOURCE を使用し、ローカル証跡の名前を指定します。

7. ソースで、ADD RMTTRAIL コマンドを使用して、中間システムに作成するリモート証跡を指定します。

```
ADD RMTTRAIL <remote_trail_1>, EXTRACT <pump_1>
```

   - EXTRACT 引数を使用して、リモート証跡を *pump_1* データ・ポンプ・グループにリンクします。リンクされたデータ・ポンプは、この証跡に書込みを行います。

8. ソースで、EDIT PARAMS コマンドを使用して *pump_1* データ・ポンプのパラメータ・ファイルを作成します。次のパラメータと、データベース環境に適用する他のパラメータを含めます。

```
-- Identify the data pump group:
EXTRACT <pump_1>
-- Specify database login information as needed for the database:
[SOURCEDB <dsn_1>,][USERID <user>[, PASSWORD <pw>]]
-- Specify the name or IP address of the intermediary system:
RMTHOST <target_1>, MGRPORT <portnumber>
-- Specify the remote trail on the intermediary system:
RMTTRAIL <remote_trail_1>
-- Allow mapping, filtering, conversion or pass data through as-is:
[PASSTHRU | NOPASSTHRU]
-- Specify tables to be captured:
TABLE <owner>.<table>;
```

注意 PASSTHRU モードを使用するには、ソース・オブジェクト名とターゲット・オブジェクト名が同一である必要があります。列マッピング、フィルタリング、SQLEXEC 機能、変換など、データ操作を必要とする機能は、パラメータ・ファイルに指定できません。通常の処理とパススルー処理を結合するには、PASSTHRU および NOPASSTHRU を異なる TABLE 文と組み合せます。

中間システム

中間システムの Manager プロセスを構成する手順

9. 中間システムで、第 2 章の指示に従って Manager プロセスを構成します。

10. Manager のパラメータ・ファイルで、PURGEOLDEXTRACTS パラメータを使用して、証跡からのファイルの消去を制御します。

中間システムのデータ・ポンプを構成する手順

11. 中間システムで、ADD EXTRACT コマンドを使用してデータ・ポンプ・グループを作成します。説明上、このグループを *pump_2* と呼びます。

```
ADD EXTRACT <pump_2>, EXTTRAILSOURCE <local_trail_1>, BEGIN <time>
```

- データソース・オプションとして EXTTRAILSOURCE を使用し、このシステムに作成した証跡の名前を指定します。

12. 中間システムで、ADD RMTTRAIL コマンドを使用して、ターゲット・システムのリモート証跡を指定します。

```
ADD RMTTRAIL <remote_trail_2>, EXTRACT <pump_2>
```

- EXTRACT 引数を使用して、リモート証跡を *pump_2* データ・ポンプにリンクします。リンクされたデータ・ポンプは、この証跡に書込みを行います。

13. 中間システムで、EDIT PARAMS コマンドを使用して *pump_2* データ・ポンプのパラメータ・ファイルを作成します。次のパラメータと、データベース環境に適用する他のパラメータを含めます。

```
-- Identify the data pump group:
EXTRACT <pump_2>
-- State whether or not source and target definitions are identical:
SOURCEDEFS <full_pathname> | ASSUMETARGETDEFS
-- Specify the target definitions file if SOURCEDEFS was used:
TARGETDEFS <full_pathname>
-- Specify the name or IP address of the target system:
RMTHOST <target_2>, MGRPORT <portnumber>
-- Specify the remote trail on the target system:
RMTTRAIL <remote_trail_2>
-- Allow mapping, filtering, conversion or pass data through as-is:
[PASSTHRU | NOPASSTHRU]
-- Specify tables to be captured:
TABLE <owner>.<table>;
```

- データ・ポンプで変換およびトランスフォーメーションを実行する場合、SOURCEDEFS および TARGETDEFS を使用して定義ファイルを指定します。
- データ・ポンプで変換およびトランスフォーメーションを実行する場合、NOPASSTHRU(デフォルト) を使用します。それ以外の場合、PASSTHRU を使用します。

ターゲット・システム

ターゲットの Manager プロセスを構成する手順

14. ターゲット・システムで、第 2 章の指示に従って Manager プロセスを構成します。
15. Manager のパラメータ・ファイルで、PURGEOLDEXTRACTS パラメータを使用して、証跡からのファイルの消去を制御します。

ターゲットの Replicat グループを構成する手順

16. ターゲットで、Replicat のチェックポイント表を作成します。手順については、121 ページの「チェックポイント表の作成」を参照してください。
17. ターゲットで、ADD REPLICAT コマンドを使用して Replicat グループを作成します。説明上、このグループを *rep* と呼びます。

```
ADD REPLICAT <rep>, EXTTRAIL <remote_trail_2>, BEGIN <time>
```

    - EXTTRAIL 引数を使用して、Replicat グループをこのシステムの証跡にリンクします。

18. ターゲットで、EDIT PARAMS コマンドを使用して Replicat グループのパラメータ・ファイルを作成します。次のパラメータと、データベース環境に適用する他のパラメータを含めます。

```
-- Identify the Replicat group:
REPLICAT <rep>
-- State whether or not source and target definitions are identical:
SOURCEDEFS <full_pathname> | ASSUMETARGETDEFS
-- Specify database login information as needed for the database:
[TARGETDB <dsn_2>,] [USERID <user id>[, PASSWORD <pw>]]
-- Specify error handling rules:
REPERROR (<error>, <response>)
-- Specify tables for delivery:
MAP <owner>.<table>, TARGET <owner>.<table>[, DEF <template name>];
```

カスケード・レポート構成の作成

Oracle GoldenGate では、カスケード同期がサポートされます。この同期では、Oracle GoldenGate によってデータ変更がソース・データベースから 2 番目のデータベースに伝播され、次に 3 番目のデータベースに伝播されます。この構成は、次の場合に使用します。

- 1 つ以上のターゲット・システムがソースに直接接続していないが、中間システムが双方向に接続できる場合。
- ソース・システムからのネットワーク・アクティビティを制限する場合。
- 地理的に非常に離れた場所にある 2 つ以上のサーバーにデータを送信する場合 (たとえば、シカゴからロサンゼルスに送信し、次にロサンゼルスから中国全土のサーバーに送信する場合)。

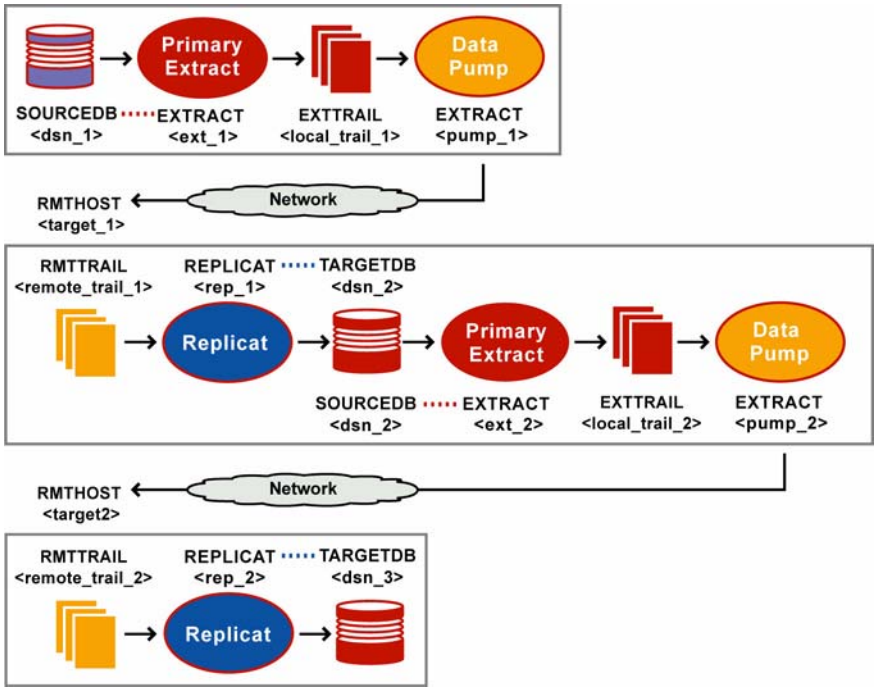
この構成では、次の処理が実行されます。

- ソースのプライマリ Extract は、取得データをローカル証跡に書き込み、データ・ポンプはそのデータをカスケード内の 2 番目のシステムにあるリモート証跡に送信します。

- 2 番目のシステムで、Replicat はデータをローカル・データベースに適用します。
- 同じシステムの別のプライマリ Extract が、ローカル・データベースからデータを取得してローカル証跡に書き込みます。Replicat アクティビティを取得してローカル・ビジネス・アプリケーション・アクティビティを無視するように Extract グループを構成します。この動作を制御する Extract パラメータは、IGNOREAPPLOPS および GETREPLICATES です。
- データ・ポンプは、データをカスケード内の 3 番目のシステムにあるリモート証跡に送信し、そのデータは別の Replicat によってローカル・データベースに適用されます。

注意 レプリケートされた変更を 2 番目のシステムのデータベースに適用する*必要がない*場合は、42 ページの「中間システムでデータ・ポンプを使用するレポート構成の作成」を参照してください。

図 11 カスケード構成



作成するオブジェクトのビジュアル表現は、図 11 を参照してください。

ソース・システム

ソースの Manager プロセスを構成する手順

1. ソースで、第 2 章の指示に従って Manager プロセスを構成します。
2. Manager のパラメータ・ファイルで、PURGEOLDEXTRACTS パラメータを使用して、証跡からのファイルの消去を制御します。

ソースのプライマリ Extract グループを構成する手順

3. ソースで、ADD EXTRACT コマンドを使用してソース Extract グループを作成します。説明上、このグループを *ext_1* と呼びます。

```
ADD EXTRACT <ext_1>, TRANLOG, BEGIN <time> [, THREADS <n>]
```

   - TRANLOG をデータソース・オプションとして使用します。Z/OS 上の DB2 では、TRANLOG に続けてブートストラップ・データセット (BSDS) 名を指定します。

4. ソースで、ADD EXTTRAIL コマンドを使用してローカル証跡を作成します。

```
ADD EXTTRAIL <local_trail_1>, EXTRACT <ext>
```

   - EXTRACT 引数を使用して、この証跡を *ext_1* Extract グループにリンクします。

5. ソースで、EDIT PARAMS コマンドを使用して *ext_1* Extract グループのパラメータ・ファイルを作成します。次のパラメータと、データベース環境に適用する他のパラメータを含めます。

```
-- Identify the Extract group:
EXTRACT <ext_1>
-- Specify database login information as needed for the database:
[SOURCEDB <dsn_1>,][USERID <user>[, PASSWORD <pw>]]
-- Specify the local trail that this Extract writes to:
EXTTRAIL <local_trail_1>
-- Specify tables to be captured:
TABLE <owner>.<table>;
```

ソースのデータ・ポンプを構成する手順

6. ソースで、ADD EXTRACT コマンドを使用してデータ・ポンプ・グループを作成します。説明上、このグループを *pump_1* と呼びます。

```
ADD EXTRACT <pump_1>, EXTTRAILSOURCE <local_trail_1>, BEGIN <time>
```

   - データソース・オプションとして EXTTRAILSOURCE を使用し、ローカル証跡の名前を指定します。

7. ソースで、ADD RMTTRAIL コマンドを使用して、カスケード内の 2 番目のシステムに作成するリモート証跡を指定します。

```
ADD RMTTRAIL <remote_trail_1>, EXTRACT <pump_1>
```

   - EXTRACT 引数を使用して、リモート証跡を *pump_1* データ・ポンプ・グループにリンクします。リンクされたデータ・ポンプは、この証跡に書込みを行います。

8. ソースで、EDIT PARAMS コマンドを使用して *pump_1* データ・ポンプのパラメータ・ファイルを作成します。次のパラメータと、データベース環境に適用する他のパラメータを含めます。

```
-- Identify the data pump group:
EXTRACT <pump_1>
-- Specify database login information as needed for the database:
[SOURCEDB <dsn_1>,][USERID <user>[, PASSWORD <pw>]]
-- Specify the name or IP address of second system in cascade:
RMTHOST <target_1>, MGRPORT <portnumber>
-- Specify the remote trail on the second system:
RMTTRAIL <remote_trail_1>
-- Allow mapping, filtering, conversion or pass data through as-is:
[PASSTHRU | NOPASSTHRU]
-- Specify tables to be captured:
TABLE <owner>.<table>;
```

注意 PASSTHRU モードを使用するには、ソース・オブジェクト名とターゲット・オブジェクト名が同一である必要があります。列マッピング、フィルタリング、SQLEXEC 機能、変換など、データ操作を必要とする機能は、パラメータ・ファイルに指定できません。通常の処理とパススルー処理を結合するには、PASSTHRU および NOPASSTHRU を異なる TABLE 文と組み合せます。

カスケード内の 2 番目のシステム

2 番目のシステムの Manager プロセスを構成する手順

9. 2 番目のシステムで、第 2 章の指示に従って Manager プロセスを構成します。
10. Manager のパラメータ・ファイルで、PURGEOLDEXTRACTS パラメータを使用して、証跡からのファイルの消去を制御します。

2 番目のシステムの Replicat グループを構成する手順

11. Replicat のチェックポイント表を作成します。手順については、121 ページの「チェックポイント表の作成」を参照してください。
12. 2 番目のシステムで、ADD REPLICAT コマンドを使用して Replicat グループを作成します。説明上、このグループを *rep_1* と呼びます。

```
ADD REPLICAT <rep_1>, EXTTRAIL <remote_trail_1>, BEGIN <time>
```

- EXTTRAIL オプションを使用して、*rep_1* グループをローカル・システムにあるリモート証跡 *remote_trail_1* にリンクします。

13. 2 番目のシステムで、EDIT PARAMS コマンドを使用して Replicat グループのパラメータ・ファイルを作成します。次のパラメータと、データベース環境に適用する他のパラメータを含めます。

```
-- Identify the Replicat group:
REPLICAT <rep_1>
-- State whether or not source and target definitions are identical:
SOURCEDEFS <full_pathname> | ASSUMETARGETDEFS
-- Specify database login information as needed for the database:
[TARGETDB <dsn_2>,] [USERID <user id>[, PASSWORD <pw>]]
-- Specify error handling rules:
REPERROR (<error>, <response>)
-- Specify tables for delivery:
MAP <owner>.<table>, TARGET <owner>.<table>[, DEF <template name>];
```

2 番目のシステムの Extract グループを構成する手順

14. 2 番目のシステムで、ADD EXTRACT コマンドを使用してローカル Extract グループを作成します。説明上、このグループを *ext_2* と呼びます。

```
ADD EXTRACT <ext_2>, TRANLOG, BEGIN <time> [, THREADS <n>]
```

- TRANLOG をデータソース・オプションとして使用します。Z/OS 上の DB2 では、TRANLOG に続けてブートストラップ・データセット (BSDS) 名を指定します。

15. 2 番目のシステムで、ADD RMTTRAIL コマンドを使用して、3 番目のシステムに作成するリモート証跡を指定します。

```
ADD EXTTRAIL <local_trail_2>, EXTRACT <ext_2>
```

- EXTRACT 引数を使用して、このローカル証跡を *ext_2* Extract グループにリンクします。

16. 2 番目のシステムで、EDIT PARAMS コマンドを使用して *ext_2* Extract グループのパラメータ・ファイルを作成します。次のパラメータと、データベース環境に適用する他のパラメータを含めます。

```
-- Identify the Extract group:
EXTRACT <ext_2>
-- Specify database login information as needed for the database:
[SOURCEDB <dsn_2>,][USERID <user>[, PASSWORD <pw>]]
-- Specify the local trail that this Extract writes to:
EXTTRAIL <local_trail_2>
-- Ignore local DML, capture Replicat DML:
IGNOREAPPLOPS, GETREPLICATES
-- Specify tables to be captured:
TABLE <owner>.<table>;
```

注意 DDL 操作をレプリケートする場合、IGNOREAPPLOPS, GETREPLICATES 機能は、DDLOPTIONS パラメータによって制御されます。

2 番目のシステムのデータ・ポンプを構成する手順

17. 2 番目のシステムで、ADD EXTRACT コマンドを使用してデータ・ポンプ・グループを作成します。説明上、このグループを *pump_2* と呼びます。

```
ADD EXTRACT <pump_2>, EXTTRAILSOURCE <local_trail_2>, BEGIN <time>
```

- データソース・オプションとして EXTTRAILSOURCE を使用し、ローカル証跡の名前を指定します。

18. 2 番目のシステムで、ADD RMTTRAIL コマンドを使用して、カスケード内の 3 番目のシステムに作成するリモート証跡を指定します。

```
ADD RMTTRAIL <remote_trail_2>, EXTRACT <pump_2>
```

- EXTRACT 引数を使用して、リモート証跡を *pump_2* データ・ポンプ・グループにリンクします。リンクされたデータ・ポンプは、この証跡に書込みを行います。

19. 2 番目のシステムで、EDIT PARAMS コマンドを使用して *pump_2* データ・ポンプのパラメータ・ファイルを作成します。次のパラメータと、データベース環境に適用する他のパラメータを含めます。

```
-- Identify the data pump group:
EXTRACT <pump_2>
-- Specify database login information as needed for the database:
[SOURCEDB <dsn_2>,][USERID <user>[, PASSWORD <pw>]]
-- Specify the name or IP address of third system in cascade:
RMTHOST <target_2>, MGRPORT <portnumber>
-- Specify the remote trail on the third system:
RMTTRAIL <remote_trail_2>
-- Allow mapping, filtering, conversion or pass data through as-is:
[PASSTHRU | NOPASSTHRU]
-- Specify tables to be captured:
TABLE <owner>.<table>;
```

注意 PASSTHRU モードを使用するには、ソース・オブジェクト名とターゲット・オブジェクト名が同一である必要があります。列マッピング、フィルタリング、SQLEXEC 機能、変換など、データ操作を必要とする機能は、パラメータ・ファイルに指定できません。通常の処理とパススルー処理を結合するには、PASSTHRU および NOPASSTHRU を異なる TABLE 文と組み合せます。

カスケード内の 3 番目のシステム

Manager プロセスを構成する手順

20. 3 番目のシステムで、第 2 章の指示に従って Manager プロセスを構成します。

21. Manager のパラメータ・ファイルで、PURGEOLDEXTRACTS パラメータを使用して、証跡からのファイルの消去を制御します。

Replicat グループを構成する手順

22. 3 番目のシステムで、Replicat のチェックポイント表を作成します。手順については、121 ページの「チェックポイント表の作成」を参照してください。

23. 3 番目のシステムで、ADD REPLICAT コマンドを使用して Replicat グループを作成します。説明上、このグループを *rep_2* と呼びます。

```
ADD REPLICAT <rep_2>, EXTTRAIL <remote_trail_2>, BEGIN <time>
```

- EXTTRAIL オプションを使用して、*rep_2* グループを *remote_trail_2* 証跡にリンクします。

24. 3 番目のシステムで、EDIT PARAMS コマンドを使用して Replicat グループのパラメータ・ファイルを作成します。次のパラメータと、データベース環境に適用する他のパラメータを含めます。

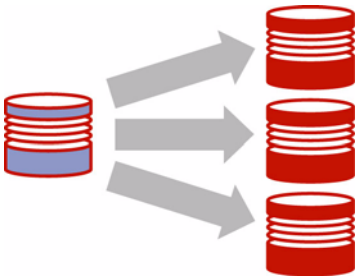
```
-- Identify the Replicat group:
REPLICAT <rep_2>
-- State whether or not source and target definitions are identical:
SOURCEDEFS <full_pathname> | ASSUMETARGETDEFS
-- Specify database login information as needed for the database:
[TARGETDB <dsn_3>,] [USERID <user id>[, PASSWORD <pw>]]
-- Specify error handling rules:
REPERROR (<error>, <response>)
-- Specify tables for delivery:
MAP <owner>.<table>, TARGET <owner>.<table>[, DEF <template name>];
```

第 5 章

リアルタイム・データ分散のための Oracle GoldenGate の使用

データ分散構成の概要

データ分散構成は、**1 対多構成**です。Oracle GoldenGate では、ソース・データベースから任意の数のターゲット・システムへの同期がサポートされます。Oracle GoldenGate では、構成内のいずれのシステムにおいても、フィルタリング機能と変換機能を備えたデータの同類転送または異機種転送がサポートされます (サポート内容はデータベース・プラットフォームごとに異なります)。



データ分散構成の考慮事項

フォルト・トレランス

データ分散構成では、データ・ポンプを使用することで、いずれかのターゲットに対するネットワーク接続に障害が発生した場合でも確実に取得データを他のターゲットに送信できます。ソース構成では、プライマリ Extract グループとデータ・ポンプ Extract グループを、ターゲットごとに 1 つ使用してください。

フィルタリングおよび変換

任意のプロセスを使用してフィルタリングおよび変換を実行できます。ただし、フィルタリング操作の実行にデータ・ポンプを使用することで、その処理のオーバーヘッドがプライマリ Extract グループから削減され、ネットワークを通じて送信されるデータ量が減少します。

データをフィルタする場合、次の処理を使用できます。

- TABLE 文 (Extract) または MAP 文 (Replicat) の FILTER 句または WHERE 句。
- SQL 問合せまたはプロシージャ
- ユーザー・イグジット

データを変換する場合、次の処理を使用できます。

- ネイティブの Oracle GoldenGate 変換関数
- 外部変換ソリューションのルールを適用して操作済データを Oracle GoldenGate に戻す Extract または Replicat プロセスからのユーザー・イグジット。
- ETL ソリューションまたは他の変換エンジンに直接データを配信する Replicat。

データ量

次の場合は標準構成で問題ありません。

- トランザクション負荷が一貫しており、レプリケートされるすべてのオブジェクト全体でほぼ均一に分散される程度の適度な量である場合。

  および

- 長時間実行トランザクションの影響を受ける表、大量の列が変更される表、または Oracle GoldenGate によってデータベースからフェッチする必要のある列 (通常は、LOB が含まれる列、Oracle GoldenGate によって実行される SQL プロシージャからの影響を受ける列、およびトランザクション・ログに記録されない列) が含まれる表が存在しない場合。

現在の環境がこれらの条件を満たしていない場合、パラレル・プロセスの 1 つ以上のセットを追加することを検討してください。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。

読取り専用と高可用性の比較

この構成では、読取り専用ターゲットがサポートされます。この構成の任意のターゲットを、高可用性をサポートするトランザクション・アクティビティにも使用する場合、79 ページの「アクティブ / アクティブ型高可用性のための Oracle GoldenGate の使用」を参照してください。

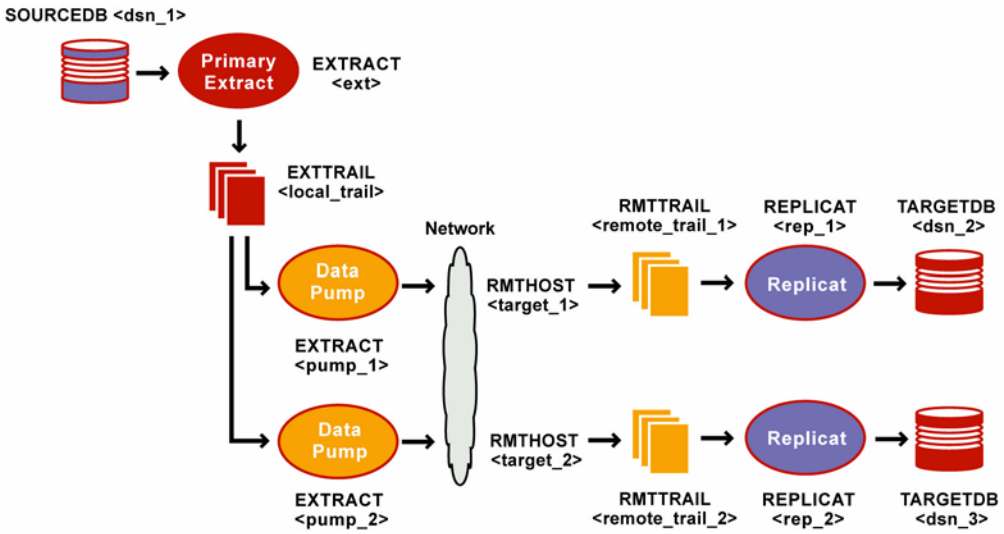
追加情報

- システムおよびデータベースの構成の追加要件は、使用中のデータベースのタイプに対応する Oracle GoldenGate の*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。
- Teradata の Extract 構成の追加要件は、『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。
- Oracle GoldenGate の変更取得および配信グループの構成の詳細は、120 ページの「オンライン変更同期の構成」を参照してください。
- Oracle GoldenGate のコマンドとパラメータの構文の詳細および説明は、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』を参照してください。
- この構成の調整方法の詳細は、『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。

データ分散構成の作成

作成するオブジェクトのビジュアル表現は、図 12 を参照してください。

図 12 データ分散の Oracle GoldenGate 構成要素



ソース・システム

Manager プロセスを構成する手順

1. ソースで、第 2 章の指示に従って Manager プロセスを構成します。
2. Manager のパラメータ・ファイルで、PURGEOLDEXTRACTS パラメータを使用して、ローカル証跡からのファイルの消去を制御します。

プライマリ Extract を構成する手順

3. ソースで、ADD EXTRACT コマンドを使用してプライマリ Extract グループを作成します。説明上、このグループを *ext* と呼びます。

```
ADD EXTRACT <ext>, TRANLOG, BEGIN <time>, [, THREADS]
```

   - TRANLOG をデータソース・オプションとして使用します。Z/OS 上の DB2 では、TRANLOG に続けてブートストラップ・データセット (BSDS) 名を指定します。

4. ソースで、ADD EXTTRAIL コマンドを使用してローカル証跡を作成します。

```
ADD EXTTRAIL <local_trail>, EXTRACT <ext>
```

   - EXTRACT 引数を使用して、この証跡をプライマリ Extract グループにリンクします。プライマリ Extract グループがこの証跡に書込みを行い、データ・ポンプ・グループがそのデータを読み取ります。

5. ソースで、EDIT PARAMS コマンドを使用してプライマリ Extract グループのパラメータ・ファイルを作成します。次のパラメータと、データベース環境に適用する他のパラメータを含めます。

```
-- Identify the Extract group:
EXTRACT <ext>
-- Specify database login information as needed for the database:
[SOURCEDB <dsn_1>,][USERID <user>[, PASSWORD <pw>]]
-- Specify the local trail that this Extract writes to:
EXTTRAIL <local_trail>
-- Specify tables to be captured:
TABLE <owner>.<table>;
```

   - EXTTRAIL を使用してローカル証跡を指定します。

データ・ポンプ Extract グループを構成する手順

6. ソースで、ADD EXTRACT コマンドを使用して、ターゲット・システムごとにデータ・ポンプを作成します。説明上、これらのグループを *pump_1* および *pump_2* と呼びます。

```
ADD EXTRACT <pump_1>, EXTTRAILSOURCE <local_trail>, BEGIN <time>
ADD EXTRACT <pump_2>, EXTTRAILSOURCE <local_trail>, BEGIN <time>
```

   - データソース・オプションとして EXTTRAILSOURCE を使用し、ローカル証跡の名前を指定します。

7. ソースで、ADD RMTTRAIL コマンドを使用して、各ターゲット・システムに作成するリモート証跡を指定します。

```
ADD RMTTRAIL <remote_trail_1>, EXTRACT <pump_1>
ADD RMTTRAIL <remote_trail_2>, EXTRACT <pump_2>
```

   - EXTRACT 引数を使用して、各リモート証跡を異なるデータ・ポンプ・グループにリンクします。リンクされたデータ・ポンプは、この証跡に書込みを行います。

8. ソースで、EDIT PARAMS コマンドを使用して各データ・ポンプのパラメータ・ファイルを作成します。次のパラメータと、データベース環境に適用する他のパラメータを含めます。

   Data pump_1

```
-- Identify the data pump group:
EXTRACT <pump_1>
-- Specify database login information as needed for the database:
[SOURCEDB <dsn_1>,][USERID <user>[, PASSWORD <pw>]]
-- Specify the name or IP address of the first target system:
RMTHOST <target_1>, MGRPORT <portnumber>
-- Specify the remote trail on the first target system:
RMTTRAIL <remote_trail_1>
-- Allow mapping, filtering, conversion or pass data through as-is:
[PASSTHRU | NOPASSTHRU]
-- Specify tables to be captured:
TABLE <owner>.<table>;
```

Data pump_2

```
-- Identify the data pump group:
EXTRACT <pump_2>
-- Specify database login information as needed for the database:
[SOURCEDB <dsn_1>,][USERID <user>[, PASSWORD <pw>]]
-- Specify the name or IP address of the second target system:
RMTHOST <target_2>, MGRPORT <portnumber>
-- Specify the remote trail on the second target system:
RMTTRAIL <remote_trail_2>
-- Allow mapping, filtering, conversion or pass data through as-is:
[PASSTHRU | NOPASSTHRU]
-- Specify tables to be captured:
TABLE <owner>.<table>;
```

注意 PASSTHRU モードを使用するには、ソース・オブジェクト名とターゲット・オブジェクト名が同一である必要があります。列マッピング、フィルタリング、SQLEXEC 関数、変換など、データ操作を必要とする関数は、パラメータ・ファイルに指定できません。通常の処理とパススルー処理を結合するには、PASSTHRU および NOPASSTHRU を異なる TABLE 文と組み合せます。

ターゲット・システム

Manager プロセスを構成する手順

9. 各ターゲットで、第 2 章の指示に従って Manager プロセスを構成します。

10. Manager の各パラメータ・ファイルで、PURGEOLDEXTRACTS パラメータを使用して、証跡からのファイルの消去を制御します。

Replicat グループを構成する手順

11. 各ターゲットで、Replicat のチェックポイント表を作成します。手順については、121 ページの「チェックポイント表の作成」を参照してください。

12. 各ターゲットで、ADD REPLICAT コマンドを使用して、そのシステムのリモート証跡に対する Replicat グループを作成します。説明上、これらのグループを *rep_1* および *rep_2* と呼びます。

Target_1

```
ADD REPLICAT <rep_1>, EXTTRAIL <remote_trail_1>, BEGIN <time>
```

Target_2

```
ADD REPLICAT <rep_2>, EXTTRAIL <remote_trail_2>, BEGIN <time>
```

- EXTTRAIL 引数を使用して、Replicat グループを適切な証跡にリンクします。

13. 各ターゲットで、EDIT PARAMS コマンドを使用して Replicat グループのパラメータ・ファイルを作成します。次のパラメータと、データベース環境に適用する他のパラメータを使用します。

Target_1

```
-- Identify the Replicat group:
REPLICAT <rep_1>
-- State whether or not source and target definitions are identical:
SOURCEDEFS <full_pathname> | ASSUMETARGETDEFS
-- Specify database login information as needed for the database:
[TARGETDB <dsn_2>,] [USERID <user id>[, PASSWORD <pw>]]
-- Specify error handling rules:
REPERROR (<error>, <response>)
-- Specify tables for delivery:
MAP <owner>.<table>, TARGET <owner>.<table>[, DEF <template name>];
```

Target_2

```
-- Identify the Replicat group:
REPLICAT <rep_2>
-- State whether or not source and target definitions are identical:
SOURCEDEFS <full_pathname> | ASSUMETARGETDEFS
-- Specify database login information as needed for the database:
[TARGETDB <dsn_3>,] [USERID <user id>[, PASSWORD <pw>]]
-- Specify error handling rules:
REPERROR (<error>, <response>)
-- Specify tables for delivery:
MAP <owner>.<table>, TARGET <owner>.<table>[, DEF <template name>];
```

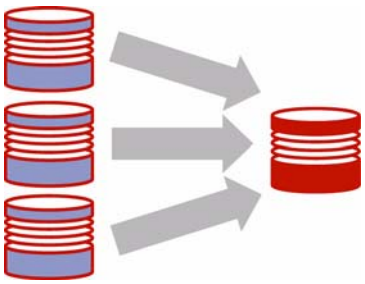
- どの Replicat グループに対しても任意の数の MAP 文を使用できます。特定の Replicat グループに対するすべての MAP 文では、そのグループにリンクされた証跡に含まれる同じオブジェクトを指定する必要があります。

第 6 章

リアルタイム・データ・ウェアハウスのための Oracle GoldenGate の構成

データ・ウェアハウス構成の概要

データ・ウェアハウス構成は、**多対 1 構成**です。複数のソース・データベースが 1 つのターゲット・ウェアハウス・データベースにデータを送信します。Oracle GoldenGate では、構成内のいずれのシステムにおいても、フィルタリング機能と変換機能を備えたデータの同類転送または異機種転送がサポートされます (サポート内容はデータベース・プラットフォームごとに異なります)。



データ・ウェアハウス構成の考慮事項

データ・レコードの分離

この構成では、各ソース・データベースがターゲット・システムに異なるレコードを提供すると仮定されます。2 つ以上のソース・システムの同じ表内に同じレコードが存在し、そのレコードがどのシステムでも変更される可能性がある場合、両方のソースでそのレコードが同時に変更されてターゲット表にレプリケートされたときの競合を解決するため、競合解決ルーチンが必要です。競合解決の詳細は、79 ページの「アクティブ / アクティブ型高可用性のための Oracle GoldenGate の使用」を参照してください。

データ記憶域

ソース・システムとターゲット・システムにデータ記憶域を分割することで、ターゲット・システムに大量のディスク領域を用意する必要がなくなります。これを行うには、ネットワークを通じて各 Extract からターゲットに直接データを送信するのではなく、各ソースでデータ・ポンプを使用します。

- プライマリ Extract は、各ソースのローカル証跡に書込みを行います。
- 各ソースのデータ・ポンプ Extract は、ローカル証跡を読み取り、TCP/IP を通じてそのデータを専用の Replicat グループに送信します。

フィルタリングおよび変換

ソース・システムからデータのすべてをデータ・ウェアハウスに送信するわけではない場合、データ・ポンプを使用してフィルタリングを実行できます。これによって、その処理のオーバーヘッドがプライマリ Extract グループから削減され、ネットワークを通じて送信されるデータ量が減少します。

データをフィルタする場合、次の処理を使用できます。

- TABLE 文 (Extract) または MAP 文 (Replicat) の FILTER 句または WHERE 句。
- SQL 問合せまたはプロシージャ
- ユーザー・イグジット

データを変換する場合、次の処理を使用できます。

- ネイティブの Oracle GoldenGate 変換関数
- 外部変換ソリューションのルールを適用して操作済データを Oracle GoldenGate に戻す Extract または Replicat プロセスからのユーザー・イグジット。
- ETL ソリューションまたは他の変換エンジンに直接データを配信する Replicat。

データ量

次の場合は標準構成で問題ありません。

- トランザクション負荷が一貫しており、レプリケートされるすべてのオブジェクト全体でほぼ均一に分散される程度の適度な量である場合。

  および

- 長時間実行トランザクションの影響を受ける表、大量の列が変更される表、または Oracle GoldenGate によってデータベースからフェッチする必要のある列 (通常は、LOB が含まれる列、Oracle GoldenGate によって実行される SQL プロシージャからの影響を受ける列、およびトランザクション・ログに記録されない列) が含まれる表が存在しない場合。

現在の環境がこれらの条件を満たしていない場合、パラレル・プロセスの 1 つ以上のセットを追加することを検討してください。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。

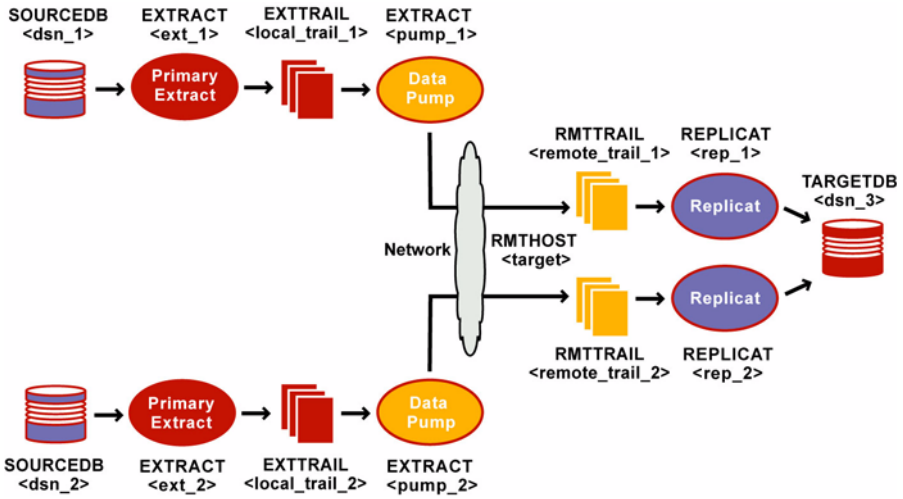
追加情報

- システムおよびデータベースの構成の追加要件は、使用中のデータベースのタイプに対応する Oracle GoldenGate の*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。
- Teradata の Extract 構成の追加要件は、『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。
- Oracle GoldenGate の変更取得および配信グループの構成の詳細は、120 ページの「オンライン変更同期の構成」を参照してください。
- Oracle GoldenGate のコマンドとパラメータの構文の詳細および説明は、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』を参照してください。
- この構成の調整方法の詳細は、『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。

データ・ウェアハウス構成の作成

作成するオブジェクトのビジュアル表現は、図 13 を参照してください。

図 13 データ・ウェアハウスの構成



ソース・システム

Manager プロセスを構成する手順

1. 各ソースで、第 2 章の指示に従って Manager プロセスを構成します。
2. Manager の各パラメータ・ファイルで、PURGEOLDEXTRACTS パラメータを使用して、ローカル・システムにある証跡からのファイルの消去を制御します。

プライマリ Extract グループを構成する手順

3. 各ソースで、ADD EXTRACT コマンドを使用してプライマリ Extract グループを作成します。説明上、これらのグループを *ext_1* および *ext_2* と呼びます。

   Extract_1

   ```
   ADD EXTRACT <ext_1>, TRANLOG, BEGIN <time> [, THREADS <n>]
   ```

   Extract_2

   ```
   ADD EXTRACT <ext_2>, TRANLOG, BEGIN <time> [, THREADS <n>]
   ```

   - TRANLOG をデータソース・オプションとして使用します。Z/OS 上の DB2 では、TRANLOG に続けてブートストラップ・データセット (BSDS) 名を指定します。

4. 各ソースで、ADD EXTTRAIL コマンドを使用してローカル証跡を作成します。

   Extract_1

   ```
   ADD EXTTRAIL <local_trail_1>, EXTRACT <ext_1>
   ```

Extract_2

```
ADD EXTTRAIL <local_trail_2>, EXTRACT <ext_2>
```

- EXTRACT 引数を使用して、各 Extract グループを同じシステムのローカル証跡にリンクします。プライマリ Extract がこの証跡に書込みを行い、データ・ポンプがそのデータを読み取ります。

5. 各ソースで、EDIT PARAMS コマンドを使用してプライマリ Extract のパラメータ・ファイルを作成します。次のパラメータと、データベース環境に適用する他のパラメータを含めます。

Extract_1

```
-- Identify the Extract group:
EXTRACT <ext_1>
-- Specify database login information as needed for the database:
[SOURCEDB <dsn_1>,][USERID <user>[, PASSWORD <pw>]]
-- Specify the local trail that this Extract writes to:
EXTTRAIL <local_trail_1>
-- Specify tables to be captured:
TABLE <owner>.<table>;
```

Extract_2

```
-- Identify the Extract group:
EXTRACT <ext_2>
-- Specify database login information as needed for the database:
[SOURCEDB <dsn_2>,][USERID <user>[, PASSWORD <pw>]]
-- Specify the local trail that this Extract writes to:
EXTTRAIL <local_trail_2>
-- Specify tables to be captured:
TABLE <owner>.<table>;
```

データ・ポンプを構成する手順

6. 各ソースで、ADD EXTRACT コマンドを使用してデータ・ポンプ Extract グループを作成します。説明上、これらのポンプを *pump_1* および *pump_2* と呼びます。

Data pump_1

```
ADD EXTRACT <pump_1>, EXTTRAILSOURCE <local_trail_1>, BEGIN <time>
```

Data pump_2

```
ADD EXTRACT <pump_2>, EXTTRAILSOURCE <local_trail_2>, BEGIN <time>
```

- データソース・オプションとして EXTTRAILSOURCE を使用し、ローカル・システムの証跡の名前を指定します。

7. 各ソースで、ADD RMTTRAIL コマンドを使用してターゲットにリモート証跡を作成します。

Source_1

```
ADD RMTTRAIL <remote_trail_1>, EXTRACT <pump_1>
```

Source_2

```
ADD RMTTRAIL <remote_trail_2>, EXTRACT <pump_2>
```

- EXTRACT 引数を使用して、各リモート証跡を異なるデータ・ポンプにリンクします。データ・ポンプが TCP/IP を通じてこの証跡に書込みを行い、Replicat がそのデータを読み取ります。

8. 各ソースで、EDIT PARAMS コマンドを使用してデータ・ポンプ・グループのパラメータ・ファイルを作成します。次のパラメータと、データベース環境に適用する他のパラメータを含めます。

Data pump_1

```
-- Identify the data pump group:
EXTRACT <pump_1>
-- Specify database login information as needed for the database:
[SOURCEDB <dsn_1>,][USERID <user>[, PASSWORD <pw>]]
-- Specify the name or IP address of the target system:
RMTHOST <target>, MGRPORT <portnumber>
-- Specify the remote trail on the target system:
RMTTRAIL <remote_trail_1>
-- Allow mapping, filtering, conversion or pass data through as-is:
[PASSTHRU | NOPASSTHRU]
-- Specify tables to be captured:
TABLE <owner>.<table>;
```

Data pump_2

```
-- Identify the data pump group:
EXTRACT <pump_2>
-- Specify database login information as needed for the database:
[SOURCEDB <dsn_2>,][USERID <user>[, PASSWORD <pw>]]
-- Specify the name or IP address of the target system:
RMTHOST <target>, MGRPORT <portnumber>
-- Specify the remote trail on the target system:
RMTTRAIL <remote_trail_2>
-- Allow mapping, filtering, conversion or pass data through as-is:
[PASSTHRU | NOPASSTHRU]
-- Specify tables to be captured:
TABLE <owner>.<table>;
```

- データ・ポンプでデータのフィルタリングまたは変換を行う場合、NOPASSTHRU を使用します。また、データベースで定義参照を有効化する場合、必要に応じて SOURCEDB パラメータおよび USERID パラメータを使用します。データ・ポンプでデータのフィルタリングまたは変換を行わない場合、PASSTHRU を使用して参照を回避します。

> **注意** PASSTHRU モードを使用するには、ソース・オブジェクト名とターゲット・オブジェクト名が同一である必要があります。列マッピング、フィルタリング、SQLEXEC 関数、変換など、データ操作を必要とする関数は、パラメータ・ファイルに指定できません。通常の処理とパススルー処理を結合するには、PASSTHRU および NOPASSTHRU を異なる TABLE 文と組み合せます。

ターゲット・システム

Manager プロセスを構成する手順

9. 第 2 章の指示に従って Manager プロセスを構成します。
10. Manager のパラメータ・ファイルで、PURGEOLDEXTRACTS パラメータを使用して、証跡からのファイルの消去を制御します。

Replicat グループを構成する手順

11. ターゲットで、ADD REPLICAT コマンドを使用して、作成した各リモート証跡に対する Replicat グループを作成します。説明上、これらのグループを *rep_1* および *rep_2* と呼びます。

    Replicat_1

    ```
    ADD REPLICAT <rep_1>, EXTTRAIL <remote_trail_1>, BEGIN <time>
    ```

    Replicat_2

    ```
    ADD REPLICAT <rep_2>, EXTTRAIL <remote_trail_2>, BEGIN <time>
    ```

    - EXTTRAIL 引数を使用して、Replicat グループを証跡にリンクします。

12. ターゲットで、EDIT PARAMS コマンドを使用して各 Replicat グループのパラメータ・ファイルを作成します。次のパラメータと、データベース環境に適用する他のパラメータを含めます。

    Replicat_1

    ```
    -- Identify the Replicat group:
    REPLICAT <rep_1>
    -- State whether or not source and target definitions are identical:
    SOURCEDEFS <full_pathname> | ASSUMETARGETDEFS
    -- Specify database login information as needed for the database:
    [TARGETDB <dsn_3>,] [USERID <user id>[, PASSWORD <pw>]]
    -- Specify error handling rules:
    REPERROR (<error>, <response>)
    -- Specify tables for delivery:
    MAP <owner>.<table>, TARGET <owner>.<table>[, DEF <template name>];
    ```

    Replicat_2

    ```
    -- Identify the Replicat group:
    REPLICAT <rep_2>
    -- State whether or not source and target definitions are identical:
    SOURCEDEFS <full_pathname> | ASSUMETARGETDEFS
    -- Specify database login information as needed for the database:
    [TARGETDB <dsn_3>,] [USERID <user id>[, PASSWORD <pw>]]
    -- Specify error handling rules:
    REPERROR (<error>, <response>)
    -- Specify tables for delivery:
    MAP <owner>.<table>, TARGET <owner>.<table>[, DEF <template name>];
    ```

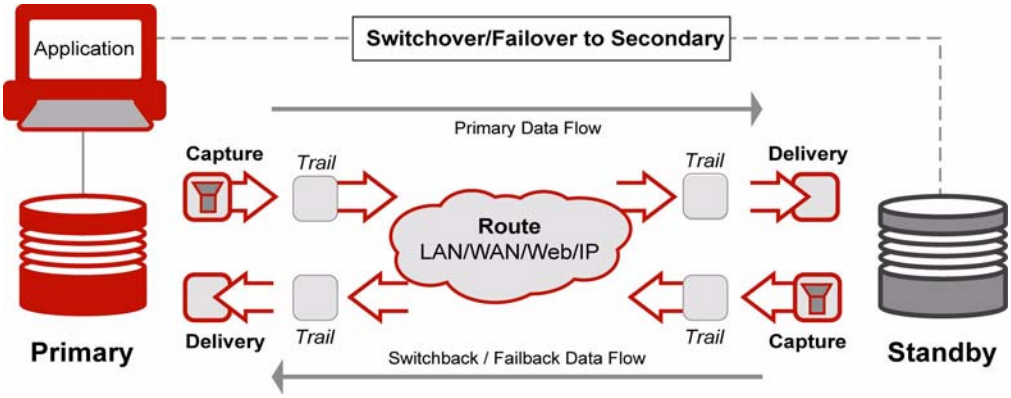
    - どの Replicat グループに対しても任意の数の MAP 文を使用できます。特定の Replicat グループに対するすべての MAP 文では、そのグループにリンクされた証跡に含まれる同じオブジェクトを指定する必要があります。

第 7 章

ライブ・スタンバイ・データベース管理のための Oracle GoldenGate の使用

ライブ・スタンバイ構成の概要

Oracle GoldenGate では、**アクティブ / パッシブ型の双方向**構成がサポートされます。この構成では、Oracle GoldenGate によって、計画停止および計画外停止時のフェイルオーバー用に準備されたライブ・スタンバイ・システムの完全レプリカ・データベースにアクティブ・プライマリ・データベースからデータがレプリケートされます。



この構成では、ライブ・スタンバイ・システム上にアクティブではない Oracle GoldenGate Extract グループとアクティブではないデータ・ポンプが存在します。どちらのグループも、スイッチオーバーまたはフェイルオーバーでライブ・スタンバイ・システムにユーザー・アプリケーションが切り替えられるまで停止されています。ユーザー・アクティビティがスタンバイに移動すると、これらのグループはローカル証跡へのトランザクションの取得を開始します。データは、プライマリ・データベースが再度使用可能になるまでディスク上に格納されます。

プライマリ・システムに障害が発生した場合、Oracle GoldenGate の Manager プロセスと Replicat プロセスは、スタンバイから取得されたデータベース・インスタンスと連携して、プライマリ・システムのリカバリ後に 2 つのシステム間で同等性を回復します。ユーザーは、適切な時点でプライマリ・システムに再度移動され、Oracle GoldenGate 構成は将来のフェイルオーバーに備えて再び準備完了モードに戻ります。

ライブ・スタンバイ構成の考慮事項

信頼できるソース

プライマリ・データベースは、*信頼できるソース*です。これは、通常の運用モード時に*アクティブ・ソース*であるデータベースで、初期同期フェーズや後続のすべての再同期時に他のデータベースの導出元になるデータベースです。信頼できるソースのデータは定期的にバックアップしてください。

複製スタンバイ

ライブ・スタンバイのほとんどの実装では、ソース・データベースとターゲット・データベースは、内容および構造において同一です。データのマッピング、変換およびフィルタリングは、通常、この種の構成では適切な処理ではありませんが、ユーザーのビジネス・モデルで必要とされる場合、Oracle GoldenGate ではこのような機能もサポートされます。これらの機能をサポートするには、TABLE パラメータおよび MAP パラメータのオプションを使用します。

スタンバイ・システムでの DML

アプリケーションが対応している場合、レポートや問合せにライブ・スタンバイ・システムを使用できますが、DML は使用できません。Oracle GoldenGate 構成のオブジェクトに影響するライブ・スタンバイ・システム上にアクティブなトランザクション型アプリケーションを配置する場合、そのアプリケーションはアクティブ / アクティブ構成で設定する必要があります。79 ページの第 8 章を参照してください。

Oracle GoldenGate プロセス

通常の運用モード時には、ライブ・スタンバイ・システムのプライマリ Extract およびデータ・ポンプは停止状態にし、アクティブ・ソースの Replicat も停止状態にします。この処理によって、スタンバイ・システムで誤って発生した DML がアクティブ・ソースに伝播することを防止します。データをアクティブ・ソースからスタンバイ・システムに移動する Extract、データ・ポンプおよび Replicat のみアクティブにできます。

バックアップ・ファイル

プライマリ・システムとスタンバイ・システムにある Oracle GoldenGate の作業ディレクトリは、定期的にバックアップしてください。このバックアップには、Oracle GoldenGate のインストール・ディレクトリのルート・レベルにインストールされているすべてのファイルと、そのディレクトリ内のすべてのサブディレクトリを含める必要があります。Oracle GoldenGate 環境のバックアップを保持しておくことで、プロセス・グループおよびパラメータ・ファイルを再作成する必要がなくなります。

フェイルオーバーの準備

プライマリ・システムとライブ・スタンバイ・システムは、計画済のスイッチオーバーまたは計画外のソース障害の発生時にユーザーが即座にアクセスできるように準備しておく必要があります。高可用性計画の次の構成要素を、各システムですぐに使用できるように準備してください。

- 挿入、更新および削除権限を付与するスクリプト。
- ライブ・スタンバイ・システムでトリガーおよびカスケード削除制約を有効化するスクリプト。(これらは、使用中のデータベースのタイプに対応する Oracle GoldenGate の*インストレーションおよびセットアップ・ガイド*に記載された設定手順で無効になっています。)
- アプリケーション・サーバーをスイッチオーバーし、アプリケーションを起動して、レプリケーション環境に含まれない必須ファイルをコピーするスクリプト。
- ソース・システムに障害が発生した場合にユーザーをライブ・スタンバイに移動するためのフェイルオーバー手順。

データベースによって生成される順序値

データベース生成による値がキーの一部として使用される場合、その値の範囲は、重複しないように各システムで異なっている必要があります。アプリケーションが対応している場合、値に位置識別子を追加して強制的に一意性を確保できます。

Oracle Database の場合、各データベースで一意性を確保しながら順序をレプリケートするように Oracle GoldenGate を構成できます。順序をレプリケートするには、SEQUENCE パラメータおよび MAP パラメータを使用します。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。

データ量

次の場合は標準構成で問題ありません。

- トランザクション負荷が一貫しており、レプリケートされるすべてのオブジェクト全体でほぼ均一に分散される程度の適度な量である場合。

  および

- 長時間実行トランザクションの影響を受ける表、大量の列が変更される表、または Oracle GoldenGate によってデータベースからフェッチする必要のある列 (通常は、LOB が含まれる列、Oracle GoldenGate によって実行される SQL プロシージャからの影響を受ける列、およびトランザクション・ログに記録されない列) が含まれる表が存在しない場合。

現在の環境がこれらの条件を満たしていない場合、パラレル・プロセスの 1 つ以上のセットを追加することを検討してください。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。

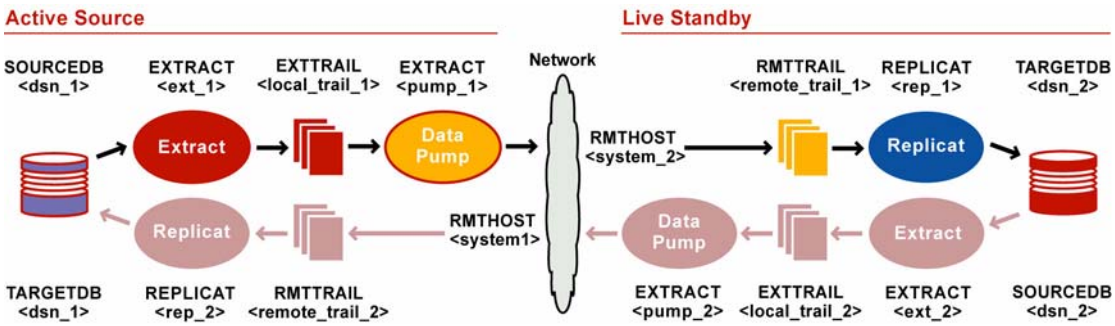
追加情報

- システムおよびデータベースの構成の追加要件は、使用中のデータベースのタイプに対応する Oracle GoldenGate の*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。
- Teradata の Extract 構成の追加要件は、『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。
- Oracle GoldenGate の変更取得および配信グループの構成の詳細は、120 ページの「オンライン変更同期の構成」を参照してください。
- Oracle GoldenGate のコマンドとパラメータの構文の詳細および説明は、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』を参照してください。
- この構成の調整方法の詳細は、『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。

ライブ・スタンバイ構成の作成

作成するオブジェクトのビジュアル表現は、図 14 を参照してください。

図 14 Oracle GoldenGate ライブ・スタンバイの構成要素



両システムの前提条件

1. Replicat のチェックポイント表を作成します。手順については、121 ページの「チェックポイント表の作成」を参照してください。
2. 第 2 章の指示に従って Manager プロセスを構成します。

アクティブ・ソースからスタンバイに対する構成

プライマリ Extract グループを構成する手順

アクティブ・ソースで次の手順を実行します。

1. ADD EXTRACT コマンドを使用してプライマリ Extract グループを作成します。説明上、このグループを *ext_1* と呼びます。

   ```
   ADD EXTRACT <ext_1>, TRANLOG, BEGIN <time> [, THREADS <n>]
   ```

   - TRANLOG をデータソースとして使用します。
   - Z/OS 上の DB2 では、TRANLOG の後にブートストラップ・データセット (BSDS) 名を指定します。

2. ADD EXTTRAIL コマンドを使用してローカル証跡を追加します。説明上、この証跡を *local_trail_1* と呼びます。

   ```
   ADD EXTTRAIL <local_trail_1>, EXTRACT <ext_1>
   ```

   - EXTRACT では、この証跡に書込みを行う *ext_1* グループを指定します。

3. EDIT PARAMS コマンドを使用して *ext_1* グループのパラメータ・ファイルを作成します。次のパラメータと、データベース環境に適用する他のパラメータを含めます。

```
-- Identify the Extract group:
EXTRACT <ext_1>
-- Specify database login information as needed for the database:
[SOURCEDB <dsn_1>,][USERID <user>[, PASSWORD <pw>]]
-- Specify the local trail that this Extract writes to:
EXTTRAIL <local_trail_1>
-- Specify sequences to be captured:
SEQUENCE <owner.sequence>;
-- Specify tables to be captured:
TABLE <owner>.*;
-- Exclude specific tables from capture if needed:
TABLEEXCLUDE <owner.table>
```

データ・ポンプを構成する手順

アクティブ・ソースで次の手順を実行します。

1. ADD EXTRACT コマンドを使用してデータ・ポンプ・グループを作成します。説明上、このグループを *pump_1* と呼びます。

   ```
   ADD EXTRACT <pump_1>, EXTTRAILSOURCE <local_trail_1>, BEGIN <time>
   ```

   - EXTTRAILSOURCE では、データソースとして *local_trail_1* を指定します。

2. ADD RMTTRAIL コマンドを使用して、スタンバイ・システムに作成するリモート証跡を指定します。

   ```
   ADD RMTTRAIL <remote_trail_1>, EXTRACT <pump_1>
   ```

   - EXTRACT では、この証跡に書込みを行う *pump_1* データ・ポンプを指定します。

3. EDIT PARAMS コマンドを使用して *pump_1* グループのパラメータ・ファイルを作成します。次のパラメータと、データベース環境に適用する他のパラメータを含めます。

   ```
   -- Identify the data pump group:
   EXTRACT <pump_1>
   -- Specify database login information as needed for the database:
   [SOURCEDB <dsn_1>,][USERID <user>[, PASSWORD <pw>]]
   -- Specify the name or IP address of the standby system:
   RMTHOST <system_2>, MGRPORT <portnumber>
   -- Specify the remote trail on the standby system:
   RMTTRAIL <remote_trail_1>
   -- Pass data through without mapping, filtering, conversion:
   PASSTHRU
   -- Specify sequences to be captured:
   SEQUENCE <owner.sequence>;
   -- Specify tables to be captured:
   TABLE <owner>.*;
   -- Exclude specific tables from capture if needed:
   TABLEEXCLUDE <owner.table>
   ```

   注意 ライブ・スタンバイ構成では、通常、ソースとターゲットのデータ構造は同一であるため、PASSTHRU モードが仮定されます。このモードでは、列マッピング、フィルタリング、SQLEXEC 関数、変換などのデータ操作は実行できません。

Replicat グループを構成する手順

ライブ・スタンバイ・システムで次の手順を実行します。

1. Replicat のチェックポイント表を作成します。手順については、121 ページの「チェックポイント表の作成」を参照してください。
2. ADD REPLICAT コマンドを使用して Replicat グループを作成します。説明上、このグループを *rep_1* と呼びます。

   ```
   ADD REPLICAT <rep_1>, EXTTRAIL <remote_trail_1>, BEGIN <time>
   ```

   - EXTTRAIL では、この Replicat が読み取る証跡として *remote_trail_1* を指定します。

3. EDIT PARAMS コマンドを使用して *rep_1* グループのパラメータ・ファイルを作成します。次のパラメータと、データベース環境に適用する他のパラメータを含めます。

   ```
   -- Identify the Replicat group:
   REPLICAT <rep_1>
   -- State that source and target definitions are identical:
   ASSUMETARGETDEFS
   -- Specify database login information as needed for the database:
   [TARGETDB <dsn_2>,] [USERID <user id>[, PASSWORD <pw>]]
   -- Specify error handling rules:
   REPERROR (<error>, <response>)
   -- Specify tables for delivery:
   MAP <owner>.*, TARGET <owner>.*;
   -- Exclude specific tables from delivery if needed:
   MAPEXCLUDE <owner.table>
   ```

スタンバイからアクティブ・ソースに対する構成

注意　この手順は、これまでに作成した構成を逆にするイメージです。

プライマリ Extract グループを構成する手順

ライブ・スタンバイ・システムで次の手順を実行します。

1. ADD EXTRACT コマンドを使用して Extract グループを作成します。説明上、このグループを *ext_2* と呼びます。

   ```
   ADD EXTRACT <ext_2>, TRANLOG, BEGIN <time> [, THREADS <n>]
   ```

   - TRANLOG をデータソースとして使用します。
   - Z/OS 上の DB2 では、TRANLOG の後にブートストラップ・データセット (BSDS) 名を指定します。

2. ADD EXTTRAIL コマンドを使用してローカル証跡を追加します。説明上、この証跡を *local_trail_2* と呼びます。

   ```
   ADD EXTTRAIL <local_trail_2>, EXTRACT <ext_2>
   ```

   - EXTRACT では、この証跡に書込みを行う *ext_2* グループを指定します。

3. EDIT PARAMS コマンドを使用して *ext_2* グループのパラメータ・ファイルを作成します。次のパラメータと、データベース環境に適用する他のパラメータを含めます。

```
-- Identify the Extract group:
EXTRACT <ext_2>
-- Specify database login information as needed for the database:
[SOURCEDB <dsn_2>,][USERID <user>[, PASSWORD <pw>]]
-- Specify the local trail that this Extract writes to:
EXTTRAIL <local_trail_2>
-- Specify sequences to be captured:
SEQUENCE <owner.sequence>;
-- Specify tables to be captured:
TABLE <owner>.*;
-- Exclude specific tables from capture if needed:
TABLEEXCLUDE <owner.table>
```

データ・ポンプを構成する手順

ライブ・スタンバイ・システムで次の手順を実行します。

1. ADD EXTRACT コマンドを使用してデータ・ポンプ・グループを作成します。説明上、このグループを *pump_2* と呼びます。

```
ADD EXTRACT <pump_2>, EXTTRAILSOURCE <local_trail_2>, BEGIN <time>
```

   - EXTTRAILSOURCE では、データソースとして *local_trail_2* を指定します。

2. ADD RMTTRAIL コマンドを使用して、アクティブ・ソース・システムに作成するリモート証跡を追加します。

```
ADD RMTTRAIL <remote_trail_2>, EXTRACT <pump_2>
```

   - EXTRACT では、この証跡に書込みを行う *pump_2* データ・ポンプを指定します。

3. EDIT PARAMS コマンドを使用して *pump_2* グループのパラメータ・ファイルを作成します。次のパラメータと、データベース環境に適用する他のパラメータを含めます。

```
-- Identify the data pump group:
EXTRACT <pump_2>
-- Specify database login information as needed for the database:
[SOURCEDB <dsn_2>,][USERID <user>[, PASSWORD <pw>]]
-- Specify the name or IP address of the active source system:
RMTHOST <system_1>, MGRPORT <portnumber>
-- Specify the remote trail on the active source system:
RMTTRAIL <remote_trail_2>
-- Pass data through without mapping, filtering, conversion:
PASSTHRU
-- Specify sequences to be captured:
SEQUENCE <owner.sequence>;
-- Specify tables to be captured:
TABLE <owner>.*;
-- Exclude specific tables from capture if needed:
TABLEEXCLUDE <owner.table>
```

Replicat グループを構成する手順

アクティブ・ソースで次の手順を実行します。

1. ADD REPLICAT コマンドを使用して Replicat グループを作成します。説明上、このグループを *rep_2* と呼びます。

   ```
   ADD REPLICAT <rep_2>, EXTTRAIL <remote_trail_2>, BEGIN <time>
   ```

   - EXTTRAIL では、この Replicat が読み取る証跡として *remote_trail_2* を指定します。

2. EDIT PARAMS コマンドを使用して *rep_2* グループのパラメータ・ファイルを作成します。次のパラメータと、データベース環境に適用する他のパラメータを含めます。

   ```
   -- Identify the Replicat group:
   REPLICAT <rep_2>
   -- State that source and target definitions are identical:
   ASSUMETARGETDEFS
   -- Specify database login information as needed for the database:
   [TARGETDB <dsn_1>,] [USERID <user id>[, PASSWORD <pw>]]
   -- Handle collisions between failback data copy and replication:
   HANDLECOLLISIONS
   -- Specify error handling rules:
   REPERROR (<error>, <response>)
   -- Specify tables for delivery:
   MAP <owner>.*, TARGET <owner>.*;
   -- Exclude specific tables from delivery if needed:
   MAPEXCLUDE <owner.table>
   ```

計画済スイッチオーバーでのユーザー・アクティビティの移動

この手順では、データベースに影響しないシステム・メンテナンスなどの手順をプライマリ・システムで実行できるように、計画された安全な方法でユーザー・アプリケーション・アクティビティをプライマリ・データベースからライブ・スタンバイ・システムに移動します。

ライブ・スタンバイへのユーザー・アクティビティの移動

1. (オプション) セカンダリ・システムでシステム・メンテナンスを実行する必要がある場合、セカンダリ・システムからプライマリ・システムにユーザーを戻した後に、ここでの手順を使用して即座に、または一定時間後に、その処理を実行できます。いずれにせよ、セカンダリ・システムを停止する必要がある場合は、その時間の長さにかかわらず、次の危険があることに注意してください。

   - プライマリ・システムのローカル証跡は、スタンバイのオフライン中にデータが蓄積するためにディスク領域を使い果す可能性があります。この場合、プライマリ Extract は異常終了します。
   - スタンバイのオフライン中にプライマリ・システムに障害が発生すると、機能再開時にデータ変更をライブ・スタンバイに適用できなくなるため、同期状態が破損して、ライブ・スタンバイの完全な再インストールが必要になります。

2. **プライマリ**・システムで、ユーザー・アプリケーションを停止します。ただし、このシステムのプライマリ Extract およびデータ・ポンプの実行は継続し、未処理のトランザクション・データをすべて取得します。

3. **プライマリ**・システムで、EOF に到達して処理するレコードがなくなったことを示すメッセージが戻されるまで、プライマリ Extract で次のコマンドを発行します。このメッセージは、すべてのトランザクションが取得されたことを示します。

```
LAG EXTRACT <ext_1>
```

4. **プライマリ**・システムで、プライマリ Extract プロセスを停止します。

```
LAG EXTRACT <ext_1>
```

5. **プライマリ**・システムで、EOF に到達して処理するレコードがなくなったことを示すメッセージが戻されるまで、データ・ポンプで次のコマンドを発行します。このメッセージは、ポンプによってすべての取得データがライブ・スタンバイに送信されたことを示します。

```
LAG EXTRACT <pump_1>
```

6. **プライマリ**・システムで、データ・ポンプを停止します。

```
STOP EXTRACT <pump_1>
```

7. **ライブ・スタンバイ**・システムで、EOF(ファイルの終わり) に到達したことを示すメッセージが戻されるまで、STATUS REPLICAT コマンドを発行します。このメッセージで、Replicat によってすべてのデータが証跡からデータベースに適用されたことを確認します。

```
STATUS REPLICAT <rep_1>
```

8. **ライブ・スタンバイ**・システムで、Replicat を停止します。

```
STOP REPLICAT <rep_1>
```

9. **ライブ・スタンバイ**・システムで、次の操作を実行します。
   - ビジネス・アプリケーションのユーザーに挿入、更新および削除権限を付与するスクリプトを実行します。
   - トリガーおよびカスケード削除制約を有効化するスクリプトを実行します。
   - アプリケーション・サーバーをスイッチオーバーし、アプリケーションを起動して、レプリケーション環境に含まれない必須ファイルをコピーするスクリプトを実行します。

10. **ライブ・スタンバイ**・システムで、現在のタイムスタンプに基づいてデータの取得を開始するようにプライマリ Extract を変更します。そうしないと、ADD EXTRACT コマンドによってグループが作成された日時にまで遡る操作を検索するために、Extract で不要な時間が費やされます。

```
ALTER EXTRACT <ext_2>, BEGIN NOW
```

11. **ライブ・スタンバイ**・システムで、プライマリ Extract を起動してトランザクション変更の取得に備えます。

```
START EXTRACT <ext_2>
```

    注意 **ライブ・スタンバイ**・システムのデータ・ポンプを起動しないでください。また、**プライマリ**・システムの Replicat を起動しないでください。データは、プライマリ・データベースがユーザー・アクティビティに再度対応できるようになるまで、ライブ・スタンバイのローカル証跡に格納しておく必要があります。

12. ユーザー・アクティビティを**ライブ・スタンバイ**・システムに切り替えます。

13. **プライマリ・システム**で、システム・メンテナンスを実行します。

プライマリ・システムへのユーザー・アクティビティの再移動

1. **ライブ・スタンバイ**・システムで、ユーザー・アプリケーションを停止します。ただし、プライマリ Extract の実行は継続し、未処理のトランザクション・データをすべて取得します。
2. **プライマリ**・システムで、Replicat を起動して、ライブ・スタンバイ・システムからの変更の受信に備えます。

   ```
   START REPLICAT <rep_2>
   ```

3. **ライブ・スタンバイ**・システムで、データ・ポンプを起動して、ローカル証跡に格納されているデータを TCP/IP を通じてプライマリ・システムに移動します。

   ```
   START EXTRACT <pump_2>
   ```

4. **ライブ・スタンバイ**・システムで、EOF に到達して処理するレコードがなくなったことを示すメッセージが戻されるまで、プライマリ Extract で次のコマンドを発行します。このメッセージは、すべてのトランザクションが取得されたことを示します。

   ```
   LAG EXTRACT <ext_2>
   ```

5. **ライブ・スタンバイ**・システムで、プライマリ Extract を停止します。

   ```
   STOP EXTRACT <ext_2>
   ```

6. **ライブ・スタンバイ**・システムで、EOF に到達して処理するレコードがなくなったことを示すメッセージが戻されるまで、データ・ポンプで次のコマンドを発行します。このメッセージは、ポンプによってすべての取得データがプライマリ・システムに送信されたことを示します。

   ```
   LAG EXTRACT <pump_2>
   ```

7. **ライブ・スタンバイ**・システムで、データ・ポンプを停止します。

   ```
   STOP EXTRACT <pump_2>
   ```

8. **プライマリ**・システムで、EOF(ファイルの終わり) に到達したことを示すメッセージが戻されるまで、STATUS REPLICAT コマンドを発行します。このメッセージで、Replicat によってすべてのデータが証跡からデータベースに適用されたことを確認します。

   ```
   STATUS REPLICAT <rep_2>
   ```

9. **プライマリ**・システムで、Replicat を停止します。

   ```
   STOP REPLICAT <rep_2>
   ```

10. **プライマリ**・システムで、次の操作を実行します。
    - ビジネス・アプリケーションのユーザーに挿入、更新および削除権限を付与するスクリプトを実行します。
    - トリガーおよびカスケード削除制約を有効化するスクリプトを実行します。
    - アプリケーション・サーバーをスイッチオーバーし、アプリケーションを起動して、レプリケーション環境に含まれない必須ファイルをコピーするスクリプトを実行します。

11. **プライマリ**・システムで、現在のタイムスタンプに基づいてデータの取得を開始するようにプライマリ Extract を変更します。そうしないと、スタンバイ・システムでのユーザーの作業中にすでに取得してレプリケートされている操作を検索するために、Extract で不要な時間が費やされます。

    ```
    ALTER EXTRACT <ext_1>, BEGIN NOW
    ```

12. **プライマリ**・システムで、プライマリ Extract を起動してトランザクション変更の取得に備えます。

```
START EXTRACT <ext_1>
```

13. ユーザー・アクティビティを**プライマリ**・システムに切り替えます。
14. (オプション) システム・メンテナンスを**ライブ・スタンバイ**・システムで実行する必要がある場合、プライマリ・システムでデータ・ポンプを起動する前にその処理を即座に実行できます。取得データは、スタンバイのオフライン中にプライマリ・システムに蓄積されることに注意してください。
15. **プライマリ**・システムで、データ・ポンプを起動します。

```
START EXTRACT <pump_1>
```

16. **ライブ・スタンバイ**・システムで、Replicat を起動します。

```
START REPLICAT <rep_1>
```

計画外フェイルオーバーでのユーザー・アクティビティの移動

ライブ・スタンバイへのユーザー・アクティビティの移動

この手順では次の操作を実行します。

- ユーザー・アクティビティ用のライブ・スタンバイを準備します。
- プライマリ・システムのすべてのトランザクションがライブ・スタンバイに適用されることを保証します。
- Oracle GoldenGate をアクティブ化してライブ・スタンバイでトランザクション変更を取得します。
- ユーザーをライブ・スタンバイ・システムに移動します。

ライブ・スタンバイにユーザーを移動する手順

ライブ・スタンバイ・システムで次の手順を実行します。

1. Replicat によってすべてのデータが証跡からデータベースに適用されたことを確認するため、EOF (**ファイルの終わり**) **に到達した**ことを示すメッセージが戻されるまで、STATUS REPLICAT コマンドを発行します。

```
STATUS REPLICAT <rep_1>
```

2. Replicat プロセスを停止します。

```
STOP REPLICAT <rep_1>
```

3. ビジネス・アプリケーションのユーザーに挿入、更新および削除権限を付与するスクリプトを実行します。
4. トリガーおよびカスケード削除制約を有効化するスクリプトを実行します。
5. アプリケーション・サーバーをフェイルオーバーし、アプリケーションを起動して、レプリケーション環境に含まれない必須ファイルをコピーするスクリプトを実行します。
6. ライブ・スタンバイでプライマリ Extract プロセスを起動します。

```
START EXTRACT <ext_2>
```

7. スタンバイ・システムにユーザーを移動して、作業を開始させます。

注意 スタンバイでデータ・ポンプ・グループを起動しないでください。ユーザー・トランザクションは、ユーザー・アクティビティをプライマリ・システムに戻すまで、同じ場所で蓄積する必要があります。

プライマリ・システムへのユーザー・アクティビティの再移動

この手順では次の操作を実行します。

- Oracle GoldenGate 環境をリカバリします。
- リストアしたプライマリ・システムにライブ・スタンバイのデータをコピーします。
- コピーの作成中に発生したユーザー・トランザクションを伝播します。
- コピーの結果と伝播された変更とを調整します。
- ユーザーをスタンバイ・システムからリストアしたプライマリ・システムに移動します。
- ライブ・スタンバイを再度保持するためにレプリケーションを準備します。

プライマリ・システムのリカバリの完了後に、次の手順を実行します。

ソース Oracle GoldenGate 環境をリカバリする手順

1. **プライマリ**・システムで、バックアップから Oracle GoldenGate ディレクトリをリカバリします。
2. **プライマリ**・システムで、GGSCI を実行します。
3. **プライマリ**・システムで、プライマリ Extract グループを削除します。

```
DELETE EXTRACT <ext_1>
```

4. **プライマリ**・システムで、ローカル証跡を削除します。

```
DELETE EXTTRAIL <local_trail_1>
```

5. **プライマリ**・システムで、バックアップからリストアしたパラメータ・ファイルと一致するように同じ名前を使用して、プライマリ Extract グループを再度追加します。説明上、このグループを *ext_1* と呼びます。この手順によって、Extract のチェックポイントが障害前の状態からクリーンな状態に初期化されます。

```
ADD EXTRACT <ext_1>, TRANLOG, BEGIN <time> [, THREADS <n>]
```

   - すべてのデータベースで TRANLOG をデータソースとして使用します。
   - Z/OS 上の DB2 では、TRANLOG の後にブートストラップ・データセット (BSDS) 名を指定します。

6. **プライマリ**・システムで、前と同じ名前を使用してローカル証跡を再度追加します。説明上、この証跡を *local_trail_1* と呼びます。

```
ADD EXTTRAIL <local_trail_1>, EXTRACT <ext_1>
```

   - EXTRACT では、この証跡に書込みを行う *ext_1* グループを指定します。

7. **プライマリ**・システムで、Manager プロセスを起動します。

```
START MANAGER
```

スタンバイからプライマリ・システムにデータベースをコピーする手順

1. **プライマリ**・システムで、トリガーおよびカスケード削除制約を無効化するスクリプトを実行します。
2. **スタンバイ**・システムで、データベースのホット・コピーの作成を開始します。
3. **スタンバイ**・システムで、コピーが終了したときの時刻を記録します。
4. **スタンバイ・システム**で、アプリケーションへのユーザー・アクセスを停止します。すべてのオープン・トランザクションの完了を待機します。

コピー中に作成されたデータ変更を伝播する手順

1. **プライマリ**・システムで、Replicat を起動します。

   ```
   START REPLICAT <rep_2>
   ```

2. **ライブ・スタンバイ**・システムで、データ・ポンプを起動します。この操作によって、蓄積されたユーザー・トランザクションをスタンバイからプライマリ・システムの証跡に転送します。

   ```
   START EXTRACT <pump_2>
   ```

3. **プライマリ**・システムで、初期ロード中にユーザーがスタンバイ・システムで生成したすべてのデータ変更が適用されたとわかるまで、INFO REPLICAT コマンドを発行します。前に記録した時刻を参照してください。たとえば、コピーが 12:05 に停止した場合、変更のレプリケーションによってその時刻までデータが適用されていることを確認します。

   ```
   INFO REPLICAT <rep_2>
   ```

4. **プライマリ**・システムで、次のコマンドを発行して HANDLECOLLISIONS パラメータをオフにし、初期ロードのエラー処理を無効化します。

   ```
   SEND REPLICAT <rep_2>, NOHANDLECOLLISIONS
   ```

5. **プライマリ**・システムで、Replicat によってすべてのデータが証跡からデータベースに適用されたことを確認するため、EOF(**ファイルの終わり**) **に到達した**ことを示すメッセージが戻されるまで、STATUS REPLICAT コマンドを発行します。

   ```
   STATUS REPLICAT <rep_2>
   ```

6. **ライブ・スタンバイ**・システムで、データ・ポンプを停止します。この操作によって、ユーザー・トランザクションをスタンバイからプライマリ・システムの証跡に転送することを停止します。

   ```
   STOP EXTRACT <pump_2>
   ```

7. **プライマリ**・システムで、Replicat プロセスを停止します。

   ```
   STOP REPLICAT <rep_2>
   ```

この時点で、プライマリ・データベースとスタンバイ・データベースは、同期状態に戻っていることが必要です。

(オプション) 同期を検証する手順

1. Oracle GoldenGate Veridata などの比較ツールを使用して、ソース・データベースとスタンバイ・データベースの同等性について比較します。
2. Oracle GoldenGate Veridata などの修復ツールを使用して、非同期状態を修復します。

プライマリ・システムにユーザーを切り替える手順

1. **プライマリ**・システムで、ビジネス・アプリケーションのユーザーに挿入、更新および削除権限を付与するスクリプトを実行します。
2. **プライマリ**・システムで、トリガーおよびカスケード削除制約を有効化するスクリプトを実行します。
3. **プライマリ**・システムで、アプリケーション・サーバーをフェイルオーバーし、アプリケーションを起動して、レプリケーション環境に含まれない必須ファイルをコピーするスクリプトを実行します。
4. **プライマリ**・システムで、プライマリ Extract プロセスを起動します。

   ```
   START EXTRACT <ext_1>
   ```

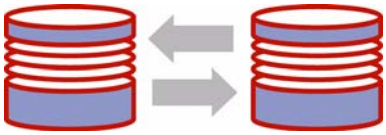
5. **プライマリ**・システムで、アプリケーションへのユーザー・アクセスを許可します。

第 8 章

アクティブ / アクティブ型高可用性のための Oracle GoldenGate の使用

アクティブ / アクティブ構成の概要

Oracle GoldenGate では、**アクティブ / アクティブ型の双方向**構成がサポートされます。この構成では、同一のデータセットが含まれる 2 つのシステムが存在し、アプリケーション・ユーザーはどちらのシステムでもそれらのデータを変更できます。Oracle GoldenGate は、両方のデータセットを最新状態に維持するため、トランザクション・データ変更を一方のデータベースから他方のデータベースにレプリケートします。



双方向構成では、各システムにアクティブな Oracle GoldenGate プロセスの完全なセットがあります。Extract プロセスによって一方のシステムで取得されたデータは、他方のシステムに伝播され、ローカル Replicat プロセスによって適用されます。

この構成では、負荷分散がサポートされます。この機能は、ビジネス・アプリケーションが任意の 2 つのピアで同一である場合に、障害耐久力を確保するために使用できます。双方向同期は、Oracle GoldenGate によってサポートされるすべてのデータベース・タイプでサポートされます。

アクティブ / アクティブ構成の考慮事項

サポートされるデータベース

Oracle GoldenGate では、次のデータベースでアクティブ / アクティブ構成がサポートされます。

- c-tree
- DB2(z/OS および LUW 上)
- MySQL
- Oracle
- SQL/MX
- SQL Server
- Sybase
- Teradata

データベース固有の除外事項

使用中のデータベースのタイプに対応する Oracle GoldenGate のインストレーション・ガイドを参照して、双方向構成のサポートになんらかの制限がないかどうかを確認してください。

TRUNCATES

TRUNCATES の双方向レプリケーションはサポートされませんが、データの双方向へのレプリケート中に、これらの操作を一方向にレプリケートするように構成できます。アクティブ / アクティブ構成で TRUNCATES をレプリケートするには (そのデータベースで Oracle GoldenGate によってサポートされる場合)、TRUNCATES が、ただ 1 つのデータベースから取得され、毎回その同じデータベースからのみ取得される必要があります。

環境は次のように構成します。

- すべてのデータベース・ロールを構成して、この目的のために指定したデータベース以外のどのデータベースからも TRUNCATE を実行できないようにします。
- TRUNCATE が許可されるシステムで、GETTRUNCATES パラメータを含めるように Extract および Replicat のパラメータ・ファイルを構成します。
- 他方のシステムで、IGNORETRUNCATES パラメータを含めるように Extract および Replicat のパラメータ・ファイルを構成します。このシステムでは、Oracle GoldenGate 構成に含まれるアプリケーションによって TRUNCATES を実行しないでください。

DDL

Oracle GoldenGate では、Oracle アクティブ / アクティブ構成で DDL レプリケーションがサポートされます。構成の詳細は、141 ページの第 14 章を参照してください。

データベースの数

スケーラビリティと障害耐久力を向上する最も一般的な peer-to-peer ソリューションでは、同一のデータベースを 2 つ使用します。これより多くすると、停止時間なしで行われる再同期がきわめて複雑になり、競合解決ルーチンの設計や管理がさらに複雑になります。

データベース構成

データベースの 1 つを、*信頼できるソース*として指定する必要があります。これは、初期同期フェーズや必要な後続のすべての再同期時に他のデータベースの導出元になるプライマリ・データベースおよびそのホスト・システムです。信頼できるソースのデータは定期的にバックアップしてください。

アプリケーション設計

アクティブ / アクティブ型のレプリケーションは、市販のパッケージ・ビジネス・アプリケーションでサポートされていない場合、それらのアプリケーションで使用することは推奨されません。これらのアプリケーションで障害となるのは次の点です。

- パッケージ・アプリケーションには、Oracle GoldenGate でサポートされないオブジェクトおよびデータ型が含まれる可能性があります。
- パッケージ・アプリケーションでは、ユーザーが制御できない自動 DML 操作が実行される可能性がありますが、その操作が Oracle GoldenGate によってレプリケートされると、Replicat による適用時に競合が発生します。
- 通常、ユーザーは、アクティブ / アクティブ型のレプリケーションに必要とされる変更を行うためのデータ構造を制御できません。

キー

競合を正確に検出するためには、すべてのレコードは一意の非 NULL 識別子である必要があります。可能であれば、主キーを作成してください。不可能な場合は、一意キーを使用するか、MAP パラメータおよび TABLE パラメータの KEYCOLS オプションを使用して代替キーを作成します。一意の識別子がないと、Oracle GoldenGate では、WHERE 句で有効なすべての列が使用されますが、表に大量の列が含まれる場合、これによってパフォーマンスが低下します。キーの詳細は、ご使用のデータベース用 Oracle GoldenGate インストール・ガイドを参照してください。

データ整合性を保持してエラーを防止するため、特定の表で使用するキーには次のことが求められます。

- その特定の表が存在するすべてのデータベースで同じ列が含まれる必要があります。
- データベース間の対応する行の各セットで同じ値が含まれる必要があります。

トリガーおよびカスケード削除

トリガーおよび ON DELETE CASCADE 制約では、Oracle GoldenGate によってレプリケート可能な DML 操作が生成されます。ローカル DML がこれらの操作でレプリケートされた DML と競合しないようにするには、次の操作を実行します。

- Replicatによって適用されるDML操作を無視するようにトリガーを変更します。Oracleデータベースの一部のバージョンでは、DBOPTIONS パラメータを SUPPRESSTRIGGERS オプションとともに使用して、Replicat セッションのトリガーを無効にできます。重要な詳細は、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』を参照してください。
- ON DELETE CASCADE 制約を無効化し、親表に対するトリガーを使用して子表に対する必要な削除を実行します。このトリガーは、親表で削除操作が実行される前に子表が削除されるように、BEFORE トリガーとして作成してください。これによって、カスケード削除の論理的な順序は逆になりますが、操作を正しい順序でレプリケートし、表が見つからないというターゲットのエラーを防止するために必要です。

  注意 IDENTITY 列は、Sybase の双方向構成では使用できません。SQL Server での IDENTITY の他の制限は、このデータベースに対応する Oracle GoldenGate のインストレーション・ガイドを参照してください。

データベース生成による値

双方向構成では、データベース生成による順序値をレプリケートしないでください。値の範囲は、重複しないように各システムで異なっている必要があります。たとえば、2 つのデータベースが存在する環境では、一方のサーバーで偶数値を生成し、他方で奇数値を生成します。*n* 個のサーバーが存在する環境では、各キーを異なる値で開始し、環境内のサーバーの数を単位として値を増分します。この方法は、すべてのタイプのアプリケーションまたはデータベースに使用できるわけではありません。アプリケーションが対応している場合、値に位置識別子を追加して強制的に一意性を確保できます。

データ量

次の場合は標準構成で問題ありません。

- トランザクション負荷が一貫しており、レプリケートされるすべてのオブジェクト全体でほぼ均一に分散される程度の適度な量である場合。

  および

- 長時間実行トランザクションの影響を受ける表、大量の列が変更される表、または Oracle GoldenGate によってデータベースからフェッチする必要のある列 (通常は、LOB が含まれる列、Oracle GoldenGate によって実行される SQL プロシージャからの影響を受ける列、およびトランザクション・ログに記録されない列) が含まれる表が存在しない場合。

現在の環境がこれらの条件を満たしていない場合、パラレル・プロセスの 1 つ以上のセットを追加することを検討してください。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。

競合解決

個別のシステム上にある同一のデータセットに対して (ほぼ) 同時に変更が行われた場合に発生する衝突を処理するために、統一された競合解決手順を両方のシステムに用意しておく必要があります。アクティブ / アクティブ環境では、競合は即座に識別され、可能なかぎり自動的に処理される必要がありますが、様々なビジネス・アプリケーションがこの処理に関する独自の要件セットを持っています。91 ページの「競合の管理」を参照してください。

追加情報

- Oracle GoldenGate の変更取得および配信グループの構成の詳細は、120 ページの「オンライン変更同期の構成」を参照してください。
- Teradata の Extract 構成の追加要件は、『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。
- Oracle GoldenGate のコマンドとパラメータの構文の詳細および説明は、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』を参照してください。
- この構成の調整方法の詳細は、『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。

データ・ループの防止

双方向構成では、あるシステムから別のシステムにレプリケートされた SQL 変更は、最初のシステムに再度レプリケートされないようにする必要があります。そうしないと、システム間でデータの移動が繰り返され、次の例のように無限ループに陥ります。

1. ユーザー・アプリケーションがシステム A のある行を更新します。
2. Extract は、システム A でその行を抽出してシステム B に送信します。
3. Replicat は、システム B でその行を更新します。
4. Extract は、システム B でその行を抽出してシステム A に戻します。
5. その行がシステム A に適用されます (2 回目)。
6. このループが無限に継続します。

データのループバックを防止するには、状況に応じて次の処理を行う必要があります。

- Replicat によって生成される SQL 操作を取得しないようにします。ただし、Extract のパラメータ・ファイルに指定されているオブジェクトがビジネス・アプリケーションに含まれる場合は、それらのビジネス・アプリケーションによって生成される SQL 操作の取得を有効化します。
- ローカル Replicat トランザクションを識別します (Extract プロセスでそれらのトランザクションを無視するため)。

Replicat 操作の取得の防止

使用中のデータベースに応じて、Replicat 操作を取得しないように明示的に指示する必要がある場合とない場合があります。

Replicat トランザクションの取得の防止 (Teradata)

Replicat によって Teradata データベースに適用される SQL の取得を防止するには、Teradata レプリケーションを上書きするように Replicat セッションを設定します。Replicat のパラメータ・ファイルのルート・レベルで次の SQLEXEC 文を使用します。

```
SQLEXEC "SET SESSION OVERRIDE REPLICATION ON;"
SQLEXEC "COMMIT;"
```

これらの SQLEXEC 文によって、起動時に自動的に Replicat セッションを設定するプロシージャが実行されます。

Replicat トランザクションの取得の防止 (他のデータベース)

Teradata 以外のデータベースでは、Replicat によって生成された SQL 操作が Extract ではデフォルトで無視されます。この機能を制御するパラメータは次のとおりです。

- GETAPPLOPS | IGNOREAPPLOPS: *Replicat 以外*のビジネス・アプリケーションによって生成されたデータ操作 (DML) を、Extract が特定の証跡またはファイルに書き込む内容に含めるかどうかを制御します。
- GETREPLICATES | IGNOREREPLICATES: *Replicat* によって生成された DML 操作を、Extract が特定の証跡またはファイルに書き込む内容に含めるかどうかを制御します。

Extract プロセスを起動する前に、これらのパラメータが存在しないこと、または GETAPPLOPS および IGNOREREPLICATES に設定されていることを確認してください。

Replicat トランザクションの識別

DB2(z/OS および LUW 上)

Extract のパラメータ・ファイルで次のパラメータ文を使用して、Replicat ユーザー名を識別します。

```
TRANLOGOPTIONS EXCLUDEUSER <user name>
```

このパラメータ文によって、このユーザーにより生成されるすべてのデータ・トランザクションが Replicat トランザクションとしてマークされます。ユーザー名は、Extract によって読み取られるトランザクション・レコードに含まれます。

MySQL および NonStop SQL/MX

Extract のパラメータ・ファイルで次のパラメータ文を使用して、Replicat のチェックポイント表の名前を識別します。

```
TRANLOGOPTIONS FILTERTABLE <table_name>
```

Replicat は、その各トランザクションの終了時に、チェックポイント手順の一環としてチェックポイント表にチェックポイントを書き込みます。(これは、ADD CHECKPOINTTABLE コマンドを使用して作成された表です。) すべての Replicat トランザクションにはこの表への書込みが含まれるため、この表を使用して双方向構成の Replicat トランザクションを識別できます。FILTERTABLE によってチェックポイント表の名前を識別し、その表に対する操作が含まれるトランザクションを Extract で無視するようにします。

注意 PURGEDATA は、双方向構成の NonStop SQL/MX ではサポートされません。PURGEDATA/TRUNCATE 操作は DDL であり、暗黙的トランザクションであるため、Oracle GoldenGate ではこのトランザクション内のチェックポイント表を更新できません。

SQL Server

Extract のパラメータ・ファイルで次のパラメータ文を使用して、Replicat トランザクション名を識別します。

```
TRANLOGOPTIONS EXCLUDETRANS <transaction name>
```

このパラメータ文は、Replicat トランザクション名がデフォルトの ggs_repl 以外に設定されている場合にのみ必要です。

Sybase

次のいずれかの操作を実行します。

- Extract のパラメータ・ファイルで次のパラメータ文を使用して、Replicat トランザクション名を識別します。

  ```
  TRANLOGOPTIONS EXCLUDETRANS <transaction name>
  ```

- Extract のパラメータ・ファイルで次のパラメータ文を使用して、Replicat ユーザー名を識別します。

  ```
  TRANLOGOPTIONS EXCLUDEUSER <user name>
  ```

  EXCLUDEUSER によって、このユーザーにより生成されるすべてのトランザクションが Replicat トランザクションとしてマークされます。ユーザー名は、Extract によって読み取られるトランザクション・レコードに含まれます。

- 何も処理をせず、Replicat で ggs_repl というデフォルトのトランザクション名を使用できるようにします。

Teradata

Teradata データベースに適用される Replicat トランザクションを識別する必要はありません。

c-tree

Extract では、c-tree データベースに適用される Replicat トランザクションを自動的に識別します。

Oracle

(Oracle 10g 以上) 次のいずれかの操作を実行して Replicat データベース・ユーザーを指定します。このユーザーによって生成されるすべてのトランザクションが、取得から除外されます。Extract は、トランザクション・レコードでこの情報を使用できます。

- Extract のパラメータ・ファイルで次のパラメータ文を使用して、名前で Replicat データベース・ユーザーを識別します。

  ```
  TRANLOGOPTIONS EXCLUDEUSER <user name>
  ```

- Extract のパラメータ・ファイルで次のパラメータ文を使用して、数値の Oracle ユーザー ID(UID) で Replicat データベース・ユーザーを識別します。

  ```
  TRANLOGOPTIONS EXCLUDEUSERID <user-id>
  ```

(Oracle 9i 以前) GGSCI で ADD TRACETABLE コマンドを使用してトレース表を作成します。

Oracle トレース表を作成する手順

ソース・データベースごとに次の手順を行います。

1. GGSCI を実行します。
2. GGSCI で、次のコマンドを発行してデータベースにログインします。

```
DBLOGIN USERID <user>, PASSWORD <password>
```

   - <user> は、Extract プロセスに割り当てられているデータベース・ユーザー (USERID パラメータにリストされています)、<password> はそのユーザーのパスワードです。表はこのユーザーに属しているスキーマで作成される必要があります。

3. GGSCI で、次のコマンドを発行してトレース表を作成します。

```
ADD TRACETABLE [<owner>.<table name>]
```

   - <owner>.<table name> は、GGS_TRACE のデフォルト以外の名前を使用する場合にのみ必要です。所有者は、Extract プロセスに割り当てられているデータベース・ユーザー (USERID パラメータにリストされています) にする必要があります。可能な場合は、デフォルト名を使用します。

Extract プロセスと Replicat プロセスに Oracle トレース表を関連付ける手順

GGS_TRACE 以外の名前を使用して Oracle トレース表を作成した場合、Extract と Replicat の各パラメータ・ファイルで、TABLE または MAP パラメータの前に次のパラメータ文を指定します。

```
TRACETABLE [<owner>.<table name>]
```

条件 : <owner>.<table name> は、GGS_TRACE のデフォルト以外の名前を使用する場合にのみ必要です。所有者は、USERID パラメータに指定されている Extract ユーザーにする必要があります。可能な場合は、デフォルト名を使用します。

Oracle 表による処理

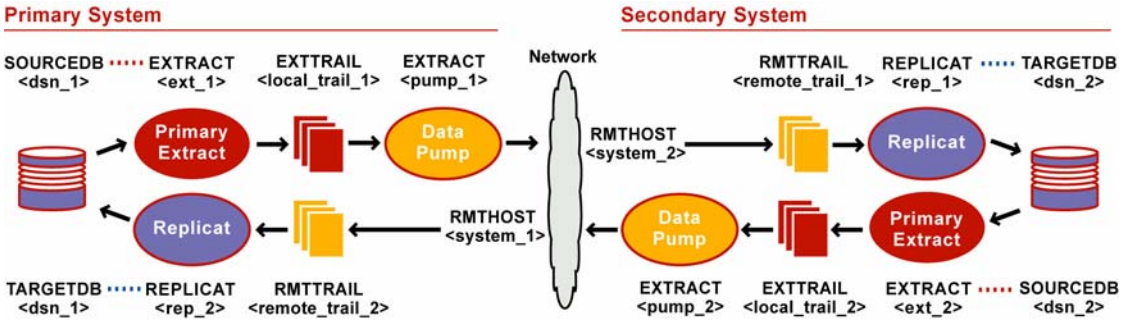
注意 推奨されませんが、トレース表と EXCLUDEUSER(または EXCLUDEUSERID) を両方とも同じ Extract 構成で使用した場合、どちらも正常に動作します。それぞれにどのトランザクションが指定されても、GETREPLICATES または IGNOREREPLICATES のルールに従って処理されます。

アクティブ / アクティブ構成の作成

作成するオブジェクトのビジュアル表現は、図 15 を参照してください。

注意 競合を回避するには、レプリケーション待機時間を最小限に抑える必要があります。この構成で許容できる待機時間レベルを達成できない場合、パラレル・プロセスの追加を検討してください。詳細は、307 ページの「プロセス・グループの追加」を参照してください。

図 15 アクティブ / アクティブ同期の Oracle GoldenGate 構成要素



両システムの前提条件

1. Replicat のチェックポイント表を作成します。手順については、121 ページの「チェックポイント表の作成」を参照してください。
2. 第 2 章の指示に従って Manager プロセスを構成します。

プライマリ・システムからセカンダリ・システムに対する構成

プライマリ Extract グループを構成する手順

プライマリ・システムで次の手順を実行します。

1. ADD EXTRACT コマンドを使用してプライマリ Extract グループを作成します。説明上、このグループを *ext_1* と呼びます。

   ```
   ADD EXTRACT <ext_1>, TRANLOG, BEGIN <time> [, THREADS <n>]
   ```

   - TRANLOG をデータソースとして使用します。
   - Z/OS 上の DB2 では、TRANLOG の後にブートストラップ・データセット (BSDS) 名を指定します。

2. ADD EXTTRAIL コマンドを使用してローカル証跡を追加します。説明上、この証跡を *local_trail_1* と呼びます。

   ```
   ADD EXTTRAIL <local_trail_1>, EXTRACT <ext_1>
   ```

   - EXTRACT では、この証跡に書込みを行う *ext_1* グループを指定します。

3. EDIT PARAMS コマンドを使用して *ext_1* グループのパラメータ・ファイルを作成します。次のパラメータと、データベース環境に適用する他のパラメータを含めます。

   ```
   -- Identify the Extract group:
   EXTRACT <ext_1>
   -- Specify database login information as needed for the database:
   [SOURCEDB <dsn_1>,][USERID <user>[, PASSWORD <pw>]]
   -- Specify the local trail that this Extract writes to:
   EXTTRAIL <local_trail_1>
   --
   -- Exclude Replicat transactions. Uncomment ONE of the following:
   -- DB2 z/OS and LUW, and Sybase:
   ```

```
-- TRANLOGOPTIONS EXCLUDEUSER <Replicat_user>
-- SQL Server and Sybase:
-- TRANLOGOPTIONS EXCLUDETRANS <transaction_name>
-- SQL/MX:
-- TRANLOGOPTIONS FILTERTABLE <checkpoint_table_name>
-- Teradata:
-- SQLEXEC "SET SESSION OVERRIDE REPLICATION ON;"
-- SQLEXEC "COMMIT;"
-- Oracle:
-- TRACETABLE <trace_table_name>
--

-- Specify API commands if Teradata:
VAM <library name>, PARAMS ("<param>" [, "<param>"] [, ...])
-- Capture before images for conflict resolution:
GETUPDATEBEFORES
-- Specify tables to be captured and (optional) columns to fetch:
TABLE <owner>.* [, FETCHCOLS <cols> | FETCHCOLSEXCEPT <cols>];
-- Exclude specific tables from capture if needed:
TABLEEXCLUDE <owner.table>
```

データ・ポンプを構成する手順

プライマリ・システムで次の手順を実行します。

1. ADD EXTRACT コマンドを使用してデータ・ポンプ・グループを作成します。説明上、このグループを *pump_1* と呼びます。

   ```
   ADD EXTRACT <pump_1>, EXTTRAILSOURCE <local_trail_1>, BEGIN <time>
   ```

   - EXTTRAILSOURCE では、データソースとして *local_trail_1* を指定します。

2. ADD RMTTRAIL コマンドを使用して、セカンダリ・システムに作成するリモート証跡を追加します。説明上、この証跡を *remote_trail_1* と呼びます。

   ```
   ADD RMTTRAIL <remote_trail_1>, EXTRACT <pump_1>
   ```

   - EXTRACT では、この証跡に書込みを行う *pump_1* データ・ポンプを指定します。

3. EDIT PARAMS コマンドを使用して *pump_1* グループのパラメータ・ファイルを作成します。次のパラメータと、データベース環境に適用する他のパラメータを含めます。

   ```
   -- Identify the data pump group:
   EXTRACT <pump_1>
   -- Specify database login information as needed for the database:
   [SOURCEDB <dsn_1>,][USERID <user>[, PASSWORD <pw>]]
   -- Specify the name or IP address of the secondary system:
   RMTHOST <system_2>, MGRPORT <portnumber>
   -- Specify the remote trail on the secondary system:
   RMTTRAIL <remote_trail_1>
   -- Pass data through without mapping, filtering, conversion:
   PASSTHRU
   -- Specify tables to be captured:
   TABLE <owner>.*;
   -- Exclude specific tables from capture if needed:
   TABLEEXCLUDE <owner.table>
   ```

注意　通常、双方向構成のデータ構造は同一であるため、PASSTHRU モードでパフォーマンスが向上します。

Replicat グループを構成する手順

セカンダリ・システムで次の手順を実行します。

1. ADD REPLICAT コマンドを使用して Replicat グループを作成します。説明上、このグループを *rep_1* と呼びます。

   ```
   ADD REPLICAT <rep_1>, EXTTRAIL <remote_trail_1>, BEGIN <time>
   ```

   - EXTTRAIL では、この Replicat が読み取る証跡として *remote_trail_1* を指定します。

2. EDIT PARAMS コマンドを使用して *rep_1* グループのパラメータ・ファイルを作成します。次のパラメータと、データベース環境に適用する他のパラメータを含めます。

   ```
   -- Identify the Replicat group:
   REPLICAT <rep_1>
   -- State that source and target definitions are identical:
   ASSUMETARGETDEFS
   -- Specify database login information as needed for the database:
   [TARGETDB <dsn_2>,] [USERID <user id>[, PASSWORD <pw>]]
   -- Specify error handling rules:
   REPERROR (<error>, <response>)
   -- Specify tables for delivery and call conflict-resolution routines:
   MAP <owner>.*, TARGET <owner>.*, SQLEXEC (<SQL specification>);
   -- Exclude specific tables from delivery if needed:
   MAPEXCLUDE <owner.table>
   -- Specify mapping of exceptions to exceptions table:
   MAP <owner>.*, TARGET <owner>.<exceptions>, EXCEPTIONSONLY;
   ```

セカンダリ・システムからプライマリ・システムに対する構成

注意　この手順は、これまでに作成した構成を逆にするイメージです。

プライマリ Extract グループを構成する手順

セカンダリ・システムで次の手順を実行します。

1. ADD EXTRACT コマンドを使用してプライマリ Extract グループを作成します。説明上、このグループを *ext_2* と呼びます。

   ```
   ADD EXTRACT <ext_2>, TRANLOG, BEGIN <time> [, THREADS <n>]
   ```

   - TRANLOG をデータソースとして使用します。
   - Z/OS 上の DB2 では、TRANLOG の後にブートストラップ・データセット (BSDS) 名を指定します。

2. ADD EXTTRAIL コマンドを使用してローカル証跡を追加します。説明上、この証跡を *local_trail_2* と呼びます。

   ```
   ADD EXTTRAIL <local_trail_2>, EXTRACT <ext_2>
   ```

   - EXTRACT では、この証跡に書込みを行う *ext_2* グループを指定します。

3. EDIT PARAMS コマンドを使用して *ext_2* グループのパラメータ・ファイルを作成します。次のパラメータと、データベース環境に適用する他のパラメータを含めます。

```
-- Identify the Extract group:
EXTRACT <ext_2>
-- Specify database login information as needed for the database:
[SOURCEDB <dsn_2>,][USERID <user>[, PASSWORD <pw>]]
-- Specify the local trail that this Extract writes to:
EXTTRAIL <local_trail_2>
--
-- Exclude Replicat transactions. Uncomment ONE of the following:
-- DB2 z/OS and LUW, and Sybase:
-- TRANLOGOPTIONS EXCLUDEUSER <Replicat_user>
-- SQL Server and Sybase:
-- TRANLOGOPTIONS EXCLUDETRANS <transaction_name>
-- SQL/MX:
-- TRANLOGOPTIONS FILTERTABLE <checkpoint_table_name>
-- Teradata:
-- SQLEXEC "SET SESSION OVERRIDE REPLICATION ON;"
-- SQLEXEC "COMMIT;"
-- Oracle:
-- TRACETABLE <trace_table_name>
--
-- Capture before images for conflict resolution:
GETUPDATEBEFORES
-- Specify tables to be captured and (optional) columns to fetch:
TABLE <owner>.* [, FETCHCOLS <cols> | FETCHCOLSEXCEPT <cols>];
-- Exclude specific tables from capture if needed:
TABLEEXCLUDE <owner.table>
```

> **注意** Oracle DBFS データを取得するには、各ノードの TABLE 文で、内部的に生成されたローカルの*読取り / 書込み* DBFS 表を指定します。これらの表を識別し、Oracle GoldenGate による伝播が行われるように DBFS を構成する方法の詳細は、『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。

データ・ポンプを構成する手順

セカンダリ・システムで次の手順を実行します。

1. ADD EXTRACT コマンドを使用してデータ・ポンプ・グループを作成します。説明上、このグループを *pump_2* と呼びます。

```
ADD EXTRACT <pump_2>, EXTTRAILSOURCE <local_trail_2>, BEGIN <time>
```

   - EXTTRAILSOURCE では、データソースとして *local_trail_2* を指定します。

2. ADD RMTTRAIL コマンドを使用して、プライマリ・システムに作成するリモート証跡を追加します。説明上、この証跡を *remote_trail_2* と呼びます。

```
ADD RMTTRAIL <remote_trail_2>, EXTRACT <pump_2>
```

   - EXTRACT では、この証跡に書込みを行う *pump_2* データ・ポンプを指定します。

3. EDIT PARAMS コマンドを使用して *pump_2* グループのパラメータ・ファイルを作成します。次のパラメータと、データベース環境に適用する他のパラメータを含めます。

```
-- Identify the data pump group:
EXTRACT <pump_2>
-- Specify database login information as needed for the database:
[SOURCEDB <dsn_2>,][USERID <user>[, PASSWORD <pw>]]
-- Specify the name or IP address of the primary system:
RMTHOST <system_1>, MGRPORT <portnumber>
-- Specify the remote trail on the primary system:
RMTTRAIL <remote_trail_2>
-- Pass data through without mapping, filtering, conversion:
PASSTHRU
-- Specify tables to be captured:
TABLE <owner>.*;
-- Exclude specific tables from capture if needed:
TABLEEXCLUDE <owner.table>
```

注意 Oracle DBFS データを伝播するには、各ノードの TABLE 文で、内部的に生成されたローカルの*読取り / 書込み* DBFS 表を指定します。Oracle GoldenGate による伝播が行われるように DBFS を構成する方法の詳細は、『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。

Replicat グループを構成する手順

プライマリ・システムで次の手順を実行します。

1. ADD REPLICAT コマンドを使用して Replicat グループを作成します。説明上、このグループを *rep_2* と呼びます。

```
ADD REPLICAT <rep_2>, EXTTRAIL <remote_trail_2>, BEGIN <time>
```

   - EXTTRAIL では、この Replicat が読み取る証跡として *remote_trail_2* を指定します。

2. EDIT PARAMS コマンドを使用して *rep_2* グループのパラメータ・ファイルを作成します。次のパラメータと、データベース環境に適用する他のパラメータを含めます。

```
-- Identify the Replicat group:
REPLICAT <rep_2>
-- State that source and target definitions are identical:
ASSUMETARGETDEFS
-- Specify database login information as needed for the database:
[TARGETDB <dsn_1>,] [USERID <user id>[, PASSWORD <pw>]]
-- Specify error handling rules:
REPERROR (<error>, <response>)
-- Specify tables for delivery and call conflict-resolution routines:
MAP <owner>.*, TARGET <owner>.*, SQLEXEC (<SQL specification>);
-- Exclude specific tables from delivery if needed:
MAPEXCLUDE <owner.table>
-- Specify mapping of exceptions to exceptions table:
MAP <owner>.*, TARGET <owner>.<exceptions>, EXCEPTIONSONLY;
```

注意 Oracle DBFS データをマップするには、内部的に生成されたソースの*読取り / 書込み*表をリモートの*読取り専用*表にマップします。Oracle GoldenGate による伝播が行われるように DBFS を構成する方法の詳細は、『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。

競合の管理

Oracle GoldenGate は非同期ソリューションであるため、個別のシステム上にある同一のデータセットに対して (ほぼ) 同時に変更が行われた場合、競合が発生する可能性があります。競合は、同時変更のタイミングが次の非同期状態のいずれかにつながる場合に発生します。

- レプリケートされた挿入操作がターゲットにすでに存在する行を追加しようとした場合。
- レプリケートされた更新操作の変更前イメージがターゲットの現在の行と一致しない場合。
- レプリケートされた削除操作がターゲットに存在しない行を削除しようとした場合。

たとえば、データベース A のユーザー A がある行を更新し、データベース B のユーザー B が同じ行を更新するとします。ユーザー A のトランザクションがデータベース B に同期されるより前にユーザー B のトランザクションが実行されると、レプリケートされたトランザクションで競合が発生します。

競合発生の可能性の最小化

可能であれば、競合発生のすべての可能性を最小化または排除してください。そのいくつかの方法を次に示します。

- 各データベースで変更できる列を制限するようにアプリケーションを構成します。たとえば、地理的地域に基づいてアクセスを制限できます (異なる販売地域でその独自の顧客レコードのみを変更できるようにするなど)。別の例として、あるデータベースの顧客サービス・アプリケーションには顧客表の NAME 列と ADDRESS 列の変更のみを許可し、別のデータベースの財務アプリケーションには BALANCE 列の変更のみを許可することが可能です。どちらの例の場合でも、同じレコードの同時更新による競合は発生しません。
- 同期の待機時間を最小限に抑えます。データベース A のユーザーA とデータベース B のユーザーB がほぼ同時に同じ行を更新しても、ユーザー B のトランザクションが完了する前にユーザー A のトランザクションがターゲット行にレプリケートされていれば、競合は回避されます。Oracle GoldenGate プロセスのパフォーマンス向上のための推奨事項は、『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。

Oracle GoldenGate で競合を解決する方法

競合解決ルーチンは特定のアプリケーションやビジネス・ルールに応じてカスタマイズする必要があるため、Oracle GoldenGate には競合を処理するデフォルトの手順が用意されていません。Oracle GoldenGate では、ビジネス・ルールに合せて独自に作成されたカスタム競合解決ルーチンを使用できます。

Oracle GoldenGate を構成して競合解決ルーチンを実行および管理する手順

次の構成要素を自動化された競合解決システムに統合する方法の例は、96 ページの「競合の検出および解決の例」を参照してください。

1. 競合解決ルーチンを作成します。Oracle GoldenGate では、ストアド SQL プロシージャまたは問合せとしてルールを指定するのが最も効果的です。93 ページの「適切なルーチンを記述するためのガイドライン」および 94 ページの「競合解決の方法」を参照してください。
2. MAP 文で SQLEXEC パラメータを使用して、Oracle GoldenGate プロセスから競合解決ルーチンをコールします。同じ MAP 文で、必要に応じて FILTER 句を使用すれば、SQL の出力を基に特定の条件でフィルタを実行できます。この方法では、複数の MAP 文を使用して表ごとに異なるルールを作成できます (これらのルールは必要に応じて簡単にも複雑にもできます)。MAP オプションの詳細は、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』を参照してください。

3. REPERROR パラメータを使用して、特定のエラー番号にエラー処理ルールを割り当てます。デフォルトのルールまたはエラー固有のルール (あるいはその両方) を割り当てることができます。
4. (オプション) REPERROR に対する EXCEPTION のレスポンスを指定する場合、MAP パラメータに EXCEPTIONSONLY または MAPEXCEPTION を使用して、競合を発生させた操作を例外表に記録できます。競合を発生させた同じトランザクション内で例外表のデータを問い合せ、トランザクションの結果を調整する計算、手動による競合の処理、またはカスタム・プログラムを使用した競合の解決を行うことができます。たとえば、購買トランザクションの競合が原因で利用できなくなった在庫品目を例外表で追跡することで、トランザクションが終了し競合がなくなった時点で、在庫切れの品目に対応して請求書を調整できます。履歴ツールとして例外表を使用すると、発生した競合とその解決方法を簡単に検出できます。また、競合解決ルーチンのトラブルシューティングにも役立ちます。

Oracle GoldenGate の構成オブジェクトは柔軟であるため、表または表のグループごとに異なるルールを作成できます。たとえば、一部のターゲット表ではレプリケートされた操作を破棄して既存の値を保持する一方、他のターゲット表ではレプリケートされた操作で既存の値を上書きできます。

例外マッピングの使用方法の詳細は、110 ページの「DML 操作中の Replicat エラーの処理」を参照してください。

競合解決のための例外表を作成する手順

例外表の作成方法は、データベースの他の表を作成する場合と同じです。例外表には、次の列を含めることが推奨されます。

- 例外表の主キー値。
- 操作タイプ。
- 競合が発生した表の名前。
- 変更されていた行の主キー値。これにより、必要に応じて行を再検索してデータを検証できます。
- ターゲット行に適用されたデータのイメージ。
- レプリケートされたソース・レコードの変更前イメージ。これは、古いイメージと上書きされたイメージを比較して、競合解決ルーチンによる解決が適切であったかどうかを判断する際に役立ちます これは、在庫などの数量的な値を解決する場合に重要です。
- ターゲット側のレコードに存在していた上書きされたデータの変更前イメージ。これは、通知を行う場合や、解決ルーチンで解決されなかった不一致を解消する場合に役立ちます。
- 競合が解決された時刻。
- 競合で優先されたレコードのタイムスタンプ (タイムスタンプベースの解決を使用している場合)。
- 上書きされる前に元のターゲット行に記録されていたタイムスタンプ (タイムスタンプベースの解決を使用している場合)。優先されたレコードのタイムスタンプよりこの値が新しいか、または古いことがわかれば (ビジネス・ルールに依存)、タイムスタンプベースのルーチンが成功した証拠になります。
- 競合で優先されたシステム、データベースまたはユーザーの名前 (信頼できるソースによる解決を使用している場合)。
- 廃棄表を使用している場合、それに行が破棄されたかどうかを示すインジケータ (Y/N)。
- ユーザーに競合結果を通知するかどうかを示すインジケータ (Y/N)。
- ユーザーに競合結果を通知したかどうかを示すインジケータ (Y/N)。

現在のルーチンがすべての状況で予定どおり動作することを確認したら、解決ルーチンのオーバーヘッドを削減するために、例外表に記録されるデータの量を減らすことができます。

適切なルーチンを記述するためのガイドライン

同じ競合は同じ最終結果となるように、すべてのデータベースで同一の競合解決手順を使用してください。

1 つの競合解決方法で、発生する可能性のあるすべての競合に対応できるわけではありません。障害のリスクを最小化するため、状況に応じて、論理的な優先順でコールできる複数のルーチンを作成する必要があります。

変更前イメージを使用した比較の実行

Oracle GoldenGate には、変更前の行の値を取得するメカニズムがあります。競合解決ルーチンで各行の変更前イメージを使用して、更新操作のための比較を実行できます。

変更前イメージを使用すると、ターゲット表に含まれる 1 つ以上の列の既存イメージと、Replicat によって適用される変更における同じ列の変更前イメージとを比較できます。通常、Oracle GoldenGate では、なんらかの変更が発生するまでソースとターゲットのデータは同一であると仮定されるため、これらの比較は実行されません。ただし、いずれかのシステムで同じ行が変更される可能性がある場合、同期状態を仮定することはできません。この場合、競合解決ルールを適用するには変更前後の比較が必要です。使用する変更前イメージの詳細は、96 ページの競合解決の例を参照してください。

変更前の値を使用する手順

1. 変更前の値を抽出するには、GETUPDATEBEFORES パラメータを使用します。
2. 競合解決手順で変更前の値を参照するには、次の形式を使用します。

```
BEFORE.<column_name>
```

フェッチされた列値の使用

データベースでは変更された列のみがトランザクション・ログに記録される場合、状況に応じてデータベースから列値 (TIMESTAMP 列や外部キー列など) をフェッチして、競合解決ルーチンで使用する必要があります。

データベースから列値をフェッチする手順

Extract のパラメータ・ファイルで TABLE 文の FETCHCOLS オプションと FETCHCOLSEXCEPT オプションを使用し、指定した列の値をデータベースからフェッチします。

(Oracle) Oracle Database から変更前の値を使用する場合の精度を高めるため、Oracle GoldenGate では、FETCHCOLS を使用するかわりに、COLS 句を指定して ADD TRANDATA コマンドで発行することが推奨されます。競合解決に必要な列は、サプリメンタル・ロギングに含まれます。

競合解決の方法

次に、競合解決ルーチンで通常使用されるいくつかの方法を示します。異なる方法を組み合せて使用することで、1 つの方法が失敗しても別の方法を適用できます。ルーチンの作成で支援が必要な場合は、Oracle サポートに連絡してください。詳細は、http://support.oracle.com にアクセスしてください。

タイムスタンプによる優先方法

競合はタイムスタンプに基づいて解決でき、そこでは最初に (場合によっては最後に) 変更されたレコードが常に優先されます。タイムスタンプによる優先方法を使用するには、次の要件があります。

- 各レコードには、レコードが挿入または更新された日時を示すタイムスタンプ列が含まれている必要があります。タイムスタンプを列に格納するには、データベース・トリガーを使用するか、アプリケーション・コードを変更します。
- すべてのデータベースのタイムスタンプが同一で、すべてのサーバーのタイムゾーンを同じにする必要があります。

タイムスタンプに基づいて競合を検出して解決する場合は、まず行を通常どおり適用します。レプリケートされた行のタイムスタンプ列の変更前イメージを、データベースの現在のタイムスタンプ列と比較します。

- これらが一致している場合、競合はありません。
- これらが異なる場合、データベースの行のタイムスタンプを、レプリケートされた行のタイムスタンプの変更後イメージと比較します。

最も古いタイムスタンプ (またはルールによっては最も新しいタイムスタンプ) を持つ行が優先されます。

信頼できるソースによる優先方法

競合はトランザクションのソースに基づいて解決できます。*信頼できるソース*からのデータ変更は他のソースからの変更よりも優先されます。この解決方法は、実装が比較的容易です。信頼できるソースには、サーバーの場所などの簡単なものや、トランザクションを実行するデータベース・ユーザーに割り当てられた優先度に基づく階層構造などの複雑なものがあります。

タイムスタンプと信頼できるソースの組合せによる優先方法

タイムスタンプか信頼できるソースの一方では十分でない場合、両方を組み合せて使用することで競合を解決できます。これは、ローカルの変更とレプリケートされた変更の両方が同じ行に対して同時に発生するためにタイムスタンプに依存できない場合に必要になることがあります。2 番目の競合解決ルーチンで、信頼できるソースによる解決を実装できます。

デルタ値による優先方法

アプリケーションで UPDATE 文を使用して一定の数量を減分する場合、単に一方のソースまたはタイムスタンプを他方に優先させるルールでは、両方の変更の正味の結果が解決されないため、競合を正確には解決できません。両方のシステムのデータ値は、不正確になります。このような例には、在庫残高、口座残高および売上残高などがあります。その場合は、変更前イメージを現行イメージと比較して実際の値のかわりに正味の変更値をレプリケートすることで、数量的に解決する必要があります。97 ページの例を参照してください。

レプリケーションに対する SQL 固有のルールによる優先方法

SQL 操作のタイプごとにルールを作成できます。

- 挿入操作 : レプリケートされた挿入のキー値がターゲット表に含まれる既存のレコードのキー値と一致する場合、レプリケートされた行で既存の行を置換できます。
- 更新操作 : レプリケートされた更新のキーがターゲット表に存在しない場合、更新を挿入に変換できます。
- 削除操作 : レプリケートされた削除のキーがターゲット表に存在しない場合、その削除は無視できます (ビジネス・ポリシーで許可されることを前提とします)。1 つのデータベースから行を削除する場合、その行が別のデータベースに存在していなくても問題ありません (そのレコードはいずれにせよ削除されます)。

元の値と変換された値を追跡するには、例外表を使用します。91 ページの「Oracle GoldenGate で競合を解決する方法」を参照してください。

アプリケーション固有のルール

Oracle GoldenGate によって実行される SQL プロシージャとユーザー・イグジットを使用して、分散処理に対応しているアプリケーションに付属する組込みの競合解決方法をサポートできます。このような方法には IP 永続ルーターやアプリケーション権限などがあり、同じデータが複数のユーザーによって変更されるのを防ぎます。SQL プロシージャとユーザー・イグジットの使用方法の詳細は、第 18 章を参照してください。

処理の終了

表に対して他の変更が加えられる前にエラーを解決できるように、Replicat の処理を終了できます。これは、自動競合解決ルーチンが正しく動作しない場合や、例外が発生する場合にのみ推奨されます。Replicat を停止すると、ターゲット・データの待機時間が増加します。

解決通知の処理

競合の検出と解決が行われた場合、ビジネス・アプリケーションのユーザーの想定している結果が変更されたことを、必要に応じて各自に通知する必要があります。たとえば、2 人の顧客が航空機の座席を予約して、2 人に同じ座席が割り当てられた場合、どちらか 1 人に対して代替席が割り当てられる (割り当てられた) と通知する必要があります。

通知を処理する最も簡単な方法は、例外表に対して定期的に実行されるバッチ・ジョブを作成し、適切なメッセージを発行して、ユーザーに対する代替結果を取得し、データベースを必要に応じて更新してそれらの変更を反映することです。このジョブは、データベースにコミットされたデータのみが作業の対象となるように、別のデータベース・ユーザーとして実行する必要があります。

例外表の使用方法の詳細は、91 ページの「Oracle GoldenGate で競合を解決する方法」を参照してください。

競合の検出および解決の例

タイムスタンプに基づく競合解決

次に、タイムスタンプに基づく基本的な競合検出の例を示します。

```
MAP swilkes.date_test, TARGET swilkes.date_test, &
REPERROR (21000, DISCARD), &
SQLEXEC (ID lookup, ON UPDATE, &
QUERY "select count(*) conflict from date_test where t_id = ? and &
t_timestamp > ?", &
PARAMS (p1 = t_id, p2 = t_timestamp), BEFOREFILTER, ERROR REPORT, &
TRACE ALL),&
FILTER (lookup.conflict = 0, ON UPDATE, RAISEERROR 21000);
```

注意　この例は、Replicat パラメータ・ファイルの一部であり、競合解決に関連する部分のみを示しています。

この例では、既存のターゲット・レコードの方が新しい場合に、レプリケートされたレコード更新 (UPDATE) が Replicat によって適用されないようにすることが目的です。最新の変更が優先されます。

競合検出コードでは、SQLEXEC 問合せを使用して、レプリケートされたレコードとキーが同じで、タイムスタンプが新しいすべてのレコードをターゲット表から抽出します。対象の列は、t_id キー列と t_timestamp タイムスタンプ列です。問合せは、UPDATE 操作に対してのみ、かつ出力をフィルタ・ルールに適用できるように FILTER 句の前で実行します。問合せは、lookup という論理名で実行します。

問合せの結果は conflict という変数に割り当てられ、lookup.conflict という表記法で FILTER 句への入力として使用されます。

FILTER 句の内容は、次のとおりです。

- 問合せの結果が 0 (ゼロ)、つまりターゲット・レコードの方が古い場合、フィルタは成功してレプリケートされたレコードがターゲットに適用されます。
- 問合せの結果が 1、つまりターゲット・レコードの方が新しい場合、フィルタは失敗し RAISEERROR によってユーザー定義のエラーが表示されます。このエラーによって、REPERROR で指定されたエラー処理レスポンスが起動されます。この例では、レプリケートされたレコードは破棄されます。

注意　複数の FILTER 文を使用して追加ルールを適用できます。これらは、パラメータ・ファイルに記述された順に実行されます。

この例は、次のとおり、例外 MAP 文と組み合せて使用することも可能です。

- REPERROR アクションを DISCARD ではなく EXCEPTION に変更します。
- 例外 MAP 文を使用して、失敗したレコード (操作) を例外表にマップします。

次に例を示します。

```
MAP swilkes.date_test, TARGET swilkes.date_test, &
REPERROR (21000, EXCEPTION), &
SQLEXEC (ID lookup, ON UPDATE, &
QUERY "select count(*) conflict from date_test where t_id = ? and &
t_timestamp > ?", &
PARAMS (p1 = t_id, p2 = t_timestamp), BEFOREFILTER, ERROR REPORT, &
TRACE ALL),&
FILTER (lookup.conflict = 0, ON UPDATE, RAISEERROR 21000);

MAP swilkes.date_test, TARGET swilkes.date_test_exc, EXCEPTIONSONLY, &
INSERTALLRECORDS, &
COLMAP (USEDEFAULTS, errtype = "UPDATE FILTER FAILED.");
```

2 番目の MAP 文が例外 MAP 文です。EXCEPTIONSONLY オプションによって、先行する MAP 文 (同じ swilkes.date_test ソース表をマップする文) によって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、例外 MAP 文がアクティブになります。INSERTALLRECORDS パラメータによって、Replicat は、エラーの原因である各操作を swilkes.date_test_exc 例外表の新しいレコードとして挿入します。これにより、すべての競合履歴が Point-in-Time スナップショットとして保持されます。

通常、このような例外表には、ソース表と同じ列に加え、コンテキスト情報を取得する追加列が含まれます。この例では、追加列 errtype にエラーの原因 (フィルタの失敗) が反映されます。例外表の作成の詳細は、92 ページを参照してください。

正味の変更値に基づく競合解決

次に、アプリケーションが UPDATE 文を使用して列値を減分する場合に、両方のシステムで正確な在庫数量を維持する方法の例を示します。更新から実際の値をレプリケートするかわりに、変更前イメージを変更後イメージと比較して、その差分をレプリケートします。

この例で、ボウル (Bowl) 品目の在庫数は 10 です。2 人の顧客が別々のデータベースにログインして同じ品目を注文し、1 人が 3 個、別の 1 人が 5 個購入する場合、実際の在庫には合計で 2 個残るので、2 人の注文は成功するはずです。

各システムでの在庫の更新は次のようになります。

システム A (3 個のボウルを購入):

```
Update inventory set quantity = 7 where item = 'Bowl' and quantity = 10;
```

システム B (5 個のボウルを購入):

```
Update inventory set quantity = 5 where item = 'Bowl' and quantity = 10;
```

システム A のトランザクションがシステム B に適用されると、システム B の数量の変更前イメージは 10 である必要がありますが、実際には 5 であるため、失敗します。逆に、システム B トランザクションがシステム A に適用されると、変更前イメージは 10 である必要がありますが、実際には 7 であるため、失敗します。

この競合を解決するには、実際の数を使用するのではなく、変更のソースで在庫の正味の変更を計算し、それをレプリケートされたトランザクションでターゲット表に適用します。式は次のとおりです。

```
Update inventory set quantity = (before image  after image) where item = 'Bowl';
```

システム A への適用 :

```
Update inventory set quantity = (5  3) where item = 'Bowl';
```

システム B への適用 :

```
Update inventory set quantity = (7  5) where item = 'Bowl';
```

これらの文が両方とも成功すると、ボウルの在庫数は両方の場所で 2 (正しい数量) になります。

第 9 章

Oracle GoldenGate のセキュリティの構成

セキュリティ・オプションの概要

次のセキュリティ機能を使用して、Oracle GoldenGate 環境および処理対象のデータを保護できます。

表 4　Oracle GoldenGate のセキュリティ・オプション

セキュリティ機能	説明
暗号化	次のものを暗号化および復号化するオプションを使用できます。 ◆ 抽出ファイルまたは証跡のデータ ◆ データベース・パスワード ◆ TCP/IP を通じて送信されるデータ
コマンド・セキュリティ	GGSCI を通じて Oracle GoldenGate コマンドにアクセスするためのユーザーレベルの権限を設定します。
接続セキュリティ	ソース・システムではなくターゲット・システムからの接続の確立を許可します。ターゲットが内部ファイアウォール内にある信頼できるネットワーク・ゾーン内に存在する場合に使用します。

暗号化の使用

この項では、次のものを暗号化および復号化する方法について説明します。

- Oracle GoldenGate によって処理されるデータを保持する証跡または抽出ファイル
- データベース・パスワード
- TCP/IP を通じて送信されるデータ

データを暗号化する方法

次の暗号化方法が使用されます。

- 証跡または抽出ファイルを暗号化する場合、Oracle GoldenGate では、256 鍵バイト置換が使用されます。これらのファイルに書き込まれるすべてのレコードは、データ・リンク経由とファイル内部の両方で暗号化されます。

- TCP/IP を通じて送信されるデータベース・パスワードまたはデータを暗号化する場合、Oracle GoldenGate では、Blowfish 暗号化が使用されます。Blowfish は、DES または IDEA の簡易な代替方式として使用できる対称ブロック暗号です。Oracle GoldenGate による Blowfish の実装では、32 ビットから 128 ビットの可変長の鍵を使用します。Blowfish 暗号化は、Oracle GoldenGate の証跡暗号化と組み合せることが可能です。

証跡または抽出ファイルの暗号化

任意のローカルまたはリモートの証跡やファイルに含まれるデータを暗号化できます。

注意 **(DB2 on z/OS) FORMATASCII を使用して ASCII 形式でファイルにデータを書き込む場合、この機能は使用できません。証跡またはファイルには、デフォルトの正規形式で書き込む必要があります。**

証跡または抽出ファイルを暗号化する手順

1. Extract のパラメータ・ファイルで、暗号化するすべての証跡またはファイルの前に次のパラメータをリストします。このパラメータの各インスタンスの後に、複数の証跡またはファイルをリストできます。

   ```
   ENCRYPTTRAIL
   ```

2. Extract のパラメータ・ファイルにリストされているファイルまたは証跡の暗号化を無効にするには、次のパラメータを各エントリの前に記載します。

   ```
   NOENCRYPTTRAIL
   ```

3. Replicat のパラメータ・ファイルで、Replicat がデータを復号化して処理できるように次のパラメータを記載します。

   ```
   DECRYPTTRAIL
   ```

   Extract データ・ポンプで DECRYPTTRAIL を使用して、列マッピング、フィルタリング、変換などのためにデータを復号化することもできます。その後は、後続の証跡またはファイルのためにその復号化を維持するか、それらのファイルにデータが書き込まれる前に ENCRYPTTRAIL を使用して再度暗号化することができます。

データベース・ユーザー・パスワードの暗号化

Oracle GoldenGate を使用して次のデータベース・パスワードを暗号化できます。

- ソース・データベースやターゲット・データベースにログインするために Extract や Replicat などのプロセスによって使用されるデータベース・パスワード。(Oracle GoldenGate プロセスに対してデータベース・ログインを必要としないデータベース・タイプもあります。)
- Oracle ASM ユーザー用のデータベース・パスワード。

データベース・ユーザー・パスワードを暗号化する手順

1. GGSCI を実行して ENCRYPT PASSWORD コマンドを発行し、暗号化パスワードを生成します。このコマンドには次のオプションがあります。

   - オプションが指定されないデフォルトの ENCRYPT PASSWORD コマンドでは、Oracle GoldenGate によってランダムに生成されるデフォルトの鍵を使用して暗号化パスワードが生成されます。

     ```
     ENCRYPT PASSWORD <password>
     ```

- ENCRYPTKEY <keyname> オプションが指定された ENCRYPT PASSWORD では、ENCKEYS 参照ファイルに含まれるユーザー定義の鍵を使用して暗号化パスワードが生成されます。

  ```
  ENCRYPT PASSWORD <password> ENCRYPTKEY <keyname>
  ```

  <keyname> には、ローカルの ENCKEYS ファイルの記載に従って、使用する鍵の論理名を指定します。このオプションを使用するには、最初に鍵を生成し、ローカル・システムで ENCKEYS ファイルを作成して、生成された鍵に対応するエントリをそのファイルに作成します。手順については、102 ページの「暗号化鍵の生成」を参照してください。

暗号化パスワードは、ENCRYPT PASSWORD コマンドの実行時に画面に出力されます。

2. 暗号化パスワードをコピーして、Oracle GoldenGate の適切なパラメータ文に貼り付けます (表 5 を参照)。

   条件 :

   - <user> は、Oracle GoldenGate プロセスまたは (Oracle のみ) ホスト文字列のデータベース・ユーザー名です。Oracle ASM の場合、ユーザー名は SYS である必要があります。
   - <encrypted_password> は、ENCRYPT PASSWORD コマンドの結果からコピーした暗号化パスワードです。
   - ENCRYPTKEY DEFAULT は、パスワードが ENCRYPTKEY オプションなしで ENCRYPT PASSWORD を使用して暗号化された場合に必要です。
   - ENCRYPTKEY <keyname> は、パスワードが ENCRYPTKEY <keyname> オプション付きで ENCRYPT PASSWORD を使用して暗号化された場合に必要です。ENCKEYS 参照ファイルの記載に従って鍵の論理名を指定します。

表 5　Oracle GoldenGate パラメータ・ファイルでの暗号化パスワードの指定

パスワード暗号化の対象	使用するパラメータ
Oracle GoldenGate データベース・ユーザー	`USERID <user>, PASSWORD <encrypted_password>, & ENCRYPTKEY {DEFAULT | <keyname>}`
Oracle ASM インスタンスの Oracle GoldenGate ユーザー	`TRANLOGOPTIONS ASMUSER SYS@<ASM_instance_name>, & ASMPASSWORD <encrypted_password>, & ENCRYPTKEY {DEFAULT | <keyname>}`

TCP/IP を通じて送信されるデータの暗号化

取得データは、Oracle GoldenGate によって TCP/IP ネットワークを通じてターゲット・システムに送信される前に暗号化できます。データは、ターゲット・システムで Oracle GoldenGate の証跡に書き込まれる前に Oracle GoldenGate によって復号化されます (証跡暗号化が指定されていない場合)。デフォルトでは、ネットワークを通じて送信されるデータは暗号化されません。

TCP/IP を通じて送信されるデータを暗号化する手順

1. ソース・システムで、1 つ以上の暗号化鍵を生成し、ENCKEYS ファイルを作成します。102 ページの「暗号化鍵の生成」を参照してください。

2. 作成した ENCKEYS ファイルを、すべてのターゲット・システムにある Oracle GoldenGate のインストール・ディレクトリにコピーします。ソースの ENCKEYS ファイルに含まれる鍵の名前および値は、ターゲットの ENCKEYS ファイルに含まれる名前および値と一致している必要があります。一致していないと、データ交換に失敗するため、Extract および Collector は次のメッセージとともに中断します。

   ```
   GGS error 118  TCP/IP Server with invalid data.
   ```

3. これが標準 Extract グループであるかパッシブ Extract グループであるかに応じて (106 ページを参照)、RMTHOST パラメータまたは RMTHOSTOPTIONS パラメータの ENCRYPT オプションを使用して、暗号化のタイプおよび鍵の論理名を次のように指定します。

   ```
   ENCRYPT BLOWFISH, KEYNAME <keyname>
   ```

   条件 :

   - BLOWFISH では、Blowfish 暗号化を指定します。
   - <keyname> は、ENCKEYS ファイルの記載に基づいた、使用する暗号化鍵の論理名です。

   例 :

   ```
   RMTHOST sys1, MGRPORT 7840, ENCRYPT BLOWFISH, KEYNAME superkey
   RMTHOSTOPTIONS ENCRYPT BLOWFISH, KEYNAME superkey
   ```

4. 静的 Collector および Blowfish 暗号化を使用する場合、Collector の起動文字列に次の追加パラメータを挿入します。

   ```
   -KEYNAME <name>
   -ENCRYPT BLOWFISH
   ```

   条件 :

   - KEYNAME <name> では、鍵の名前を指定します。
   - ENCRYPT BLOWFISH では、Blowfish 暗号化を指定します。

   Collector は、これらのパラメータを RMTHOST の KEYNAME オプションおよび ENCRYPT オプションで指定されているパラメータと照合します。

暗号化鍵の生成

次を実行する場合、1 つ以上の暗号化鍵と、2 つの ENCKEYS 参照ファイル (1 つはソースに、1 つはターゲットに配置) を作成する必要があります。

- TCP/IP を通じて送信されるデータの暗号化
- *ユーザー定義*の鍵を使用したデータベース・パスワードの暗号化

この手順は、次の場合には*不要*です。

- Oracle GoldenGate によって生成されたデフォルトの鍵を使用してデータベース・パスワードを暗号化する場合
- 証跡または抽出ファイルを暗号化する場合

独自の鍵を定義するか、Oracle GoldenGate の KEYGEN ユーティリティを実行してランダムに鍵を作成できます。

独自の鍵を定義する手順

- 鍵の名前には、空白または引用符のない 1 から 24 個の英数字文字列を指定できます。
- 鍵の値には、最大128ビット(16バイト)の、引用符付きの英数字文字列("Dailykey"など)または0x接頭辞付きの 16 進文字列 (0x420E61BE7002D63560929CCA17A4E1FB など) を指定できます。

KEYGEN を使用して鍵を生成する手順

ソース・システムでディレクトリを Oracle GoldenGate のホーム・ディレクトリに変更し、次のシェル・コマンドを発行します。必要に応じて、複数の鍵を作成できます。鍵の値は画面に戻されます。

```
KEYGEN <key length> <n>
```

条件 :

- <key length> は、最大 128 ビットの暗号化鍵の長さです。
- <n> は、生成する鍵の数を示します。

例 :

```
KEYGEN 128 4
```

Oracle GoldenGate で使用するために鍵を格納する手順

1. ソース・システムで、新しい ASCII テキスト・ファイルを開きます。
2. 生成した鍵ごとに、論理名に続けて鍵の値を入力します。行を分けて複数の鍵の定義を配置します。鍵の名前または値は、引用符で囲まないでください。囲むとテキストとして解釈されます。次のサンプルの ENCKEYS ファイルを参考用として使用してください。

```
## Encryption keys

## Key name         Key value
superkey            0x420E61BE7002D63560929CCA17A4E1FB
secretkey           0x027742185BBF232D7C664A5E1A76B040
superkey1           0x42DACD1B0E94539763C6699D3AE8E200
superkey2           0x0343AD757A50A08E7F9A17313DBAB045
superkey3           0x43AC8DCE660CED861B6DC4C6408C7E8A
```

3. このファイルを Oracle GoldenGate のインストール・ディレクトリに ENCKEYS という名前 (拡張子なし) で保存します。名前は大文字にする必要があります。
4. ENCKEYS ファイルを、ターゲットにある Oracle GoldenGate のインストール・ディレクトリにコピーします。ソースの ENCKEYS ファイルに含まれる鍵の名前および値は、ターゲットの ENCKEYS ファイルに含まれる名前および値と一致している必要があります。一致していないと、データ交換に失敗するため、Extract および Collector は次のメッセージとともに中断します。

```
GGS error 118  TCP/IP Server with invalid data.
```

コマンド・セキュリティの使用

Oracle GoldenGate でコマンド・セキュリティを確立して、Oracle GoldenGate 機能へのアクセスを許可するユーザーを制御できます。たとえば、特定のユーザーに INFO コマンドおよび STATUS コマンドの発行を許可する一方で、START コマンドおよび STOP コマンドの使用を拒否できます。セキュリティ・レベルは、オペレーティング・システムのユーザー・グループによって定義されます。

Oracle GoldenGate コマンドに対するセキュリティを実装するには、Oracle GoldenGate ディレクトリに CMDSEC ファイルを作成します。このファイルがない場合、すべてのユーザーにすべての Oracle GoldenGate コマンドに対するアクセス権が付与されます。

コマンド・セキュリティを実装する手順

1. 新しい ASCII テキスト・ファイルを開きます。
2. 次の構文および 105 ページの例を参照して、制限するコマンドごとに 1 つ以上のセキュリティ・ルールを作成します (1 行に 1 つのルール)。ルールは、個別性の最も高いもの (ワイルドカードのないもの) から最も低いものへと指定します。セキュリティ・ルールは、CMDSEC ファイルの一番上から下に向かって処理されます。条件が満たされる最初のルールは、アクセスを許可するかどうかを決定するルールです。

   次の各構成要素を空白またはタブ文字で区切ります。

   ```
   <command name> <command object> <OS group> <OS user> <YES | NO>
   ```

   条件 :

   - <command name> は、GGSCI コマンド名またはワイルドカードです (START、STOP、* など)。
   - <command object> は、任意の GGSCI コマンド・オブジェクトまたはワイルドカードです (EXTRACT、REPLICAT、MANAGER など)。
   - <OS group> は、Windows または UNIX のユーザー・グループ名です。UNIX システムでは、グループ名のかわりに数値のグループ ID を指定できます。ワイルドカードを使用すると、すべてのグループを指定できます。
   - <OS user> は、Windows または UNIX のユーザー名です。UNIX システムでは、ユーザー名のかわりに数値のユーザー ID を指定できます。ワイルドカードを使用すると、すべてのユーザーを指定できます。
   - <YES | NO> では、コマンドに対するアクセス権を付与するか、または禁止するかを指定します。

3. このファイルを Oracle GoldenGate のホーム・ディレクトリに CMDSEC という名前 (UNIX システムでは大文字を使用) で保存します。

次の例は、UNIX システムでの CMDSEC ファイルの適切な実装を示しています。

表 6　サンプルの CMDSEC ファイルとその説明

ファイルの内容	説明
`#GG command security`	コメント行
`STATUS REPLICAT * Smith NO`	STATUS REPLICAT がユーザー Smith に対して拒否されます。
`STATUS * dpt1 * YES`	先行するルールを除き、すべての STATUS コマンドが dpt1 のすべてのユーザーに対して許可されます。
`START REPLICAT root * YES`	START REPLICAT が、root グループのすべてのメンバーに対して許可されます。
`START REPLICAT * * NO`	先行するルールを除き、START REPLICAT がすべてのユーザーに対して拒否されます。
`* EXTRACT 200 * NO`	すべての EXTRACT コマンドが、ID が 200 のすべてのグループに対して拒否されます。
`* * root root YES`	root ユーザーに対して任意のコマンドが許可されます。
`* * * * NO`	すべてのユーザーに対してすべてのコマンドが拒否されます。この行によって、先行するルールで明示的にアクセスを許可または拒否していない他のすべてのユーザーのセキュリティに対応します。そうしないと、先行する明示的な許可または拒否を除き、すべてのユーザーに対してすべてのコマンドが許可されます。

次の*不適切な*例は、CMDSEC ファイルの作成時に避ける必要のある設定を示しています。

表 7　不適切な CMDSEC のエントリ

ファイルの内容	説明
`STOP * dpt2 * NO`	すべての STOP コマンドが、グループ dpt2 のすべてのユーザーに対して拒否されます。
`STOP * * Chen YES`	すべての STOP コマンドが Chen に対して許可されます。

表 7 のエントリ順序では、論理エラーが発生します。1 番目のルール (1 行目) によって、すべての STOP コマンドがグループ dpt2 のすべてのメンバーに対して拒否されます。2 番目のルール (2 行目) によって、すべての STOP コマンドがユーザー Chen に対して許可されます。このとき、Chen は、dpt2 グループのメンバーであるため、このコマンドを発行する権限を付与される必要があっても、2 番目のルールによってすべての STOP コマンドに対するアクセスを拒否されます。

このセキュリティ・ルールを適切に構成する方法は、ユーザー固有のルールを、それよりも一般的な 1 つ以上のルールの前に設定することです。したがって、エラーを修正するには、2 つの STOP ルールの順序を逆にします。

CMDSEC ファイルの保護

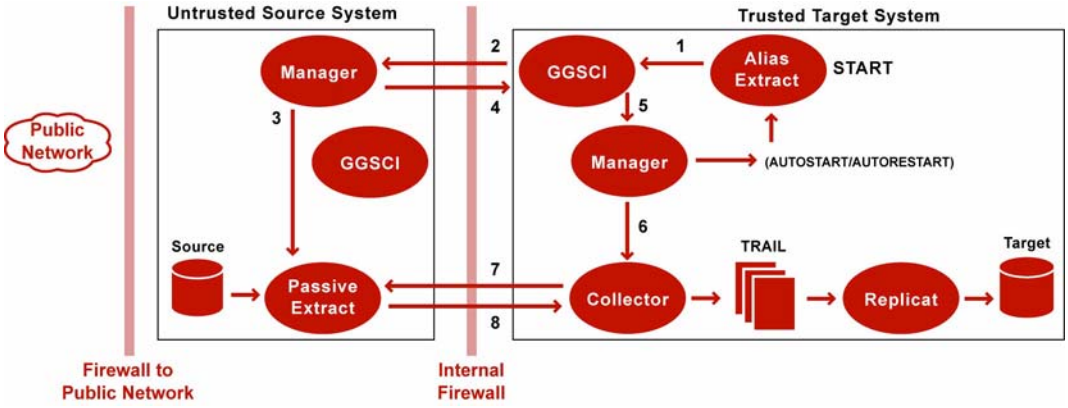
CMDSEC ファイルは、セキュリティの根源であるため、保護する必要があります。読取りアクセス権は必要に応じて付与できますが、書込みおよび削除アクセス権は、Oracle GoldenGate 管理者以外のすべてのユーザーに対して Oracle GoldenGate で拒否することをお薦めします。

ターゲット・システムからの接続開始の使用

信頼できるイントラネット・ゾーンにターゲット・システムが存在する場合、信頼度がより低いゾーンに存在するソース・システムから接続を開始すると (Oracle GoldenGate の標準的な方法)、セキュリティ・ポリシーに違反する可能性があります。また、信頼度が低いゾーンに存在するシステムに、信頼できるゾーンのシステムのポートまたは IP アドレスに関する情報 (Oracle GoldenGate の Extract パラメータ・ファイルに通常記載されている情報など) が含まれる場合も、セキュリティ・ポリシーに違反する可能性があります。

この種のイントラネット構成では、*パッシブ / エイリアス Extract* 構成を使用できます。接続は、*エイリアス Extract* グループによって、信頼できるゾーン内部のターゲット・システムから開始されます。このグループは、ソース・システムの標準 Extract グループに対するエイリアスとして機能し、この場合は*パッシブ Extract* と呼ばれます。2 つのシステム間で接続が確立されると、データはパッシブ Extract グループによって通常どおり処理され、ネットワークを通じて転送されます。

図 16 信頼できるネットワーク・ゾーンからの接続の開始



1. Oracle GoldenGate ユーザーが信頼できるシステムでエイリアス Extract を起動するか、AUTOSTART または AUTORESTART パラメータによってエイリアス Extract が自動的に起動されます。
2. 信頼できるシステムの GGSCI は、信頼度の低いシステムの Manager にメッセージを送信して、関連付けられたパッシブ Extract を起動します。信頼できるシステムの Manager のホスト名 (または IP アドレス) とポート番号が、信頼度の低いシステムに送信されます。
3. 信頼度の低いシステムで、Manager は開いているポート (Manager の DYNAMICPORTLIST パラメータのルールに準拠) を検出し、指定のポートでリスニングを行うパッシブ Extract を起動します。
4. 信頼度の低いシステムの Manager は、そのポートを信頼できるシステムの GGSCI に戻します。
5. 信頼できるシステムの GGSCI は、同じシステムの Manager にリクエストを送信し、同じシステムの Collector プロセスを起動します。

6. ターゲットの Manager は、Collector プロセスを起動して、そのプロセスに信頼度の低いシステムで Extract がリスニングしているポート番号を渡します。
7. 信頼できるシステムの Collector は、信頼度の低いシステムのパッシブ Extract に対する接続をオープンします。
8. データは、ネットワークを通じてパッシブ Extract からターゲットの Collector に送信され、Replicat で処理するために通常どおり証跡に書き込まれます。

パッシブ Extract グループの構成

信頼度の低いソース・システムのパッシブ Extract グループは、ネットワークを通じたデータの送信を担当する Extract グループの種類に応じて、次のいずれかになります。

- トランザクション・ログを読み取ってそのデータをターゲットに送信する単独 Extract グループ。
- プライマリ Extract から提供されるローカル証跡を読み取ってそのデータをターゲットに送信するデータ・ポンプ Extract グループ。この場合、プライマリ Extract(単なるデータ・ポンプ) に対する特別な構成要件はありません。

Extract グループをパッシブ・モードで作成するには、標準の ADD EXTRACT コマンドおよびオプションを使用しますが、他のコマンド・オプションに対して任意の位置に PASSIVE キーワードを追加します。例 :

```
ADD EXTRACT fin, TRANLOG, BEGIN NOW, PASSIVE, DESC "passive Extract"
ADD EXTRACT fin, PASSIVE, TRANLOG, BEGIN NOW, DESC "passive Extract"
```

パッシブ Extract グループのパラメータを構成するには、通常の方法でパラメータ・ファイルを作成します。ただし、次の点が異なります。

- RMTHOST パラメータは*除外*します (通常は、このパラメータでターゲットの Manager のホストおよびポート情報を指定します)。
- オプションの RMTHOSTOPTIONS パラメータを使用して、圧縮および暗号化のルールを指定します。

```
RMTHOSTOPTIONS
[, COMPRESS]
[, COMPRESSTHRESHOLD]
[, ENCRYPT {NONE | BLOWFISH}]
[, KEYNAME <keyname>]
[, PARAMS <collector parameters>]
[, TCPBUFSIZE <bytes>]
[, TCPFLUSHBYTES <bytes>]
```

ENCRYPT オプションと KEYNAME オプションの使用方法は、この章の 101 ページから記載されています。これらのオプションによって、TCP/IP を通じて送信されるデータを Blowfish 暗号化を使用して暗号化します。RMTHOSTOPTIONS の残りのオプションの詳細は、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』を参照してください。

Extract グループの構成の詳細は、第 12 章を参照してください。

エイリアス Extract グループの構成

信頼できるターゲットのエイリアス Extract グループは、データ処理アクティビティを実行しません。その唯一の目的は、信頼度の低いソースに対する接続の開始および終了です。この役割において、エイリアス Extract グループは、パラメータ・ファイルを使用せず、処理チェックポイントも書き込みません。チェックポイント・ファイルの用途は、パッシブ Extract グループが実行されているかどうかを判別することと、リモート接続に必要な情報を記録することに限定されます。

エイリアス・モードで Extract グループを作成するには、次のオプションのみを指定して ADD EXTRACT コマンドを使用します。

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

RMTHOST の指定によって、このグループがエイリアス Extract として識別され、情報がチェックポイント・ファイルに書き込まれます。<host name> オプションと <IP address> オプションでは、ソース・システムの名前または IP アドレスを指定します。MGRPORT では、Manager が稼働しているソース・システムのポートを指定します。

エイリアス Extract の名前は、パッシブ Extract と同じ名前にすることも、異なる名前にすることもできます。名前が異なる場合、オプションの RMTNAME を使用して、パッシブ Extract の名前を指定します。RMTNAME が使用されない場合、Oracle GoldenGate では、名前が同一であると仮定され、接続の確立時に使用されるエイリアス Extract のチェックポイント・ファイルにその名前が書き込まれます。

TCP/IP 接続のエラー処理は、ターゲット・システムの TCPERRS ファイルによって指示されます。このファイルのエラーに対するレスポンス値は、RETRY に設定することをお薦めします。デフォルトは ABEND です。このファイルには、再試行の回数および各試行間の遅延を設定するオプションも含まれます。詳細は、113 ページを参照してください。

パッシブ・プロセスとエイリアス・プロセスの起動および停止

パッシブ / エイリアス Extract 構成で Oracle GoldenGate の抽出を開始または停止するには、ターゲットの GGSCI でエイリアス Extract グループを起動または停止します。

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

または

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

コマンドは、ソース・システムに送信され、パッシブ Extract グループが起動または停止します。これらのコマンドをパッシブ Extract グループに対して直接発行しないでください。KILL EXTRACT コマンドは、パッシブ Extract グループに対して直接発行できます。

Manager パラメータの AUTOSTART および AUTORESTART を使用して自動的にプロセスを起動または再起動する場合、ソース・システムではなくターゲット・システムで使用してください。最初にエイリアス Extract が起動され、次にパッシブ Extract に起動コマンドが送信されます。

抽出アクティビティの管理

抽出処理が開始されたら、ソース・システムの GGSCI からパッシブ Extract グループに対してコマンドを発行することで、通常どおりその処理を管理および監視できます。INFO や VIEW REPORT などの GGSCI の標準監視コマンドを、ソース・システムまたはターゲット・システムから発行できます。エイリアス Extract グループに対して発行された監視コマンドは、パッシブ Extract グループに転送されます。コマンドでは、エイリアス Extract グループの名前がパッシブ Extract グループの名前で置き換えられます。たとえば、INFO EXTRACT alias は、INFO EXTRACT passive になります。コマンドの結果は、コマンドが発行されたシステムに表示されます。

その他の考慮事項

パッシブ / エイリアス Extract 構成を使用する場合、次のルールが適用されます。

- この構成では、Extract は 1 つのターゲット・システムにのみ書込みを行うことができます。
- この構成は、通常の方法で (THREADS オプションを使用して REDO スレッドの数を指定して)Extract グループを作成することで、Oracle RAC インストール環境で使用できます。
- この構成は、バッチ実行のコマンドラインで Extract が起動される場合に使用できます。135 ページの第 13 章を参照してください。
- ALTER EXTRACT コマンドは、エイリアス Extract に対して使用できません (このグループはデータ処理を実行しないため)。
- パッシブまたはエイリアス Extract グループに対して DELETE EXTRACT コマンドを使用するには、ローカル GGSCI からコマンドを発行します。
- Extract のパラメータ・ファイルの RMTTASK で指定され、一部の初期ロード方法で使用されるリモート・タスクは、この構成ではサポートされません。リモート・タスクでは、ソース・システムから接続を開始する必要があり、Extract と Replicat 間の直接接続が使用されます。

第 10 章

Oracle GoldenGate の処理エラーへの対処

Oracle GoldenGate のエラー処理の概要

Oracle GoldenGate では、次の構成要素にエラー処理オプションが提供されます。

- Extract
- Replicat
- TCP/IP

Extract エラーの処理

DML 操作が抽出される場合の Extract エラーを処理する特定のパラメータはありませんが、Extract には、予測される問題を防止するために使用できる多くのパラメータがあります。これらのパラメータによって、DML 操作の処理中に発生する可能性のある異常状態を処理します (たとえば、フェッチする行を特定できない場合の処理や、トランザクション・ログを使用できない場合の処理など)。次に、これらのパラメータの一部をリストします。完全なリストは、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』を参照してください。

- FETCHOPTIONS
- WARNLONGTRANS
- DBOPTIONS
- TRANLOGOPTIONS
- LFMMEMORY

DDL 操作に関連する抽出エラーを処理するには、DDLERROR パラメータを使用します。詳細は、175 ページを参照してください。

DML 操作中の Replicat エラーの処理

いずれかの DML 文の処理中に発生したエラーに対する Replicat のレスポンス方法を制御するには、Replicat のパラメータ・ファイルで REPERROR パラメータを使用します。REPERROR は、グローバル・パラメータとして、または MAP 文の一部として使用できます。ほとんどのエラーは DEFAULT オプションおよび DEFAULT2 オプションを使用してデフォルトの方法 (処理の中止など) で処理し、他のエラーは特定の方法で処理できます。

TRANSDISCARD と TRANSEXCEPTION 以外のオプションは、エラーを生成した個々のレコードのみに影響します。TRANSDISCARD と TRANSEXCEPTION は、エラーを生成したレコードを含むトランザクションのすべてのレコードに影響します。(ABEND オプションもトランザクション全体を対象としていますが、エラー処理は適用しません。)

REPERROR によるすべてのレスポンスを次に示します。

- ABEND: トランザクションをロールバックして処理を停止します。
- DISCARD: 廃棄ファイルにエラーを記録して処理を継続します。
- EXCEPTION: 例外処理のためにエラーを送信します (「例外としてのエラーの処理」を参照)。
- IGNORE: エラーを無視して処理を継続します。
- RETRYOP [MAXRETRIES <n>]: 操作を再試行します (オプションで最大試行回数を指定できます)。
- TRANSABORT [, MAXRETRIES <n>] [, DELAY[C]SECS <n>]: トランザクションを中断して最初の配置に戻します (オプションで最大試行回数とその間隔を指定できます)。
- RESET: 以前のすべての REPERROR ルールを削除し、デフォルトの ABEND に戻します。
- TRANSDISCARD: レプリケートされたソース・トランザクション内の操作 (コミットなど) が、エラー指定にリストされている Replicat エラーの原因となっている場合に、そのトランザクション全体を破棄します。このオプションは、ターゲットで整合性制約チェックが無効になっている場合に便利です。
- TRANSEXCEPTION: レプリケートされたソース・トランザクション内の操作 (コミットなど) が、エラー指定にリストされている Replicat エラーの原因となっている場合に、そのトランザクションのすべてのレコードに対し、例外マッピング文に従って例外マッピングを実行します。

例外としてのエラーの処理

Replicat エラーは、REPERROR パラメータに EXCEPTION オプションまたは TRANSEXCEPTION オプションを使用することで、例外として扱うことができます。その後で、これらの例外を、エラーの解決に使用できるエラー情報と組み合せて例外表にマップできます。MAP 文の次のオプションを使用して、例外を例外表に送信できます。

- EXCEPTIONSONLY (個別の例外 MAP 文が必要)
- MAPEXCEPTION (必要な MAP 文は 1 つのみ)

EXCEPTIONSONLY の使用

EXCEPTIONSONLY 句を使用できるのは、MAP 文の MAP 句および TARGET 句の内部でソース表とターゲット表の名前にワイルドカードが*使用されない*場合です。この方法では、ソース表に対して次のように 2 つの MAP 文を作成する必要があります。

- 1 番目の通常の MAP 文によってソース表を実際のターゲット表にマップします。
- 2番目の*例外MAP文*によってソース表を(ターゲット表ではなく)*例外表*にマップします。例外MAP文は、ソース表でエラーが発生した後にのみ実行されます。例外 MAP 文は、そのソース表が含まれる通常の MAP 文の直後に配置する必要があります。

MAPEXCEPTION の使用

MAPEXCEPTION を使用する必要があるのは、MAP 文の MAP 句および TARGET 句の内部でソース表とターゲット表の名前にワイルドカードが*使用される*場合です。MAPEXCEPTION 句は、通常の MAP 文 (ソース表をターゲット表にマップする文と同じ) に含めることができます。MAPEXCEPTION 句では、REPERROR 文に基づいて例外として扱われる操作を取得し、それらを例外表にマップします。この構成に必要な MAP 文は、2 つではなく 1 つだけです。

例外表について

例外表には、任意の方法 (エラーを処理したり、競合解決を開始するようにアプリケーションを構成するなど) で使用できるエラー情報が含まれます。例外表は、一般的に、通常のターゲット表と同じ列を使用し、データおよびエラーに関する追加情報を取得するために別の列を追加して作成されます。この情報を取得するには、COLMAP、Oracle GoldenGate 変換関数、および他の適切な Oracle GoldenGate オプションを使用します (たとえば、関数でエラー番号や環境情報を取得したり、COLMAP を使用して適切な列に情報をマップするなどです)。

例 1 EXCEPTIONSONLY

この例は、例外マッピング専用に EXCEPTIONS を使用する方法を示しています。REPERROR を EXCEPTIONSONLY および例外 MAP 文と組み合せて使用する方法がわかります。この例では、REPERROR に関連するパラメータのみを示し、このドキュメントのスペースを節約するためにそれ以外は省略してあります。

```
REPERROR (DEFAULT, EXCEPTION)
MAP ggs.equip_account, TARGET ggs.equip_account2, &
COLMAP (USEDEFAULTS);

MAP ggs.equip_account, TARGET ggs.equip_account_exception, &
EXCEPTIONSONLY, &
INSERTALLRECORDS &
COLMAP (USEDEFAULTS, &
DML_DATE = @DATENOW(), &
OPTYPE = @GETENV("LASTERR", "OPTYPE"), &
DBERRNUM = @GETENV("LASTERR", "DBERRNUM"), &
DBERRMSG = @GETENV("LASTERR", "DBERRMSG"));
```

この例では、REPERROR パラメータは DEFAULT エラー処理用に設定されています。EXCEPTION オプションによって、Replicat プロセスは例外を失敗した操作として扱い、処理を継続します。

次の 2 つの MAP 文があります。

- ソース表の ggs.equip_account をそのターゲット表の equip_account2 にマップする通常の MAP 文。
- 同じソース表を例外表の ggs.equip_account_exception にマップする例外 MAP 文。

この場合、表自体に含まれる同じ列に加え、次の 4 つの追加列が作成されます。

DML_DATE
OPTYPE
DBERRNUM
DBERRMSG

DML_DATE 列にデータを移入するため、@DATENOW 列変換関数を使用して失敗した操作の日時を取得し、結果を列にマップします。他の追加列にデータを移入するため、@GETENV 関数を使用して操作タイプ、データベース・エラー番号およびデータベース・エラー・メッセージを戻します。

例外 MAP 文の EXCEPTIONSONLY オプションによって、ソース表に対する操作が失敗した後にのみ文が実行されます。これによって、すべての操作が例外表に記録されることを防止します。

INSERTALLRECORDS パラメータによって、指定したソース表に対するすべての失敗した操作は、その操作タイプにかかわらず*挿入*として例外表に記録されます。

注意　　例外表に対して主キー制約または一意索引制約は指定できません。

例 2　MAPEXCEPTION

この例は、例外マッピングで MAPEXCEPTION を使用する方法を示しています。MAP 句および TARGET 句には、ワイルドカードを使用したソース表およびターゲット表の名前が含まれます。TRX で始まる名前の表を処理する際に発生する例外は、指定し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")
)
);
```

DDL 操作中の Replicat エラーの処理

ターゲットでの DDL 操作中に発生したエラーに対する Replicat のレスポンス方法を制御するには、Replicat のパラメータ・ファイルで DDLERROR パラメータを使用します。詳細は、175 ページの「Extract の DDL 処理エラーへの対処」を参照してください。

TCP/IP エラーの処理

TCP/IP エラーに対するレスポンス指示を提供するには、TCPERRS ファイルを使用します。このファイルは、Oracle GoldenGate ディレクトリにあります。

図 17　TCPERRS ファイル

```
# TCP/IP error handling parameters
# Default error response is abend
#
# Error          Response   Delay(csecs)  Max Retries

ECONNABORTED     RETRY      1000          10
ECONNREFUSED     RETRY      1000          12
ECONNRESET       RETRY      500           10
ENETDOWN         RETRY      3000          50
ENETRESET        RETRY      1000          10
ENOBUFS          RETRY      100           60
```

```
ENOTCONN        RETRY     100          10
EPIPE           RETRY     500          10
ESHUTDOWN       RETRY     1000         10
ETIMEDOUT       RETRY     1000         10
NODYNPORTS      RETRY     100          10
```

TCPERRS ファイルには、基本的なエラーに対するデフォルトのレスポンスが含まれます。指示を変更するか、新規エラー用の指示を追加するには、テキスト・エディタでファイルを開き、表 8 に示されている列の値を変更します。

表 8　TCPERRS の列

列	説明
Error	レスポンスを定義する TCP/IP エラーを指定します。
Response	定義されたエラーの後に Oracle GoldenGate で再接続を試行するかどうかを制御します。有効な値は、RETRY または ABEND です。
Delay	再接続を試行するまでに Oracle GoldenGate で待機する時間を制御します。
Max Retries	強制終了するまでに Oracle GoldenGate で再接続を試行する回数を制御します。

レスポンスが TCPERRS ファイルに明示的に定義されていない場合、Oracle GoldenGate は異常終了によって TCP/IP エラーにレスポンスします。

更新されたエラー・メッセージの管理

Oracle GoldenGate プロセスによって生成されるエラー、情報および警告メッセージは、Oracle GoldenGate のインストール・ディレクトリにある ggmessage.dat というデータ・ファイルに格納されます。このファイルのバージョンは、プロセスの起動時にチェックされます。プロセスが動作するためには、このバージョンがプロセスのバージョンと同じである必要があります。

Oracle GoldenGate エラーの解決

Oracle GoldenGate エラーの解決の詳細は、『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。

第 11 章

データ定義ファイルの作成

データ定義ファイルの概要

データ定義ファイルには、レプリケート対象データの形式 (表名、列名、データ型、データ長、オフセットなど) に関する情報が含まれています。データ定義ファイルによって、Oracle GoldenGate は、異なる種類のデータベース間でデータを移動する場合に、データの形式を両者間で変換できます。変換を実行するには、両方のデータセットの定義が Oracle GoldenGate に認識されている必要があります。Oracle GoldenGate プロセスは、1 つの定義セットを取得するために、そのプロセスにとってローカルなデータベースに問い合せますが、リモート・データベースの定義セットを取得するには定義ファイルに依存する必要があります。たとえば、Replicat プロセスは、ターゲット・データベースに問い合せますが、ソース・データベースのメタデータを取得するには定義ファイルに依存します。

データ定義ファイルを使用する場合

データ定義の異なるソース表とターゲット表 (Oracl のソース表と Microsoft SQL Server のターゲット表など) を同期する場合には、常にデータ定義ファイルが必要です。ソースとターゲットのデータベース・システムが同じであっても、対応するソース列とターゲット列が「表が同一とみなされるためのルール」のガイドラインに*準拠していない*場合、ソース表とターゲット表は異なる可能性があります。

表が同一とみなされるためのルール

ソース列とターゲット列の構造が同一であるためには、次の要件を満たす必要があります。

- 列名が同一であること (該当する場合は大 / 小文字の区別も含む)。
- データ型が同一であること。
- 列の長さが同一であること。
- 文字の列の列長さセマンティクス (バイトか文字か) が同じであること。
- 各表の順序が同じであること。

たとえば、ソースの Oracle データベースのセマンティクスがバイトとして構成されているのに対し、ターゲットのセマンティクスが文字として構成されている場合 (またはその逆)、表の構造が同一であってもソース定義ファイルが必要です。また別の例として、次に示すように、名前列の順序以外は同一であるソース表とターゲット表の組合せでも、ソース定義ファイルが必要です。

ソース

```
CREATE TABLE emp
( employee_id    NUMBER(6)
, first_name     VARCHAR2(20)
, last_name      VARCHAR2(25)
, phone_number   VARCHAR2(20)
, hire_date      DATE  DEFAULT SYSDATE
```

ターゲット

```
CREATE TABLE emp
( employee_id    NUMBER(6)
, last_name      VARCHAR2(25)
, first_name     VARCHAR2(20)
, phone_number   VARCHAR2(20)
, hire_date      DATE  DEFAULT SYSDATE
```

注意 Oracle の順序に対するデータ定義ファイルは作成しないでください。このファイルは不要であり、DEFGEN ではサポートされません。

定義ファイルのタイプ

- Oracle GoldenGate を構成してターゲット・システムで列マッピングまたは変換を実行する場合、*ソース定義ファイル*が必要です。ソース定義ファイルには、ソース表の定義が含まれています。このファイルをターゲット・システムに転送します。Replicat は、これらの定義と必要なターゲット定義を参照して変換を実行します。
- Oracle GoldenGate を構成してソース・システムで列マッピングまたは変換を実行する場合、*ターゲット定義ファイル*が必要です。ターゲット定義ファイルには、ターゲット表の定義が含まれています。このファイルをソース・システムに転送します。プライマリ Extract またはデータ・ポンプは、これらの定義と必要なソース定義を参照して変換を実行します。
- NonStop Server ターゲットに他のタイプのデータベースからレプリケートを行う場合、ソースの Windows または UNIX システムでマッピングおよび変換を実行する必要があります。NonStop 用の Replicat は、2 つの部分からなる表名およびデータ型を、NonStop プラットフォームで使用される 3 つの部分からなる名前に変換しないため、Extract では、NonStop の名前およびターゲットのデータ型を使用して証跡データをフォーマットする必要があります。したがって、この場合は変換をサポートするためにターゲットのデータ定義ファイルが常に必要です。
- Oracle GoldenGate を構成して、ソース・データベースもターゲット・データベースも*含まない*中間システムで列マッピングまたは変換を実行する場合、そのシステムにソース定義ファイル*およびターゲット定義ファイル*を用意する必要があります。

定義テンプレートを使用する場合

初期構成および起動後に Oracle GoldenGate 環境に表を追加する予定の場合、定義テンプレートを使用します。すべての表がまったく同じ構造である場合、複数の表に対応する定義テンプレートを生成できます。たとえば、顧客ごとに個別の表があり、各表のすべての列、列の順序およびデータ型が同一であれば、テンプレートを使用できます。

テンプレートを使用することで、新しい表ごとに新規定義ファイルを生成する必要がなくなり、新規定義をアクティブ化するために Oracle GoldenGate プロセスを停止して起動する作業を回避できます。これによって、新しい表を頻繁に追加する場合のメンテナンス作業が削減されます。

テンプレートは次のように操作します。

- マスター・データ定義ファイルを作成する際に定義テンプレートを指定します。
- Oracle GoldenGate 構成にすでに存在するものと同じ構造を持つ新しい表を追加する場合は、常に TABLE または MAP パラメータの DEF または TARGETDEF オプションを使用してテンプレートにその新しい表をマップできます。

注意　マスター定義ファイルでテンプレートを使用しない場合、Oracle GoldenGate 構成に新しい表を追加するたびに常に定義ファイルを生成できます。それぞれの新規定義ファイルの内容を単純にコピーして、既存のマスター定義ファイルに追加できます。

データ定義ファイルの構成

Oracle GoldenGate を構成してデータ定義ファイルを使用するには、次の操作を実行します。

- DEFGEN ユーティリティのパラメータ・ファイルを作成します。
- DEFGEN ユーティリティを実行してファイルを生成します。
- Oracle GoldenGate プロセスで定義ファイルを参照するように設定します。

DEFGEN のパラメータ・ファイルを作成する手順

ソース定義ファイルを作成するには、ソース・システムで次の手順を実行します。ターゲット定義ファイルを作成するには、この手順をターゲット・システムで実行します。

1. Oracle GoldenGate ディレクトリから GGSCI を実行します。
2. GGSCI で、次のコマンドを発行して DEFGEN のパラメータ・ファイルを作成します。

   ```
   EDIT PARAMS DEFGEN
   ```

3. 表 9 にリストされている順序でパラメータを入力します。パラメータ文ごとに新規行を開始します。

表 9　DEFGEN のパラメータ

パラメータ	説明
`DEFSFILE <full_pathname>`	DEFGEN の出力となるデータ定義ファイルの相対名または完全修飾名を指定します。

`[{SOURCEDB | TARGETDB} <dsn>,]`
`[USERID <user>[, PASSWORD <password>]]`
◆ SOURCEDB | TARGETDB では、データソース名を指定します (接続情報の一部として必要な場合)。Oracle では必要ありません。
◆ データベース認証を指定する必要がある場合、USERID および PASSWORD は必須です。SQL/MX または DB2 では、パスワードは必要ありません。Oracle では、次のようなホスト文字列を含めることができます。
`USERID ggs@ora1.ora, PASSWORD ggs123`
	データベース接続情報を指定します。
	SOURCEDB オプションと USERID オプションの詳細は、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』を参照してください。

表 9　DEFGEN のパラメータ (続き)

パラメータ	説明
TABLE <owner>.<table> [, {DEF | TARGETDEF} <template name>]; 変数 : ◆ <owner> は、スキーマ名です。 ◆ <table> は、表の名前またはワイルドカードで定義された表のグループの名前です。 ◆ [, {DEF | TARGETDEF} <template name>] では、この表の定義を使用して特定のテンプレートを作成することを指定します。このオプションは、初期ロードではサポートされません。	定義を設定する 1 つ以上の表を指定します。オプションで、定義テンプレートの基礎とする表を指定します。 ターゲット・システムに転送するソース定義ファイルを生成する場合は 1 つ以上のソース表を指定し、ソース・システムに転送するターゲット定義ファイルを生成する場合は 1 つ以上のターゲット表を指定します。 ワイルドカードの指定から表を除外するには、TABLEEXCLUDE パラメータを使用します。 **注意 :** DEFGEN では、UDT はサポートされません。

4. ファイルを保存して閉じます。
5. GGSCI を終了します。

DEFGEN を実行する手順

1. Oracle GoldenGate がインストールされているディレクトリから、次の引数を使用して DEFGEN を実行します。この例は、UNIX ファイル・システムの構造を示しています。

```
defgen paramfile dirprm/defgen.prm [reportfile dirrpt/defgen.rpt]
```

条件 :

- defgen は、プログラムの名前です。
- paramfile dirprm/defgen.prm は、DEFGEN パラメータ・ファイルの相対名またはフルパス名です。
- reportfile dirrpt/defgen.rpt は、画面および指定したレポート・ファイルに出力を送ります。画面に出力するのみの場合、この引数は省略できます。

注意　定義ファイルは変更しないでください。

2. ASCII モードを使用して、定義ファイルを Oracle GoldenGate の dirdef サブディレクトリから他のシステムに FTP 送信し、そのシステムの dirdef サブディレクトリに保存します。

Oracle GoldenGate プロセスで定義ファイルを参照するように設定する手順

次の方法でデータ定義ファイルを適切な Oracle GoldenGate プロセスにリンクします。

- Extract パラメータ・ファイルの TARGETDEFS パラメータを使用して、ターゲット定義ファイルを Extract グループまたはデータ・ポンプにリンクします。
- Replicat パラメータ・ファイルの SOURCEDEFS パラメータを使用して、ソース定義ファイルを Replicat グループにリンクします。
- Oracle GoldenGate によって、ソース・データベースもターゲット・データベースも含まない中間システムでマッピングまたは変換を実行する場合、パラメータ・ファイルの SOURCEDEFS および TARGETDEFS を使用して、ソース定義ファイルおよびターゲット定義ファイルをデータ・ポンプ Extract にリンクします。Oracle Database では、中間システムに Oracle ライブラリも存在している必要があります。

例

次に、Oracle データベース用の DEFGEN パラメータ・ファイルの例を示します。名前別の個別定義が、ord および hr スキーマのすべての表に対して作成されます。また、表 acct.cust100 に基づいて custdef テンプレートが作成されます。データベースには、それぞれ acct.cust100 と同一の定義を持つ他の acct.cust* 表が存在します。これらの表は、custdef テンプレートを参照する TABLE 文または MAP 文の DEF 句にマップできます。

```
DEFSFILE C:\ggs\dirdef\record.def
USERID ggs, PASSWORD ggs
TABLE acct.cust100, DEF custdef;
TABLE ord.*;
TABLE hr.*;
```

表は、Replicat のパラメータ・ファイルで次のようにマップされます。

```
REPLICAT acctrep
SOURCEDEFS c:\ggs\dirdef\record.def
USERID ggs, PASSWORD ggs
MAP acct.cust*, TARGET acct.cust*, DEF custdef;
MAP ord.prod, TARGET ord.prod;
MAP ord.parts, TARGET ord.parts;
MAP hr.emp, TARGET hr.emp;
MAP hr.salary, TARGET hr.salary;
```

前述の MAP 文で、ワイルドカードで指定された acct.cust* と一致するすべての表の定義は、custdef テンプレートから取得されます。

ソース定義と同様にターゲット定義も必要な場合、プライマリ Extract またはデータ・ポンプに対して同じ表をマップできます。たとえば、ターゲットが Enscribe データベースであった場合、または中間システムでマッピングまたは変換を実行する場合、構成は次のようになります。

```
EXTRACT acctext
USERID ggs, PASSWORD ggs
RMTHOST sysb, MGRPORT 7890
RMTTRAIL $data.ggsdat.rt
TABLE acct.cust*, TARGET acct.cust*, DEF custdef, TARGETDEF tcustdef;
TABLE ord.prod, TARGET ord.prod;
TABLE ord.parts, TARGET ord.parts;
TABLE hr.emp, TARGET hr.emp;
TABLE hr.salary, TARGET hr.salary;
```

前述の例で、custdef というソース・テンプレートと tcustdef というターゲット・テンプレートは、すべての acc.cust* 表で使用されます。ord および hr スキーマの表の定義は、表名に基づいた明示的な定義から取得されます。

第 12 章

オンライン変更同期の構成

オンライン変更同期の概要

オンライン変更同期では、データ変更を継続的に抽出およびレプリケートして、ほぼリアルタイムな状態にターゲット・データベースを維持します。次に、オンライン変更同期を構成するために必要な手順をまとめます。

- チェックポイント表を作成します。
- 1 つ以上の Extract グループを作成します。
- Extract のパラメータ・ファイルを作成します。
- 証跡を作成します。
- Replicat グループを作成します。
- Replicat のパラメータ・ファイルを作成します。

初期同期

この章の手順に従って変更同期グループおよび証跡を構成したら、217 ページの「初期データ・ロードの実行」を参照して同期用のターゲット表を準備します。初期ロードによって、ソース表全体がコピーされ、必要に応じてデータが変換され、トランザクション・データの移動が同期状態から開始されるようにデータがターゲット表に適用されます。変更同期を最初に開始するのは、初期同期プロセス中である必要があります。変更同期によって、ロードが適用されている間、進行中のトランザクション変更が追跡されます。

最高のパフォーマンスを得るためのプロセス・グループの構成

ビジネス・ルールを開発して、ソース・アプリケーション内で変更が発生した時点と、それらの変更がターゲット・データベースに適用される時点との間におけるラグの許容量を指定します。これらのルールによって、Oracle GoldenGate が最高のパフォーマンスを発揮するために必要な Extract および Replicat のパラレル・プロセスの数が決定されます。

Oracle GoldenGate でレプリケートする予定のすべての表に関するサイズおよびアクティビティ速度を収集します。

- アクティビティ速度の低いすべての表に、1 つの Extract グループを割り当てます。
- アクティビティ速度の高い表ごとに、専用の Extract グループを割り当てます。

これらの Extract グループは、専用のデータ・ポンプおよび Replicat グループと連携するように構成します。最高のパフォーマンスを得るための Oracle GoldenGate 構成の詳細は、『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。

グループのネーミング規則

Oracle GoldenGate のプロセス・グループに名前を付ける場合、次のルールに従います。

- 最大8つのASCII文字を使用できます(アンダースコア(_)などの英数字以外の文字にも対応)。任意のASCII 文字を使用できます (ただし、オペレーティング・システムでファイル名への使用が許可されている文字のみ)。この理由は、グループがその関連するチェックポイント・ファイルで識別されるためです。
- 次の ASCII 文字はファイル名には使用できません。

  { \ / : * ? " < > | }
- HP UX、Linux および Solaris では、コロン (:) またはアスタリスク (*) を使用してファイル名を指定できますが、推奨はされません。
- 一般的に、Oracle GoldenGate 内では、グループ名の大 / 小文字は区別されません。たとえば、finance、Finance および FINANCE は、すべて同じであるとみなされます。ただし、Linux では、グループ名 (および ADD コマンドで明示的に定義される場合はそのパラメータ・ファイル名) はすべて大文字またはすべて小文字である必要があります。グループ名およびパラメータ・ファイル名に大文字と小文字が混在していると、プロセスの起動時にエラーが発生します。
- 単一の語を使用してください。
- グループ名に port という語は使用しないでください。ただし、グループ名の一部として port という文字列を使用することは可能です。
- グループ名の末尾に数値を使用しないでください (fin1 や fin10 など)。グループ名の先頭であれば、数値を使用しても構いません (1_fin や 1fin など)。

チェックポイント表の作成

注意　チェックポイント表の機能は、c-tree データベースではサポートされません (Replicat がターゲット・データベースにアクセスできないため)。

Replicat は、予期される停止または予期されない停止の後に処理を開始する基準となる証跡内の既知の位置を指定するチェックポイントを管理します。チェックポイントのレコードを格納するために、Replicat はターゲット・データベースのチェックポイント表を使用します。これにより、Replicat のチェックポイントを Replicat トランザクション自体に含めることができるため、Replicat プロセスまたはデータベース・プロセスに障害が発生した場合でも、トランザクションは一度しか適用されません。不要になった行は削除されるため、チェックポイント表のサイズは小さく抑えられ、データベース・パフォーマンスには影響しません。

チェックポイント表作成のオプション

チェックポイント表は任意のスキーマに配置できます。可能であれば、Oracle GoldenGate 専用のものを使用してください。

Oracle GoldenGate の複数のインスタンス (複数のインストール環境) で同じチェックポイント表を使用できます。Oracle GoldenGate は、異なるインスタンスで Replicat グループの名前が同じであっても、チェックポイントを追跡します。

必要に応じて複数のチェックポイント表を使用できます。たとえば、Replicat グループごとに異なるものを使用できます。

チェックポイント表は次の方法でインストールできます。

- GLOBALS ファイルにデフォルトのチェックポイント表を指定できます。ADD REPLICAT コマンドを使用して作成された新規 Replicat グループは、特別な指示がなくてもこの表を自動的に使用します。「GLOBALS ファイルにデフォルトのチェックポイント表を指定する手順」を参照してください。
- 任意の Replicat グループを作成する際に、次のように特定のチェックポイント表の指示を指定できます。
  - グループで特定のチェックポイント表を使用するには、ADD REPLICAT コマンドで CHECKPOINTTABLE 引数を使用します。ADD REPLICAT に指定されたチェックポイント表は、GLOBALS ファイルのデフォルトの指定に優先します。ただ 1 つの Replicat グループを使用する場合、このコマンドを使用して、GLOBALS ファイルの作成を完全に省略できます。
  - グループでチェックポイント表の使用を省略するには、ADD REPLICAT コマンドで NODBCHECKPOINT 引数を使用します。チェックポイント表を使用しない場合でも、チェックポイントはディスク上のチェックポイント・ファイルに保持されますが、データの一貫性を失うリスクが生じます。詳細は、16 ページの「チェックポイントの概要」を参照してください。

チェックポイント表の実装方法にかかわらず、ADD REPLICAT コマンドを使用する前に、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ておく必要があります。「ターゲット・データベースにチェックポイント表を追加する手順」を参照してください。

GLOBALS ファイルにデフォルトのチェックポイント表を指定する手順

1. GLOBALS ファイルを作成します (または、適切な場合は既存のファイルを編集します)。UNIX または Linux システムでは、ファイル名はすべて大文字とし、ファイル拡張子を付けずに Oracle GoldenGate のルート・ディレクトリに配置する必要があります。ASCII テキスト・エディタを使用して、前述のネーミング規則に従ってファイルを作成します。または、GGSCI を使用してファイルを作成すると、自動的に正しい名前と場所を使用して保存されます。GGSCI を使用する場合、次のコマンドを使用します (GLOBALS は大文字で入力します)。

   ```
   EDIT PARAMS ./GLOBALS
   ```

2. 次のパラメータを入力します (大 / 小文字の区別はありません)。

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

   条件 : <owner>.<table> は、デフォルトのチェックポイント表の所有者および名前です。データベースでサポートされる任意の名前を指定します。

3. 表の名前を書き留め、GLOBALS ファイルを保存して閉じます。ファイルが Oracle GoldenGate のルート・ディレクトリに作成されたことを確認します。ファイル拡張子がある場合は、拡張子を削除します。

ターゲット・データベースにチェックポイント表を追加する手順

注意 GGSCI を通じてチェックポイント表を作成する次の手順は、かわりに chkpt_<db>_create.sql スクリプト (<db> はデータベース・タイプの略称) を実行することで省略できます。このスクリプトを使用することで、カスタム記憶域または他の属性を指定できます。この表の列の名前または属性は、変更しないでください。

1. Oracle GoldenGate ディレクトリから、GGSCI を実行して次のコマンドを発行し、データベースにログインします。

   ```
   DBLOGIN [SOURCEDB <dsn>][, USERID <db_user>[, PASSWORD <pw>]]
   ```

条件 :

- SOURCEDB <dsn> では、データソース名を指定します (接続情報の一部として必要な場合)。
- USERID <db_user>, PASSWORD <pw> では、必要に応じてデータベース資格証明を指定します。NonStop SQL/MX または DB2 では、PASSWORD は必要ありません。

このユーザーは、CREATE TABLE 権限を持っている必要があります。

2. GGSCI で、次のコマンドを発行してデータベースにチェックポイント表を追加します。

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

条件 :

<owner>.<table> は、表の所有者および名前です。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、GLOBALS ファイルの CHECKPOINTTABLE で指定する場合、所有者および名前は省略できます。

オンライン Extract グループの作成

オンライン Extract グループを作成するには、ソース・システムで GGSCI を実行し、ADD EXTRACT コマンドを発行します。コマンド引数はすべてカンマで区切ります。

標準、パッシブまたはデータ・ポンプ Extract グループを作成する手順

```
ADD EXTRACT <group name>
{, <datasource>}
{, BEGIN <start point>} | {<position point>}
[, PASSIVE]
[, THREADS <n>]
[, PARAMS <pathname>]
[, REPORT <pathname>]
[, DESC "<description>"]
```

条件 :

- <group name> は、Extract グループの名前です。グループ名は必須です (最大 8 文字を指定でき、大 / 小文字は区別されません)。詳細は、121 ページを参照してください。
- <datasource> は、抽出するデータのソースを指定する場合に必要です。次のいずれかを使用します。
  - TRANLOG [<bsds name>] では、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。Teradata 以外のすべてのデータベースで使用します。z/OS 上で稼働する DB2 では、<bsds> 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・データセットのファイル名を指定します。Oracle Enterprise Edition リリース 10.2 以上でこのオプションを使用する場合は、ADD EXTRACT を使用する前に (かつ、DELETE EXTRACT を発行して Extract グループを削除する前に)、Extract データベース・ユーザー (または同じ権限を持つユーザー) として DBLOGIN コマンドを発行する必要があります。
  - VAM では、*ベンダー・アクセス・モジュール* (VAM) と呼ばれる Extract API が Teradata アクセス・モジュール (TAM) とのインタフェースになるように指定します。Teradata データベースで使用します。

- VAMTRAILSOURCE <VAM trail name> では、VAM 証跡を指定します。最大保護モードの Teradata 抽出で VAM ソート Extract グループを作成する場合に使用します。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。
- EXTTRAILSOURCE <trail name> では、ローカル証跡の相対名または完全修飾名を指定します。データ・ポンプを作成する場合に使用します。データ・ポンプは、Oracle GoldenGate の任意の抽出方法と組み合せて使用できます。

○ BEGIN <start point> では、処理のための初期チェックポイントおよび開始ポイントを確定してオンライン Extract グループを定義します。このポイントより前に開始されたトランザクションは、破棄されます。次のいずれかを使用します。

- NOW では、グループを作成するために ADD EXTRACT コマンドが実行された時点のタイムスタンプが指定された変更から抽出を開始します。 ADD EXTRACT 文よりも前に Oracle GoldenGate の証跡に取得されたデータを回避しない場合は、データ・ポンプ Extract に NOW を使用しないでください。
- <YYYY-MM-DD HH:MM[:SS[.CCCCCC]]> は、開始ポイントとして正確なタイムスタンプを指定するための書式です。レプリケーションまたはロギングが有効化された時点より後の開始ポイントを使用してください。

  注意 Teradata ソースには BEGIN パラメータを使用しないでください。

○ <position point> では、特定のトランザクション・ログ・ファイル内で処理を開始する特定の位置を指定します。データベースで使用する特定の構文は、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』の ADD EXTRACT の説明を参照してください。

○ PASSIVE では、グループがパッシブ Extract であることを示します。PASSIVE を使用する場合、エイリアス Extract も使用する必要があります。詳細は、106 ページを参照してください。このオプションは、他の ADD EXTRACT オプション内に任意の順序で配置できます。

○ THREADS <n> は、Oracle Real Application Clusters (RAC) にのみ必要です。このオプションでは、クラスタで使用する REDO ログ・スレッドの数を指定します。

○ PARAMS <pathname> は、このグループのパラメータ・ファイルを Oracle GoldenGate ディレクトリの dirprm サブディレクトリ以外の場所に格納する場合に必要です。完全修飾名を指定します。デフォルトの場所をお薦めします。

○ REPORT <pathname> は、このグループのプロセス・レポートを Oracle GoldenGate ディレクトリの dirrpt サブディレクトリ以外の場所に格納する場合に必要です。完全修飾名を指定します。デフォルトの場所をお薦めします。

○ DESC "<description>" では、グループの説明を指定します。

エイリアス Extract グループを作成する手順

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

条件 :

○ RMTHOST では、このグループをエイリアス Extract として識別し、リモート・ホストの DNS 名またはその IP アドレスを指定します。

○ MGRPORT では、Manager が稼働しているリモート・システムのポートを指定します。動的 Collector を使用する場合、このオプションを使用します。

- PORT では、静的 Collector のポートを指定します。静的 Collector を実行する場合にのみ、MGRPORT のかわりに使用します。
- RMTNAME では、パッシブ Extract の名前を指定します (エイリアス Extract の名前と異なる場合)。
- DESC "<description>" では、グループの説明を指定します。

例 1　ログベース抽出

この例では、finance という Extract グループを作成します。抽出は、グループの作成時点に生成されたレコードから開始します。

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

例 2　Teradata 抽出 (プライマリ Extract)

この例では、Teradata 最大パフォーマンス・モードまたは Teradata 最大保護モードで実行される finance という Extract グループを作成します。Teradata ソースには BEGIN ポイントは使用しません。

```
ADD EXTRACT finance, VAM
```

例 3　Teradata 抽出 (VAM ソート Extract)

この例では、finance という VAM ソート Extract グループを作成します。プロセスは、VAM 証跡の /ggs/dirdat/vt から読取りを行います。

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

例 4　データ・ポンプ Extract グループ

この例では、finance というデータ・ポンプ Extract グループを作成します。このグループは、Oracle GoldenGate 証跡の c:\ggs\dirdat\lt から読取りを行います。

```
ADD EXTRACT finance, EXTTRAILSOURCE c:\ggs\dirdat\lt
```

例 5　パッシブ Extract グループ

この例では、finance というパッシブ Extract グループを作成します。抽出は、グループの作成時に生成されたレコードから開始します。このグループはパッシブとしてマークされるため、ターゲットのエイリアス Extract がこの Extract への接続を開始します。

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

例 6　パッシブ・データ・ポンプ Extract グループ

この例では、finance というデータ・ポンプ Extract グループを作成します。これは、Oracle GoldenGate 証跡の c:\ggs\dirdat\lt から読取りを行うパッシブ・データ・ポンプ Extract です。このデータ・ポンプはパッシブとしてマークされるため、ターゲットのエイリアス Extract がこのデータ・ポンプへの接続を開始します。

```
ADD EXTRACT finance, EXTTRAILSOURCE c:\ggs\dirdat\lt, PASSIVE
```

例 7　エイリアス Extract グループ

この例では、alias というエイリアス Extract グループを作成します。

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

証跡の作成

データを抽出したら、1 つ以上の証跡に移行する必要があります。証跡では、別の Oracle GoldenGate プロセスによって処理するためにデータが格納されます。証跡は、必要に応じて作成およびエージングされる一連のファイルです。証跡を読み取るプロセスは、次のとおりです。

- VAM ソート Extract: VAM 証跡として作成されたローカル証跡から抽出します (Teradata ソース・データベース用)。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。
- データ・ポンプ Extract: 後続の処理のために必要に応じてローカル証跡からデータを抽出し、そのデータをターゲット・システムに転送します。
- Replicat: 証跡を読み取って変更データをターゲット・データベースに適用します。

異なる表またはアプリケーションのデータを分離するために、複数の証跡を作成できます。TABLE 文で指定した表を、Extract パラメータ・ファイルの EXTTRAIL または RMTTRAIL パラメータ文で指定した証跡にリンクします。Oracle GoldenGate 証跡の詳細は、14 ページを参照してください。

証跡を定義する手順

ソース・システムの GGSCI で、次のコマンドを発行します。

```
ADD {RMTTRAIL | EXTTRAIL} <pathname>, EXTRACT <group name>
[, MEGABYTES <n>]
```

条件 :

- RMTTRAIL では、リモート・システムの証跡を指定します。
- EXTTRAIL では、ローカル・システムの証跡を指定します。
  - EXTTRAIL は、PASSIVE モードの Extract には使用できません。
  - EXTTRAIL は、データ・ポンプによって読み取られるローカル証跡を指定する場合、または Teradata アクセス・モジュール (TAM) と相互作用するプライマリ Extract にリンクされた VAM 証跡を指定する場合に使用する必要があります。Teradata 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。
- <pathname> は、2 文字の名前 (任意の 2 つの英数字) を含む証跡の相対名または完全修飾名です (c:\ggs\dirdat\rt など)。Oracle GoldenGate によって、処理中に作成された各証跡ファイルにシリアル番号が追加されます。通常、証跡は、Oracle GoldenGate ディレクトリの dirdat サブディレクトリに格納されます。
- EXTRACT <group name> は、この証跡に書込みを行う Extract グループの名前です。1 つの Extract グループのみが、証跡に書き込むことができます。
- MEGABYTES <n> は、各証跡ファイルのサイズを MB 単位で設定できるオプション引数です (デフォルトは 10 です)。

例 この例では、Extract グループの extvam に対して /ggs/dirdat/vt という VAM 証跡を作成します。

```
ADD EXTTRAIL /ggs/dirdat/vt, EXTRACT extvam
```

例 この例では、Extract グループの ext に対して /ggs/dirdat/lt というローカル証跡を作成します。

```
ADD EXTTRAIL /ggs/dirdat/lt, EXTRACT ext
```

例 この例では、Extract グループの finance に対して、各ファイル・サイズを約 50MB として c:\ggs\dirdat\rt という証跡を作成します。

```
ADD RMTTRAIL c:\ggs\dirdat\rt, EXTRACT finance, MEGABYTES 50
```

オンライン抽出用のパラメータ・ファイルの作成

次の手順に従って、オンライン Extract グループのパラメータ・ファイルを作成します。パラメータ・ファイルは、エイリアス Extract グループには必要ありません。詳細は、106 ページを参照してください。

1. ソース・システムの GGSCI で、次のコマンドを発行します。

   ```
   EDIT PARAMS <name>
   ```

   条件 : <name> は、ADD EXTRACT コマンドで作成した Extract グループの名前です。または、グループの作成時に代替の場所を定義した場合は、パラメータ・ファイルの完全修飾名です。

2. 表 10 に示されている順序でパラメータを入力します。パラメータ文ごとに新規行を開始します。一部のパラメータは、特定の構成にのみ適用されます。

表 10　　オンライン変更抽出のパラメータ

パラメータ	説明
`EXTRACT <group name>` ◆ <group name> は、ADD EXTRACT コマンドで作成した Extract グループの名前です。	Extract は、チェックポイント付きのオンライン・プロセスとして構成します。

`[SOURCEDB <dsn>,]`
`[USERID <user id> [, PASSWORD <pw>]]`
◆ SOURCEDB では、データソース名を指定します (接続情報で必要な場合)。Oracle では必要ありません。
◆ USERID では、必要に応じてデータベース資格証明を指定します。Oracle では、次のようなホスト文字列を含めることができます。
`USERID ggs@ora1.ora, PASSWORD ggs123`
NonStop SQL/MX または DB2 では、PASSWORD は必要ありません。
	データベース接続情報を指定します。これらのパラメータでは、オペレーティング・システム・レベルでの認証も可能です。『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』を参照してください。
	このパラメータは、グループがデータベースの存在しない中間システム上のデータ・ポンプである場合には省略できます。この場合、列マッピングまたは変換は実行されません。

`RMTHOST <hostname>,` `MGRPORT <portnumber>`	Manager が稼働しているターゲット・システムおよびポートを指定します。IP を通じてリモート・システムにデータを送信する場合にのみ必要です (ADD RMTTRAIL を使用して証跡を作成した場合)。証跡がローカル・システムに存在する場合には、必要ありません (ADD EXTTRAIL を使用した場合)。 Teradata アクセス・モジュールとのインタフェースになり、VAM 証跡に書込みを行うプライマリ Extract グループには無効です。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。 パッシブ Extract グループにも無効です。

表 10　オンライン変更抽出のパラメータ (続き)

パラメータ	説明
RMTTRAIL <full_pathname> | EXTTRAIL <full_pathname> ◆ RMTTRAIL を使用して、ADD RMTTRAIL コマンドで作成されたリモート証跡の相対名または完全修飾名を指定します。 ◆ EXTTRAIL を使用して、ADD EXTTRAIL コマンドで作成された (データ・ポンプまたは VAM ソート Extract によって読み取られる) ローカル証跡の相対名または完全修飾名を指定します。	証跡を指定します。複数の証跡を指定する場合、適切な TABLE 文を各指定の後に続けます。 EXTTRAIL は、パッシブ Extract グループには無効です。 証跡またはファイルのバージョンが異なる場合、RMTTRAIL または EXTTRAIL の FORMAT オプションを使用します。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。 証跡または抽出ファイルのバージョンは、そのファイルを*読み取るプロセス*のバージョン以下である必要があります。それ以外の場合、プロセスは異常終了します。また、データ・ポンプの出力証跡またはファイルは、Oracle GoldenGate によって強制的に入力証跡またはファイルと同じバージョンに設定されます。再起動時に、Extract は、各ファイルのバージョンがただ 1 つになるように証跡を 1 つの新規ファイルにまとめます (ファイルが空ではない場合)。

DSOPTIONS {
COMMITTEDTRANLOG, RESTARTAPPEND |
CREATETRANLOG |
SORTTRANLOG
}
	Teradata 抽出にのみ有効です。
	◆ COMMITTEDTRANLOG, RESTART APPEND を使用して、Extract によって Teradata 最大パフォーマンス・モードで完全コミット済データを受信することを示します。RESTARTAPPEND では、以前の実行からデータを再書込みするのではなく、Oracle GoldenGate 証跡の最後にデータを追加します。
	◆ CREATETRANLOG を使用して、ローカル VAM 証跡を作成して Teradata 最大保護モードでそこに書き込むことを Extract に指示します。Teradata アクセス・モジュールとのインタフェースになるプライマリ Extract グループで使用します。
	◆ SORTTRANLOG を使用して、Extract によってローカル VAM 証跡から読取りを行い、最大保護モードでコミット順にデータをソートします。VAM ソート Extract グループにのみ使用します。
	Teradata 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。

VAM <library name>, PARAMS ("<param>" [, "<param>"] [, ...])	Teradata アクセス・モジュールとのインタフェースになる Extract グループにのみ有効です。Oracle GoldenGate API に渡す必要のあるライブラリの名前およびパラメータを指定します。たとえば、TAM 初期化ファイルの名前や、コールバック・ライブラリとして使用するライブラリとのインタフェースになるプログラムなどです。 例 : VAM vam.dll, PARAMS ("inifile", "vamerge1.ini", "callbacklib", "extract.exe")

表 10　　オンライン変更抽出のパラメータ (続き)

パラメータ	説明
PASSTHRU | NOPASSTHU	(データ・ポンプの場合) 後続の TABLE 指定で通常処理とパススルー処理のどちらを使用するかを指定します。
SEQUENCE <owner>.<sequence>; ◆ <owner> は、スキーマ名です。 ◆ <sequence> は、順序の名前です。	取得する Oracle 順序を指定します。
TABLE <owner>.<table>; ◆ <owner> は、スキーマ名です。 ◆ <table> は、表の名前またはワイルドカードで定義された表のグループの名前です。 ワイルドカードの指定から表を除外するには、TABLEEXCLUDE パラメータを使用します。	データ変更を抽出する 1 つ以上の表を指定します。 スキーマ名にワイルドカードは使用できません。複数のスキーマの表からデータを抽出するには、スキーマごとに個別の TABLE 文を使用します。次に例を示します。 TABLE fin.*; TABLE hr.*;

3. 『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』にリストされている Extract の適切なオプション・パラメータを入力します。
4. パラメータ・ファイルを保存して閉じます。

オンライン Replicat グループの作成

オンライン Replicat グループを作成するには、ターゲット・システムで GGSCI を実行し、ADD REPLICAT コマンドを発行します。コマンド引数はすべてカンマで区切ります。

```
ADD REPLICAT <group name>, EXTTRAIL <pathname>
[, BEGIN <start point> | , EXTSEQNO <seqno>, EXTRBA <rba>]
[, CHECKPOINTTABLE <owner.table>]
[, NODBCHECKPOINT]
[, PARAMS <pathname>]
[, REPORT <pathname>]
```

条件 :

- <group name> は、Replicat グループの名前です。グループ名は必須です (最大 8 文字を指定でき、大 / 小文字は区別されません)。詳細は、121 ページを参照してください。
- EXTTRAIL <pathname> は、ADD RMTTRAIL コマンドで定義した証跡の相対名または完全修飾名です。
- BEGIN <start point> では、処理のための初期チェックポイントおよび開始ポイントを確定してオンライン Replicat グループを定義します。次のいずれかを使用します。
  - NOW では、グループを作成するために ADD REPLICAT コマンドが実行された時点のタイムスタンプが指定された変更からレプリケートを開始します。
  - <YYYY-MM-DD HH:MM[:SS[.CCCCCC]]> は、開始ポイントとして正確なタイムスタンプを指定するための書式です。

- EXTSEQNO <seqno>, EXTRBA <relative byte address> では、データの読取りを開始する証跡内のファイルの順序番号と、その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しない場合、処理はデフォルトで証跡の最初から開始されます。順序番号には数値を指定しますが、埋込み用の 0 (ゼロ) は使用しません。たとえば、証跡ファイルが c:\ggs\dirdat\aa000026 である場合、EXTSEQNO 26 と指定します。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 にアクセスしてください。
- CHECKPOINTTABLE <owner.table> では、GLOBALS ファイルで指定されたデフォルト以外のチェックポイント表の所有者および名前を指定します。この引数を使用するには、ADD CHECKPOINTTABLE コマンドを使用してデータベースにチェックポイント表を追加する必要があります (120 ページの「初期同期」を参照)。
- NODBCHECKPOINT では、この Replicat グループでチェックポイント表を使用しないことを指定します。
- PARAMS <pathname> は、このグループのパラメータ・ファイルを Oracle GoldenGate ディレクトリの dirprm サブディレクトリ以外の場所に格納する場合に必要です。完全修飾名を指定します。デフォルトの場所をお薦めします。
- REPORT <pathname> は、このグループのプロセス・レポートを Oracle GoldenGate ディレクトリの dirrpt サブディレクトリ以外の場所に格納する場合に必要です。完全修飾名を指定します。デフォルトの場所をお薦めします。

例 次の例では、finance という名前のオンライン Replicat グループを作成し、c:\ggs\dirdat\rt という証跡を指定します。このパラメータ・ファイルは、代替場所である \ggs\params に格納され、レポート・ファイルはデフォルトの場所に格納されます。

```
ADD REPLICAT finance, EXTTRAIL c:\ggs\dirdat\rt, PARAMS \ggs\params
```

オンライン・レプリケーション用のパラメータ・ファイルの作成

次の手順に従って、オンライン Replicat グループのパラメータ・ファイルを作成します。

1. ターゲット・システムの GGSCI で、次のコマンドを発行します。

   ```
   EDIT PARAMS <name>
   ```

   条件 : <name> は、ADD REPLICAT コマンドで作成した Replicat グループの名前です。または、グループの作成時に代替の場所を定義した場合は、パラメータ・ファイルの完全修飾名です。

2. 表 11 にリストされている順序でパラメータを入力します。パラメータ文ごとに新規行を開始します。

表 11 オンライン変更レプリケーションのパラメータ

パラメータ	説明
`REPLICAT <group name>` ◆ <group name> は、ADD REPLICAT コマンドで作成した Replicat グループの名前です。	Replicat は、チェックポイント付きのオンライン・プロセスとして構成します。

表 11　オンライン変更レプリケーションのパラメータ (続き)

パラメータ	説明
{SOURCEDEFS <full_pathname>} | ASSUMETARGETDEFS ◆ SOURCEDEFS は、ソース表とターゲット表に異なる定義が含まれる場合に使用します。DEFGEN によって生成されたソースのデータ定義ファイルを指定します。詳細は、第 11 章を参照してください。 ◆ ASSUMETARGETDEFS は、ソース表とターゲット表に同じ定義が含まれる場合に使用します。	データ定義の解釈方法を指定します。 マルチバイト・キャラクタ・セットを使用する Oracle Database では、ソースのセマンティクス設定がバイトでターゲットの設定が文字の場合、(DEFGEN で生成された定義ファイルとともに) SOURCEDEFS を使用する必要があります。これは、ソースとターゲットのデータ定義が同一である場合でも必要です。詳細は、150 ページを参照してください。
DISCARDFILE <full_pathname> [, MEGABYTES <n>] [, PURGE] ◆ <full pathname> は、廃棄ファイルの相対名または完全修飾名です。デフォルトの場所は、Oracle GoldenGate ディレクトリの dirrpt サブディレクトリです。 ◆ MEGABYTES <n> では、廃棄ファイルの最大サイズを指定します。 ◆ PURGE では、既存の廃棄ファイルを上書きします。	拒否されたレコード・データ (データベース・エラーを生成したレコードなど) を Replicat が書き込むファイルを指定します。廃棄ファイルはオプションですが、使用することをお薦めします。
[DEFERAPPLYINTERVAL <n><unit>] ◆ <n> は、遅延の時間を示す数値です。最小値は、EOFDELAY パラメータによって設定されます。最大値は 7 日です。 ◆ <unit> には次の単位を指定できます。 S | SEC | SECS | SECOND | SECONDS | MIN | MINS | MINUTE | MINUTES | HOUR | HOURS | DAY | DAYS	オプションです。Replicat がターゲット・システムに取得トランザクションを適用するまでに待機する時間を指定します。
[TARGETDB <dsn>,] [USERID <user id> [, PASSWORD <pw>]] ◆ TARGETDB では、データソース名を指定します (接続情報で必要な場合)。Oracle では必要ありません。 ◆ USERID では、必要に応じてデータベース資格証明を指定します。Oracle では、次のようなホスト文字列を含めることができます。 USERID ggs@ora1.ora, PASSWORD ggs123	データベース接続情報を指定します。これらのパラメータでは、オペレーティング・システム・レベルでの認証も可能です。『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』を参照してください。

表 11　　オンライン変更レプリケーションのパラメータ (続き)

パラメータ	説明
MAP <owner>.<table>, TARGET <owner>.<table>[, DEF <template name>]; ◆ <owner> は、スキーマ名です。 ◆ <table> は、表の名前または複数の表を示すワイルドカード定義です。 ◆ [, DEF <template name>] では、定義テンプレートを指定します。(第 11 章を参照してください。)	ソースとターゲットの 1 つ以上の表どうしの関係を指定します。 スキーマ名にワイルドカードは使用できません。複数のスキーマの表からデータを抽出するには、スキーマごとに個別の MAP 文を使用します。次に例を示します。 MAP fin.*, TARGET fin.*; MAP hr.*, TARGET hr.*; ワイルドカードの指定から表を除外するには、MAPEXCLUDE パラメータを使用します。

3. 『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』にリストされている Replicat の適切なオプション・パラメータを入力します。
4. ファイルを保存して閉じます。

オンライン処理の制御

オンライン・プロセスを起動および停止するには、GGSCI を使用します。

注意　ユーザー・アカウント制御が有効化された Windows Server 2008 では、Manager が Windows サービスとしてインストールされていない場合、Oracle GoldenGate プロセスを起動すると UAC プロンプトが表示されます。

オンライン・プロセスを初めて起動する場合の手順

通常、ソース・ユーザー・アプリケーションのアクティブ状態を維持する必要があるとすれば、本番設定で Oracle GoldenGate プロセスが最初に起動するのは、初期同期プロセスの実行中です。ターゲットにソース・データがロードされる間、Oracle GoldenGate は、進行中のユーザー変更を取得し、それらの変更とロードの結果とを調整します。詳細は、217 ページの第 16 章を参照してください。

注意　Extract が新しい Oracle GoldenGate 構成で初めて起動する場合、すべてのオープン・トランザクションはスキップされます。Extract の起動後に開始されたトランザクションのみが取得されます。

オンライン・プロセスを起動する手順

```
START {EXTRACT | REPLICAT} <group_name>
```

条件 :

<group_name> は、Extract または Replicat グループの名前か、またはグループのワイルドカード・セット (* や fin* など) です。

注意　PASSIVE モードの Extract は、関連付けられたエイリアス Extract を起動することでのみ起動できます。詳細は、106 ページを参照してください。

証跡の最初のトランザクションをスキップしたり、特定のトランザクションから開始するために必要に応じて使用できる追加の START REPLICAT オプションは、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』を参照してください。

プロセスを自動起動する手順

- Manager パラメータ・ファイルの AUTOSTART を使用して、Manager の起動時に 1 つ以上のプロセスを起動します。
- Manager パラメータ・ファイルの AUTORESTART を使用して、障害後にプロセスを再起動します。

これら 2 つのパラメータによって、START コマンドを使用して手動でプロセスを起動する必要がなくなります。

オンライン・プロセスを正常に停止する手順

```
STOP {EXTRACT | REPLICAT} <group_name>
```

条件 :

<group_name> は、Extract または Replicat グループの名前か、またはグループのワイルドカード・セット (* や fin* など) です。

> **注意** PASSIVE モードで稼働している Extract は、関連付けられたエイリアス Extract を停止することでのみ停止できます。詳細は、106 ページを参照してください。

Replicat を強制的に停止する手順

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

現在のトランザクションは中断され、プロセスは即座に停止されます。Extract を強制的に停止することはできません。

STOP で停止できないプロセスを強制終了する手順

```
KILL {EXTRACT | REPLICAT} <group name>
```

プロセスを強制終了すると正常な停止は行われず、チェックポイント情報が失われる可能性があります。

複数のプロセスを同時に制御する手順

```
<command> ER <wildcard specification>
```

条件 :

- <command> は、KILL、START または STOP です。
- <wildcard specification> は、コマンドの対象のプロセス・グループの名前を示すワイルドカードの指定です。コマンドは、ワイルドカードに一致するすべての Extract および Replicat グループに影響します。Oracle GoldenGate では、最大 100,000 のワイルドカード・エントリがサポートされます。

プロセス・グループの削除

オンライン・プロセスを停止した後に、グループを削除できます。グループを削除しても、パラメータ・ファイルは保持されます。同じパラメータ・ファイルを使用して同じグループを再作成できます。または、パラメータ・ファイルを削除して、グループの構成を永久に削除することも可能です。

Extract グループを削除する手順

1. GGSCI を実行します。

2. (Oracle Enterprise Edition 10.2 以上)Extract データベース・ユーザー (または同じ権限を持つユーザー) として DBLOGIN コマンドを発行します。

```
DBLOGIN USERID <Extract_user> [, PASSWORD <password>]
```

3. 次のコマンドを発行します。

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

! 引数を使用すると、ワイルドカードに一致するすべての Extract グループが確認なしで削除されます。

Replicat グループを削除する手順

1. このグループでチェックポイント表を使用している場合、GGSCI から次のコマンドを発行してデータベースにログインします。

```
DBLOGIN [SOURCEDB <dsn>] [USERID <user>[, PASSWORD <password>]]
```

条件 :

- SOURCEDB <dsn> では、データソース名を指定します (接続情報の一部として必要な場合)。
- USERID <user>, PASSWORD <password> では、必要に応じてデータベース資格証明を指定します。

2. 次のコマンドを発行してグループを削除します。

```
DELETE REPLICAT <group>
```

DBLOGIN を使用してデータベースにログインするかわりに、DELETE REPLICAT に ! オプションを使用できます。

```
DELETE REPLICAT <group> !
```

DELETE REPLICAT によって、チェックポイント・ファイルは削除されますが、チェックポイント表のチェックポイントは保持されます。基本の DELETE REPLICAT コマンドでは Replicat トランザクションはコミットされますが、! オプションではコミットされません。

第 13 章

バッチ実行としての変更同期の構成

バッチ変更同期の概要

Extract および Replicat を構成して個別のバッチ実行 (または*特別実行*) を実施し、特定の開始時間と終了時間の間に生成されたデータ変更を抽出およびレプリケートできます。リカバリ・ポイントは不要なため、チェックポイントはバッチ実行中に記録されません。プロセスに障害が発生した場合は、同じ開始ポイントと終了ポイントを使用して、単純にもう一度最初から開始できます。

次に、バッチ実行の設定に必要な手順をまとめます。

- バッチ Extract パラメータ・ファイルを作成します。
- バッチ Replicat パラメータ・ファイルを作成します。
- オペレーティング・システムのコマンド・シェルからプロセスを起動します。

初めて Oracle GoldenGate を実行してデータ変更を同期する場合、状況に応じて初期ロードを実行し、ターゲット表の同期を準備する必要があります。初期ロードによって、ソース表全体がコピーされ、必要に応じてデータが変換され、トランザクション・データの移動が同期状態から開始されるようにデータがターゲット表に適用されます。217 ページの「初期データ・ロードの実行」を参照してください。

バッチ抽出用のパラメータ・ファイルの作成

1. ソース・システムの Oracle GoldenGate ディレクトリから GGSCI を実行します。
2. GGSCI で、次のコマンドを発行します。

   ```
   EDIT PARAMS <name>
   ```

   条件 : <name> は、Extract グループの名前です (最大 8 文字、大 / 小文字は区別されません)。
3. 表 12 にリストされている順序でパラメータを入力します。パラメータ文ごとに新規行を開始します。

表 12　バッチ変更抽出のパラメータ

パラメータ	説明
SPECIALRUN TRANLOG [<bsds name>] | EXTTRAILSOURCE <trail name>] | EXTFILESOURCE <file name>} ◆ TRANLOG では、トランザクション・ログから抽出を行います。ログベース抽出で使用します。z/OS 上の DB2 では、<bsds> 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・データセットのファイル名を指定します。 ◆ EXTTRAILSOURCE <trail name> では、ローカル証跡から抽出を行います。証跡のセットから処理を行う場合に使用します。 ◆ EXTFILESOURCE では、ローカル抽出ファイルから抽出を行います。データ・ポンプで使用します。 両方のファイル・タイプの相対名またはフルパス名を指定します。	チェックポイントが不要なバッチ実行として Extract を構成します。 データ・ポンプを使用する場合、証跡または抽出ファイルから抽出を行うことができます。
BEGIN <begin time> ◆ <begin time> は、yyyy-mm-dd hh:mi[:ss[.cccccc]] という書式の日付です。	処理を開始するトランザクション・コミット時刻を指定します。
END {<end time> | RUNTIME} ◆ <end time> は、yyyy-mm-dd [hh:mi[:ss[.cccccc]] という書式の日付です。 ◆ RUNTIME によって、Extract は、プロセスの起動時刻に到達すると終了します。RUNTIME を使用すると、実行ごとにパラメータ・ファイルを修正して日時を変更する必要がなくなります。	処理を停止するトランザクション・コミット時刻を指定します。
[SOURCEDB <dsn>,] [USERID <user id> [, PASSWORD <pw>]] ◆ SOURCEDB では、データソース名を指定します (接続情報で必要な場合)。Oracle では必要ありません。 ◆ USERID では、必要に応じてデータベース資格証明を指定します。Oracle では、次のようなホスト文字列を含めることができます。 USERID ggs@ora1.ora, PASSWORD ggs123 NonStop SQL/MX または DB2 では、PASSWORD は必要ありません。	データベース接続情報を指定します。これらのパラメータでは、オペレーティング・システム・レベルでの認証も可能です。『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』を参照してください。 このパラメータは、グループがデータベースの存在しない中間システム上のデータ・ポンプである場合には省略できます。この場合、列マッピングまたは変換は実行されません。
RMTHOST <hostname>, MGRPORT <portnumber>	Manager が稼働しているターゲット・システムおよびポートを指定します。このオプションは、IP を通じてリモート・システムにデータを送信する場合にのみ必要です。

表 12　　バッチ変更抽出のパラメータ (続き)

パラメータ	説明
RMTFILE <full_pathname> | EXTFILE <full_pathname> ◆ RMTFILE では、ターゲット・システムの抽出ファイルを指定します。 ◆ EXTFILE では、(データ・ポンプで使用する) ローカル抽出ファイルを指定します。 相対ファイル名または完全修飾ファイル名を指定します。	データ変更を一時的に格納する抽出ファイルを指定します。複数のファイルを指定する場合、適切な TABLE 文を各指定の後に続けます。 ファイルのバージョンが異なる場合、RMTTRAIL または EXTTRAIL の FORMAT オプションを使用します。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。 証跡または抽出ファイルのバージョンは、そのファイルを*読み取る*プロセスのバージョン以下である必要があります。それ以外の場合、プロセスは異常終了します。また、データ・ポンプの出力証跡またはファイルは、Oracle GoldenGate によって強制的に入力証跡またはファイルと同じバージョンに設定されます。再起動時に、Extract は、各ファイルのバージョンがただ 1 つになるように証跡を 1 つの新規ファイルにまとめます (ファイルが空ではない場合)。
PASSTHRU | NOPASSTHU	(データ・ポンプの場合) 後続の TABLE 指定で通常処理とパススルー処理のどちらを使用するかを指定します。
TABLE <owner>.<table>; ◆ <owner> は、スキーマ名です。 ◆ <table> は、表の名前またはワイルドカードで定義された表のグループの名前です。	データ変更を抽出する 1 つ以上の表を指定します。 ワイルドカードの指定から表を除外するには、TABLEEXCLUDE パラメータを使用します。

4. 『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』にリストされている Extract の適切なオプション・パラメータを入力します。
5. パラメータ・ファイルを保存して閉じます。

バッチ・レプリケーション用のパラメータ・ファイルの作成

1. ターゲット・システムの Oracle GoldenGate ディレクトリから GGSCI を実行します。
2. GGSCI で、次のコマンドを発行します。

   ```
   EDIT PARAMS <name>
   ```

   条件 :　<name> は、Replicat グループの名前です (最大 8 文字、大 / 小文字は区別されません)。
3. 表 13 にリストされている順序でパラメータを入力します。パラメータ文ごとに新規行を開始します。

表 13　バッチ変更レプリケーションのパラメータ

パラメータ	説明
SPECIALRUN	Replicat をバッチ実行として構成します。
BEGIN <begin time> ◆ <begin time> は、yyyy-mm-dd hh:mi[:ss[.cccccc]] という書式の日付です。	処理を開始するトランザクション・コミット時刻を指定します。
END {<end time> | RUNTIME} ◆ <end time> は、yyyy-mm-dd [hh:mi[:ss[.cccccc]] という書式の日付です。 ◆ RUNTIME によって、Replicat は、プロセスの起動時刻に到達すると終了します。RUNTIME を使用すると、実行ごとにパラメータ・ファイルを修正して日時を変更する必要がなくなります。	処理を停止するトランザクション・コミット時刻を指定します。
EXTFILE <file name>	Extract パラメータ・ファイルの RMTFILE または EXTFILE で指定された抽出ファイルの相対名または完全修飾名を指定します。
{SOURCEDEFS <file name>} | ASSUMETARGETDEFS ◆ SOURCEDEFS は、ソース表とターゲット表に異なる定義が含まれる場合に使用します。DEFGEN によって生成されたソース定義ファイルの相対名またはフルパス名を指定します。詳細は、第 11 章を参照してください。 ◆ ASSUMETARGETDEFS は、ソース表とターゲット表に同じ定義が含まれる場合に使用します。	データ定義の解釈方法を指定します。 マルチバイト・キャラクタ・セットを使用する Oracle Database では、ソースのセマンティクス設定がバイトでターゲットの設定が文字の場合、(DEFGEN で生成された定義ファイルとともに)SOURCEDEFS を使用する必要があります。これは、ソースとターゲットのデータ定義が同一である場合でも必要です。
DISCARDFILE <full_pathname> [, MEGABYTES <n>] [, PURGE] ◆ <full pathname> は、廃棄ファイルの相対名または完全修飾名です。デフォルトの場所は、Oracle GoldenGate ディレクトリの dirrpt サブディレクトリです。 ◆ MEGABYTES <n> では、廃棄ファイルの最大サイズを指定します。 ◆ PURGE では、既存の廃棄ファイルを上書きします。	拒否されたレコード・データ (データベース・エラーを生成したレコードなど) を Replicat が書き込むファイルを指定します。廃棄ファイルはオプションですが、使用することをお薦めします。

表 13　バッチ変更レプリケーションのパラメータ (続き)

パラメータ	説明
[TARGETDB <dsn>,] [USERID <user id> [, PASSWORD <pw>]] ◆ TARGETDB では、データソース名を指定します (接続情報で必要な場合)。Oracle では必要ありません。 ◆ USERID では、必要に応じてデータベース資格証明を指定します。Oracle では、次のようなホスト文字列を含めることができます。 USERID ggs@ora1.ora, PASSWORD ggs123	データベース接続情報を指定します。これらのパラメータでは、オペレーティング・システム・レベルでの認証も可能です。『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』を参照してください。
MAP <owner>.<table>, TARGET <owner>.<table>[, DEF <template name>]; ◆ <owner> は、スキーマ名です。 ◆ <table> は、表の名前または複数の表を示すワイルドカード定義です。 ◆ [, DEF <template name>] では、定義テンプレートを指定します。(第 11 章を参照してください。)	ソースとターゲットの 1 つ以上の表どうしの関係を指定します。 ワイルドカードの指定から表を除外するには、MAPEXCLUDE パラメータを使用します。

4. 『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』にリストされている Replicat の適切なオプション・パラメータを入力します。
5. ファイルを保存して閉じます。

オペレーティング・システムのコマンド・シェルからのプロセスの起動

バッチ変更同期ジョブを開始するには、オペレーティング・システムのコマンド・シェルから extract プログラムおよび replicat プログラムを実行します。バッチ実行は、END パラメータに従って自動的に終了します。

注意　ユーザー・アカウント制御が有効化された Windows Server 2008 では、Manager が Windows サービスとしてインストールされていない場合、Oracle GoldenGate プロセスを起動すると UAC プロンプトが表示されます。

コマンド・シェルからプロセスを起動する手順

1. (パッシブ / エイリアス Extract 構成にのみ有効) 静的 Collector プロセスを起動します。

   ```
   server h <host> -p <port>
   ```

   条件 :

   - -h <host> は、ソース・システムの名前または IP アドレスです。
   - -p <port は、そのシステムの Extract が Collector による接続のオープンをリスニングしているポート番号です。

2. ソース・システムの Oracle GoldenGate ディレクトリから GGSCI を実行します。

3. ソース・システムとターゲット・システムの GGSCI で、Manager を起動します。

```
START MANAGER
```

注意 Windows クラスタでは、クラスタ・アドミニストレータで Manager リソースを起動します。

4. ソース・システムとターゲット・システムで、起動するプロセスに応じて次のコマンド・セットのいずれかを発行します。プログラムは Oracle GoldenGate ディレクトリから実行します。

```
extract paramfile <name>.prm reportfile <name>.rpt [-p <port>]
```

または

```
replicat paramfile <name>.prm reportfile <name>.rpt
```

条件 :

- paramfile <name>.prm は、パラメータ・ファイルの相対名または完全修飾名です。コマンド名は、pf に短縮できます。
- reportfile <name>.rpt は、レポート・ファイルの相対名または完全修飾名です。コマンド名は、rf に短縮できます。
- -p <port> は、Extract が Collector による接続のオープンをリスニングしているローカル・ポート番号です。このオプションは、Extract をパッシブ・モードで起動する場合にのみ使用します。パッシブ / エイリアス Extract 構成の詳細は、106 ページを参照してください。

注意 バッチ・モードでは、プロセスの起動に GGSCI は使用されないため、エイリアス Extract は不要です。

第 14 章

Oracle データベースでの DDL 同期の構成

DDL 同期の概要

Oracle GoldenGate では、DDL 操作のデータベース間の同期がサポートされます。DDL 同期は次の場合にアクティブになります。

- ビジネス・アプリケーションがソース・オブジェクトとターゲット・オブジェクトにアクティブにアクセスして更新している場合。
- Oracle GoldenGate のトランザクション・データ同期がアクティブな場合。

DDL のレプリケーションをサポートするコンポーネントとトランザクション・データ変更 (DML) のレプリケーションをサポートするコンポーネントは相互に独立しています。したがって、次の同期を行えます。

- DDL 変更のみ
- DML 変更のみ
- DDL と DML の両方

たとえば、バッチ実行を使用してターゲット・オブジェクトを現在の状態に維持する場合、DDL 同期を継続的な (オンライン) 実行として構成すれば、バッチ・ロードの実行時にターゲット・メタデータを常に最新の状態に保つことができます。Oracle GoldenGate のバッチ・ロードでは、ソース・カタログとターゲット・カタログの現在のメタデータを使用します。

Oracle での DDL サポートでサポートされるオブジェクトと操作のリストは、『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。

Oracle GoldenGate DDL サポートの制限

DDL 文の長さ

Oracle GoldenGate では、DDL 文の長さが文字ではなくバイトで測定されます。サポートされる長さは約 2MB で、これには、対象となるオブジェクトの名前やその DDL タイプなどの特性に応じてサイズが変化する可能性のある内部的なオーバーヘッドが含まれます。DDL がサポートされるサイズよりも長い場合、Extract は警告を発行してその DDL 操作を無視します。

無視された DDL はマーカー表に保存されます。無視された Oracle DDL 文やその他の Oracle DDL 文は、ddl_ddl2file.sql スクリプト (DDL 操作を Oracle の USER_DUMP_DEST ディレクトリのテキスト・ファイルに保存するスクリプト) を使用して取得できます。スクリプトから次の入力を求められます。

- Oracle GoldenGate DDLオブジェクトが格納されているスキーマの名前(GLOBALSファイルに指定されています)。
- Oracle GoldenGate マーカー順序番号 (Extract パラメータ・ファイルで DDLOPTIONS に REPORT オプションが使用されている場合に Extract レポート・ファイルに記録されます)。
- 出力ファイルの名前。

システム構成

- Oracle GoldenGateでは、2つのシステム間でのみ一方向構成および双方向構成(アクティブ/パッシブ型とアクティブ / アクティブ型) の DDL レプリケーションがサポートされます。Oracle アクティブ / アクティブ構成に関する特別な考慮事項は、170 ページの「アクティブ / アクティブ (双方向) 構成での DDL の伝播」を参照してください。
- Oracle GoldenGate では、同類構成でのみ DDL 同期がサポートされます。Oracle GoldenGate DDL サポートには、次の要件があります。
  - ソースとターゲットのオブジェクト定義は同一である必要があります。
  - Replicat パラメータ・ファイルで ASSUMETARGETDEFS パラメータが使用されている必要があります。オブジェクトが DDL サポート用に構成され、SOURCEDEFS パラメータが使用されている場合、Replicat は異常終了します。ASSUMETARGETDEFS 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。

フィルタリング、マッピングおよび変換

DDL

DDL 操作は Oracle GoldenGate プロセスでは変換できません。単純な文字列置換を使用したり、次のようにスキーマ名とオブジェクト名をマップすることは可能です。

- プライマリの Extract または Replicat プロセスではソース DDL を別のターゲット・オブジェクトにマップしたりフィルタしたりできますが、データ・ポンプ Extract またはで DDL のマッピングやフィルタリングを行うことはできません。DDL は、PASSTHRU モード (マッピングまたはフィルタリング不可能) のデータ・ポンプまたはを通じて伝播されます。
- 結果として、特定の名前のソース表に対して実行される DDL (ALTER TABLE TableA... など) は、同じ表名 (ALTER TABLE TableA) でデータ・ポンプまたはによって処理されます。TABLE 文の指定に関係なく、そのプロセスでは ALTER TABLE TableB としてマップできません。

DML

DDL はデータ・ポンプまたはによって変更やマップされずに渡されるので、ソース表を別のターゲット名にマップする DML 操作を実行する場合は、この制限を考慮してください。プライマリの Extract または Replicat を使用して DML のフィルタリング、マッピングおよび変換を実行し、データ・ポンプまたはでこれらの表を PASSTHRU モードに構成します。

データ・ポンプまたはで、名前マッピングを含んだ DML 操作を実行する必要がある場合、これらの表に対するレプリケートされた DDL についても、同様の名前マッピングを実行するように Replicat を構成する必要があります。

注意 DDL サポートを使用しない表を NOPASSTHRU モードで構成すると、データのフィルタリングとデータ・ポンプによる操作が可能になります。

データの受渡し用に表を構成する手順

1. データ・ポンプまたはのパラメータ・ファイルで、DDL サポートを使用する表が含まれているすべての TABLE 文の前に PASSTHRU パラメータを指定します。

2. データのフィルタリング、マッピングまたは変換を行う場合は、同じパラメータ・ファイルで、DDL サポートを使用しない表が含まれている TABLE 文の前に NOPASSTHRU パラメータを指定できます。
3. データ・ポンプまたはに対して DDL 構成パラメータ (DDL、DDLOPTIONS、DDLSUBST、PURGEDDLHISTORY、PURGEMARKERHISTORY、DDLERROR) や DDL オプションを指定した Oracle GoldenGate トレース・パラメータを使用しないでください。

PASSTHRU 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。

SQLEXEC

- SQLEXEC のストアド・プロシージャまたは問合せの影響を受けるすべてのオブジェクトは、SQL の実行前に適切な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 の実行前に発生している必要があります。
- スタンドアロンの SQLEXEC 文の影響を受けるすべてのオブジェクトは、Oracle GoldenGate プロセスが起動する前に存在している必要があります。このため、DDL サポートは、これらのオブジェクトに対して無効にする必要があります。そうしないと、SQLEXEC のプロシージャまたは問合せが実行される前に、DDL 操作によって構造が変更されたり、オブジェクトが削除される可能性があります。

ユーザー・イグジット

DDL 操作 (DDL が実行されたオブジェクトに関する情報や DDL 文のテキスト自体など) を戻すようにするには、GET_DDL_RECORD_PROPERTIES 関数を使用します。Extract プロセスでは、ソース表のレイアウトのみ取得可能です。Replicat プロセスでは、ソースまたはターゲットのレイアウトを取得できます。

このユーザー・イグジットは取得機能のみを備えています。Oracle GoldenGate には DDL レコードを操作する関数はありません。

特別な DDL のケースとその処理

切捨て

TRUNCATE 文は次のようにレプリケートできます。

- Oracle GoldenGate のフル DDL サポート (TRUNCATE TABLE、ALTER TABLE TRUNCATE PARTITION などの DDL をサポート) の一環として (『Oracle GoldenGate *Windows and UNIX 管理者ガイド*』の説明を参照)。
- スタンドアロン TRUNCATE サポートとして。このサポートによってレプリケートできる DDL は TRUNCATE TABLE のみです。GETTRUNCATES パラメータでスタンドアロンの TRUNCATE 機能を制御します。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。

重複操作によるエラーを回避するため、これらのうち同時にアクティブにできる機能は 1 つのみです。

名前変更

- 表の RENAME 操作は、同等の ALTER TABLE RENAME に変換されます。たとえば、RENAME tab1 TO tab2 は、ALTER TABLE tab1 RENAME TO tab2 に変更されます。このような変換が行われるのは、RENAME ではスキーマ名の使用がサポートされず、ALTER TABLE RENAME ではサポートされるためです。スキーマ名がターゲットの DDL 文に含まれるように、Oracle GoldenGate で変換が行われます。変換は、Replicat プロセスのレポート・ファイルにレポートされます。
- 古い表名または新しい表名の長さが 18 文字 (名前に 16 文字、引用符に 2 文字) を超えると、ALTER TABLE RENAME は失敗します。ANSI の識別子制限のため、Oracle の名前変更で許可されるのは 18 文字のみです。
- 順序やビューのRENAME操作はALTER文には変換できません(順序とビューに対する同様の文がOracleに存在しないため)。したがって、順序名の変更は、常にソース DDL と同じ所有者およびオブジェクト名でターゲットにレプリケートされ、別の名前にはマップできません。

LOB 列

Extract ラグがあると、データ (DML) 操作がソース・オブジェクトで発生してから、Extract がその操作を REDO ログから取得するまでの間に、そのオブジェクトに対して DDL が実行される可能性があります。Extract はトランザクション・レコードを順次処理し、DDL と DML は両方ともログに記録されるため、通常、新しいメタデータは DML レコードが検出される前に解決されます。ただし、LOB の場合、Extract は状況に応じて LOB 値をフラッシュバック問合せからフェッチする必要があり、メタデータが正しい順序で提供されない可能性があります。

このような非一貫性は、Oracle が DDL (DROP 以外) に対するフラッシュバック機能を備えていないために発生します。LOB がフェッチされた時点で、オブジェクト構造は現在のメタデータを反映しますが、トランザクション・ログの LOB レコードは古いメタデータを反映しています。

このような構造上の差異を解決するため、Oracle GoldenGate では名前、型および長さの一致する共通の列セットを集約し、これらの列から LOB データがフェッチされます。その結果は次のとおりです。

- フェッチされたデータは、Extract によって処理されているデータよりも新しいか、または (削除された列の場合は) 存在しない可能性があります。
- DDL で列を削除してから同じ名前で再作成した場合、データ型が異なることがあります。(これは最悪のシナリオです)。この場合、トランザクション・レコード (古いデータ型) とデータベース・レコード (新しいデータ型) の間の非互換性によって、Replicat の処理エラーが発生する可能性があります。

LOB の非一貫性を防ぐ手順

- Extract ラグを短時間に抑えます (つまり、トランザクション量が少ないか存在しない場合に、Replicat が Oracle GoldenGate 証跡の DML を処理した後にのみ、LOB が含まれている表に対して DDL 操作を実行します)。ラグを短縮するための推奨事項は、『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。
- Replicat で処理される DML 操作が残っている場合、LOB が含まれている表で Oracle GoldenGate が行識別子として使用している列に対して DDL を実行しないでください。行識別子は、主キー列、一意キーが定義されている列、または TABLE あるいは MAP パラメータの KEYCOLS 句で代替キーとして構成されている列です。これらの識別子のいずれも存在しない場合、その行内のすべての列が行識別子になります。識別子の列に対して DDL を実行する必要がある場合、次の手順を実行します。

LOB が含まれている表の行識別子に対して DDL を実行する手順

1. ソースの DML 操作を一時停止します。
2. Replicat ですべての証跡データの処理が終了するまで待機します。Replicat の終了を確認するには、処理するデータがなくなったことを示すメッセージが表示されるまで次のコマンドを発行します。

   ```
   INFO REPLICAT <group>
   ```

3. ソースに対して DDL を実行します。
4. ソースの DML 操作を再開します。

ユーザー定義タイプ

- ユーザー定義タイプに関連する DDL 操作では、ソースとターゲットの両方に暗黙的な DML 操作が発生します。重複操作による SQL エラーを回避するため、Oracle GoldenGate ではこのような DML 操作はレプリケートされません。
- ユーザー定義タイプに関する DML をレプリケートする場合、Extract ではオブジェクトに対して DDL を実行する前にそれらの変更をすべて処理する必要があります。UDT データは Extract によってフェッチされることがあるため、LOB 列に適用するのと同じ理由から、このルールが必要になります。(「LOB 列」の項を参照してください。)

変更データが取得される場合に Oracle UDT に対して DDL を実行する手順

1. オブジェクトに対する DML 操作を停止します。
2. ソース・オブジェクトとターゲット・オブジェクトの比較を、両方が同一になるまで継続します。これにより、Extract がトランザクション・ログから残りのデータ変更を取得して、ターゲットに送信したことが保証されます。
3. DDL を実行します。
4. DML 操作を再開します。

SQL のコメント

ソース DDL 文でオブジェクト名の途中にコメントが含まれる場合、ターゲット DDL 文ではそのコメントはオブジェクト名の最後に表示されます。次に例を示します。

ソース	ターゲット
`CREATE TABLE hr./*comment*/emp ...`	`CREATE TABLE hr.emp /*comment*/ ...`

これは、DDL 同期の整合性に影響しません。DDL 文の他の箇所のコメントは、レプリケート時も同じ場所に表示されます。

コンパイル・エラー

トリガー、プロシージャ、関数またはパッケージに対する CREATE 操作でコンパイル・エラーが発生しても、Oracle GoldenGate はターゲットに対してその DDL 操作を実行します。厳密に言うと、DDL 操作自体は正常に完了しているため、再帰プロシージャなどでターゲットに対する依存性の実行が可能になるよう、これらを伝播する必要があります。

時間隔パーティション化

DDL が暗黙的であるため、DDL レプリケーションは時間隔パーティション化の影響を受けません。

DDL サポートに関する構成のガイドライン

データベースの権限

Oracle GoldenGate で DDL の取得とレプリケーションをサポートするために必要なデータベース権限は、『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。

初期同期

- DDL レプリケーションを構成するには、ソース・データベースと同期しているターゲット・データベースから作業を開始します。DDL サポートは、Replicat の初期ロード方法と互換性があります。
- 初期ロードを実行する前に、DDL の抽出およびレプリケーションを無効化します。DDL 処理は、Extract および Replicat のパラメータ・ファイルの DDL パラメータによって制御されます。
- ソース・データとターゲット・データの初期同期ができたら、ソース・アプリケーションを実行する前に、NEXTVAL ですべてのソース順序値を少なくとも 1 回使用します。システム内の各順序から NEXTVAL を選択するスクリプトを使用できます。これは、Extract の実行中に行われる必要があります。

プロセス・トポロジ

- Extract または Replicat (あるいはその両方) のパラレル・プロセスを使用する場合、関連する DDL と DML を同じプロセス・ストリーム内にまとめ、データの整合性を保証します。プロセスは次のように構成します。
  - 任意のオブジェクトに対するすべての DDL と DML を、同じ Extract グループおよび同じ Replicat グループで処理します。
  - 相互に関連するすべてのオブジェクトを同じプロセス・グループで処理します。

  たとえば、ReplicatA で Table1 に対する DML を処理する場合、Table1 に対する DDL も処理する必要があります。Table2 に Table1 の外部キーがある場合、その DML 操作と DDL 操作も ReplicatA で処理される必要があります。
- Extract グループで、異なる Replicat グループによって読み取られる複数の証跡に書き込む場合、Extract はすべての DDL をすべての証跡に送信します。各 Replicat グループを使用して DDL をフィルタするには、Replicat パラメータ・ファイルで DDL パラメータのフィルタ・オプションを使用します。

オブジェクト名

- Oracle GoldenGate では、マルチバイト・キャラクタや特殊な英数字 (!、$、# など) を含むオブジェクト名がサポートされます。制限が適用されるのは、オブジェクトが TABLE パラメータまたは MAP パラメータでマップされる場合です (これらのパラメータでは、一部の使用可能な特殊文字がサポートされないため)。MAP 文と TABLE 文のオブジェクトに対する DDL では、これらのパラメータの制限が継承されます。これらのパラメータの詳細は、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』の MAP および TABLE の説明を参照してください。
- Oracle GoldenGate の標準のアスタリスク・ワイルドカード (*) を使用して、DDL 同期をサポートする構成パラメータでオブジェクト名を指定できます。ワイルドカードを正しく処理するため、WILDCARDRESOLVE パラメータはデフォルトで DYNAMIC に設定されます。WILDCARDRESOLVE を他の設定にすると、DDL 操作を処理している Oracle GoldenGate プロセスが異常終了して、プロセス・レポートにエラーが書き込まれます。
- DDL 同期をサポートする構成パラメータでは、アスタリスク・ワイルドカード (*) を使用して Oracle スキーマ名を指定できます。この機能はデフォルトでは無効です。有効化するには、GLOBALS パラメータ・ファイルで _ALLOWWILDCARDSCHEMAS パラメータを使用します。これは非公開のパラメータです。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。

- Oracle GoldenGate では、DDL 文で所有者とオブジェクト名を区切るドットの前、後または前後両方に空白を使用できます。ドットの両側に使用できる空白は 1 つのみです。たとえば、次が有効です。

```
CREATE TABLE fin . customers...
CREATE TABLE fin. customers...
CREATE TABLE fin .customers...
```

CREATE または RENAME の後のデータ継続性

CREATE 操作または RENAME 操作の結果として得られる新しい Oracle 表に対する DML 操作をレプリケートするには、その新規表の名前がパラメータ・ファイルの TABLE 文と MAP 文で指定されている必要があります。ワイルドカードを使用することで、それらを確実に含めることができます。

CREATE USER を使用して新規ユーザーを作成し、そのスキーマに新規表または名前の変更された表を移動するには、その新規ユーザー名が TABLE 文と MAP 文で指定されている必要があります。新規ユーザー fin2 を作成し、そのスキーマに新規表または名前の変更された表を移動する場合、ターゲットの同じスキーマと異なるスキーマのどちらに fin2 オブジェクトをマップするかに応じて、パラメータ文は次のようになります。

Extract:

```
TABLE fin2.*;
```

Replicat:

```
MAP fin2*, TARGET <different_schema>.*;
```

DDL スコープの理解

データベース・オブジェクトは*スコープ*に分類されます。スコープとは、オブジェクトに対する DDL 操作を Oracle GoldenGate で処理する方法を定義するカテゴリです。次のスコープがあります。

- MAPPED
- UNMAPPED
- OTHER

スコープを使用することで、DDL 操作のフィルタリング、文字列置換およびエラー処理を詳細に制御できます。

MAPPED スコープ

TABLE 文と MAP 文で指定されるオブジェクトは、*MAPPED スコープ*です。これらの文の抽出指示とレプリケーション指示は、オーバーライド・ルールが適用されないかぎり、指定したオブジェクトに対するデータ (DML) と DDL の両方に適用されます。

TABLE 文と MAP 文のオブジェクトでは、次の表にリストされている DDL 操作がサポートされます。

表 14　MAP 文と TABLE 文でマップできるオブジェクト

操作	オブジェクト[1]
CREATE	TABLE (AS SELECT を含む)
ALTER	INDEX
DROP	TRIGGER
RENAME	SEQUENCE
COMMENT ON[2]	MATERIALIZED VIEW
	VIEW
	FUNCTION
	PACKAGE
	PROCEDURE
	SYNONYM
	PUBLIC SYNONYM[3]
GRANT	TABLE
REVOKE	SEQUENCE
	MATERIALIZED VIEW
ANALYZE	TABLE
	INDEX
	CLUSTER

[1] TABLE および MAP では、これらの操作の対象となるオブジェクト名に使用される可能性のある一部の特殊文字がサポートされません。このような文字のリストは、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』の MAP パラメータと TABLE パラメータの説明を参照してください。サポートされない特殊文字が含まれるオブジェクトは、UNMAPPED スコープと OTHER スコープでサポートされます。

[2] COMMENT ON TABLE、COMMENT ON COLUMN に適用されます。

[3] 表名はスキーマ名で修飾する必要があります。

Extract の場合、MAPPED スコープでは TABLE 文の指示に従ってオブジェクトが DDL 取得用にマークされます。Replicat の場合、MAPPED スコープでは DDL がレプリケーション用にマークされ、MAP 文の TARGET 句のスキーマと名前で指定されたオブジェクトにマップされます。このマッピングを実行するため、Replicat は ALTER SESSION を発行して、Replicat セッションのスキーマを TARGET 句で指定されたスキーマに設定します。DDL に未修飾のオブジェクトが含まれる場合、ターゲットに割り当てられるスキーマは、151 ページの「DDL の未修飾のオブジェクト名の適切な識別」に記載されている状況に応じて異なります。

次の TABLE 文と MAP 文があるとします。

Extract (ソース)	**Replicat (ターゲット)**
`TABLE fin.expen;`	`MAP fin.expen, TARGET fin2.expen2;`
`TABLE hr.tab*;`	`MAP hr.tab*, TARGET hrBackup.bak_*;`

さらに、次のソース DDL 文があるとします。

```
ALTER TABLE fin.expen ADD notes varchar2(100);
```

この例では、別の所有者と表名にマップする TARGET 句が指定された MAP 文にソース表 fin.expen があるため、ターゲットの DDL 文は次のようになります。

```
ALTER TABLE fin2.expen2 ADD notes varchar2(100);
```

同様に、この例の TABLE 文と MAP 文の 2 番目のセットに対しては、次のソースおよびターゲットの DDL 文を使用できます。

ソース : `CREATE TABLE hr.tabPayables ... ;`

ターゲット : `CREATE TABLE hrBackup.bak_tabPayables ...;`

MAPPED スコープのオブジェクトでは、DDL サポートを詳細に調整しない場合、DDL 構成パラメータからオブジェクト名を省略できます。TABLE 文と MAP 文のオブジェクト名を変更する必要がある場合、それらのオブジェクトに対する DDL に変更が自動的に適用されます。

オブジェクトが TABLE 文に含まれ、MAP 文に含まれない場合、そのオブジェクトに対する DDL のスコープは、ソースでは MAPPED ですがターゲットでは UNMAPPED です。

Oracle クラスタ表と UDT のマッピング

Oracle クラスタ表または Oracle ユーザー定義タイプ (UDT) は、別のターゲット名にはマップできませんが、別のターゲット所有者にはマップできます。これらの特別なオブジェクトは、それ自体に MAPPED と UNMAPPED のスコープが混在する可能性のある基になる表で構成されるため、名前マッピングは使用できません。

ALTER INDEX のマッピング

ALTER INDEX...RENAME コマンドは、別のターゲット索引名にはマップできませんが、別のターゲット所有者にはマップできます。

有効な例 :

```
ALTER INDEX src.ind RENAME TO indnew;
```

この DDL は、ワイルドカードを使用して次のようにマップできます。

```
MAP src.* TARGET tgt.*;
```

また、次のように元の索引名をソースとターゲットの指定に使用し、明示的にマップすることもできます。

```
MAP src.ind TARGET tgt.ind;
```

前述のいずれの場合も、ターゲット DDL は次のようになります。

```
ALTER INDEX tgt.ind RENAME TO indnew;
```

無効な例 :

次のような MAP 文は無効です。

```
MAP src.ind TARGET tgt.indnew;
```

この文では、古い名前が新しい名前にマップされ、ターゲット DDL は次のようになります。

```
ALTER INDEX tgt.indnew RENAME TO indnew;
```

UNMAPPED スコープ

TABLE 文または MAP 文で DDL 操作の使用がサポートされていて、そのベース・オブジェクト名がこれらのパラメータのいずれかに含まれない場合は、*UNMAPPED スコープ*です。

オブジェクト名のスコープが、ソースでは UNMAPPED (Extract の TABLE 文にない)、ターゲットでは MAPPED (Replicat の MAP 文にある) になることも、その逆になることもあります。Oracle DDL のスコープが Replicat 構成で UNMAPPED の場合、Replicat はデフォルトで次の処理を実行します。

1. Replicat セッションの現在の所有者をソース DDL オブジェクトの所有者に設定します。
2. その所有者として DDL を実行します。
3. Replicat セッションの現在の所有者として Replicat をリストアします。

151 ページの「DDL の未修飾のオブジェクト名の適切な識別」も参照してください。

OTHER スコープ

マップできない DDL 操作は、*OTHER スコープ*です。DDL のスコープが Replicat 構成で OTHER の場合、ソース DDL と同じ所有者とオブジェクト名を使用してターゲットに適用されます。

OTHER スコープには、たとえば、システム固有の参照を作成する DDL 操作 (データ・ファイル名を操作する DDL など) があります。

OTHER スコープのその他の例を次に示します。

```
CREATE USER joe IDENTIFIED by joe;
CREATE ROLE ggs_gguser_role IDENTIFIED GLOBALLY;
ALTER TABLESPACE gg_user TABLESPACE GROUP gg_grp_user;
```

151 ページの「DDL の未修飾のオブジェクト名の適切な識別」も参照してください。

DDL の未修飾のオブジェクト名の適切な識別

Oracle DDL には、スキーマ名で修飾されていないオブジェクト名が含まれる場合があります。たとえば、次の DDL では CREATE TABLE 句の TAB 表は修飾されていません。

```
ALTER SESION SET CURRENT_SCHEMA = SRC;
CREATE TABLE tab (X NUMBER);
CREATE TABLE SRC1.tab (X NUMBER) AS SELECT * FROM tab;
```

デフォルトで、Oracle DDL 文の未修飾のオブジェクトには、次のいずれかに相当するセッション・スキーマが想定されます。

- SQL セッションを開始したユーザーのスキーマ。
- ALTER SESSION SET CURRENT_SCHEMA コマンドで設定されたスキーマ。前述の例で、SRC は未修飾の TAB 表の所有者になります。

未修飾のオブジェクトが含まれている DDL をレプリケートするため、Replicat は次の処理を実行します。

- 未修飾のオブジェクトのスコープが MAPPED (その名前が MAP の指定に一致する) 場合、Replicat は次のいずれかを実行します。
  - 未修飾のオブジェクトの実際のスキーマがソース・セッション・スキーマと同じ場合、Replicat はスキーマを MAP 文の TARGET 句で指定されたスキーマに設定します。
  - 未修飾のオブジェクトの実際のスキーマがソース・セッション・スキーマと異なる場合、Replicat はスキーマをソース・セッション・スキーマに設定します。
- 未修飾のオブジェクトのスコープが UNMAPPED または OTHER の場合、Replicat はスキーマをソース・セッション・スキーマに設定します。

ソース・セッション・スキーマは、別のターゲット・セッション・スキーマにマップできます。一部の DDL (CREATE TABLE AS SELECT など) がターゲットで成功するためには、セッション・スキーマのマッピングが必要です。このマッピングはグローバルで、同じスキーマ名を含む他のすべてのマッピングをオーバーライドします。セッション・スキーマをマップするには、DDLOPTIONS パラメータに MAPSESSIONSCHEMA オプションを使用します。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。

スキーマ・マッピングが明示的に行われないようにするには、DDLOPTIONS パラメータの NOEXPLICITSCHEMAMAPPING オプションを使用します。詳細は、『*Windows and UNIX リファレンス・ガイド*』の DDLOPTIONS パラメータの説明を参照してください。

DDL サポートの有効化

デフォルトでは、DDL レプリケーション・サポートのステータスは次のとおりです。

- ソースでは、Oracle GoldenGate DDL サポートはデフォルトで無効ですが、DDL パラメータを使用して、DDL を取得するように Extract を構成する必要があります。
- ターゲットでは、レプリケートされるトランザクション・データの整合性を保持するために、DDL サポートはデフォルトで有効です。デフォルトで、Replicat は証跡に含まれるすべての DDL 操作を処理します。必要に応じて DDL パラメータを使用し、DDL 操作を無視またはフィルタするように Replicat を構成できます。

DDL レプリケーションのフィルタリング

要件に応じて特定 (またはすべて) の DDL がターゲット・データベースに適用されるように DDL 操作をフィルタするには、Oracle データベースでは、次の方法を使用すれば、要件に応じて特定 (またはすべて) の DDL がターゲット・データベースに適用されるように DDL 操作をフィルタできます。

- *ソース・システムでの DDL トリガーによるフィルタ*。この方法では、DDL 操作の発生をトリガーとして実行時にコールされる Oracle 関数を使用します。DDL に関する情報はこの関数に渡され、これを使用して、DDL を Extract に渡すかどうかを判断できます。(デフォルトでは、すべての DDL が Extract に渡されます。) この方法では、Extract へ送信する DDL 操作を少なくすることで取得のパフォーマンスを改善でき、また、処理の早い段階でフィルタリングが必要な他の目的でも使用できます。
- *ソースまたはターゲット、あるいはその両方での DDL パラメータによるフィルタ*。この方法は Oracle GoldenGate 内で実行され、Extract と Replicat の両方でフィルタ基準を実行できます。Extract でフィルタリングを行うか、すべての DDL を証跡に送り、Replicat でフィルタリングを行います。あるいは、異なる場所の組合せでフィルタすることもできます。DDL パラメータでは、

フィルタリングを行う場所を制御でき、DDL スコープに基づいてまとめてフィルタする (たとえば、すべての MAPPED スコープを含める) 機能など、トリガーによる方法よりも多くのフィルタリング・オプションが用意されています。

- *トリガーと DDL パラメータによるフィルタの組合せ*。DDL トリガーによるフィルタリングを行ってから Extract に渡された DDL を、DDL パラメータを使用して特定のニーズに合せてさらにフィルタできます。

トリガー・レベルのフィルタリング

DDL トリガーのレベルで DDL をフィルタするには、次の手順を実行します。

1. Oracle GoldenGate のインストール・ディレクトリにある ddl_filter.sql ファイルを、これから記述するコードをテストできるテスト・マシンにコピーします。
2. ファイルを編集用に開きます。filterDDL という名前の PL/SQL 関数が含まれているので、これを変更してフィルタ基準を指定できます。この関数に渡される情報には、DDL オブジェクトの所有者、オブジェクト名、オブジェクト・タイプ、操作タイプなどがあります。DDL を実行したユーザーは、ora_login_user 変数に含まれています。Extract の処理に包含するか除外するかについて、DDL のタイプごとにフィルタ・コードを記述します。
3. (オプション) stmt 変数の先頭 30K の DDL テキストを処理する場合は、getStatement 変数を YES に設定するコードを記述します。デフォルトでは、不要なオーバーヘッドを防ぐため、DDL テキストは処理されません。
4. retVal 変数が INCLUDE または EXCLUDE のどちらであるか判別するコードを記述します。この値によって、DDL 操作を Extract に渡すかどうかが決まります。デフォルトは INCLUDE です。
5. コードを保存します。
6. テスト・システムで DDL アクティビティを停止します。
7. ddl_filter.sql ファイルを次のようにコンパイルします。

   ```
   @ddl_filter schema_name
   ```

   条件 : schema_name 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です。これらのオブジェクトの詳細は、『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。
8. テスト環境でテストし、フィルタリングが機能することを確認します。コード内のエラーによってソースとターゲットの DDL が同期しなくなることがあるため、このテストを実行することは重要です。
9. テストが成功したら、本番のソース・システムで Oracle GoldenGate のインストール・ディレクトリにファイルをコピーします。
10. ソース・システムで DDL アクティビティを停止します。

11. 前に行ったように ddl_filter.sql ファイルをコンパイルします。

```
@ddl_filter schema_name
```

12. ソース・システムで DDL アクティビティを再開します。

DDL パラメータを使用したフィルタリング

DDL パラメータは、Extract プロセスと Replicat プロセス内で DDL をフィルタするための主要な Oracle GoldenGate パラメータです。

オプションなしで DDL パラメータを使用すると、フィルタリングは行われず、すべての DDL 操作が次のように伝播されます。

- (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・オブジェクトに対して生成された、サポートされているすべての DDL 操作を取得し、証跡に送信します。
- (Replicat パラメータとして) Oracle GoldenGate の証跡からすべての DDL 操作をレプリケートし、ターゲットに適用します。これは、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。

オプションを指定して使用すると、DDL パラメータはフィルタリング・エージェントとして機能し、次に基づいて DDL 操作を包含または除外します。

- スコープ
- 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、あるいはその両方の文字列

パラメータ・ファイルで使用できる DDL パラメータは 1 つのみですが、複数の包含オプションと除外オプションを組み合せることで、必要なレベルまで DDL をフィルタできます。

- 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 に対しては有効ですが、データ・ポンプ Extract に対しては無効です。
- 組み合せることで、複数のフィルタ・オプションの指定は AND 文として論理的に連結されます。
- DDL 文をレプリケートするには、複数のオプションを使用して指定されたフィルタ基準がすべて満たされる必要があります。
- 複雑な DDL フィルタリング基準を使用する場合、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。

警告 **Oracle GoldenGate によってインストールされた DDL オブジェクトは、DDL パラメータ、TABLE パラメータ、MAP パラメータ、TABLEEXCLUDE パラメータ、または MAPEXCLUDE パラメータに含めないでください。これらのパラメータにワイルドカードを指定する場合、Oracle GoldenGate によってインストールされた DDL オブジェクトが含まれていないことを確認します。これらのオブジェクトは Oracle GoldenGate 構成に含めないようにする必要はありますが、Extract プロセスではこれらに対する操作を認識している必要があるので、EXCLUDE、TABLEEXCLUDE または MAPEXCLUDE パラメータ文を使用して構成から明示的に除外することはできません。**

注意 **DDL パラメータ文を作成する前に、この章の「処理における DDL の評価方法」を確認すると役立ちます。**

構文

```
DDL [
{INCLUDE | EXCLUDE}
    [, MAPPED | UNMAPPED | OTHER | ALL]
    [, OPTYPE <type>]
    [, OBJTYPE '<type>']
    [, OBJNAME "<name>"]
    [, INSTR '<string>']
    [, INSTRCOMMENTS '<comment_string>']
]
[...]
```

表 15　DDL の包含オプションと除外オプション

オプション	説明

`INCLUDE | EXCLUDE`	INCLUDE および EXCLUDE を使用して、包含句または除外句の開始を示します。
	◆ 包含句には、このパラメータの対象となる DDL を識別するフィルタリング基準が含まれます。
	◆ 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が含まれます。
	包含句または除外句は、INCLUDE または EXCLUDE キーワードの後に、適用されるパラメータの有効なオプションの組合せを指定して構成する必要があります。
	
	EXCLUDE を使用する場合、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.*"`
	ただし、次のいずれかは使用できます。
	`DDL INCLUDE ALL, EXCLUDE OBJNAME "hr.*"`
	`DDL INCLUDE OBJNAME "fin.*" EXCLUDE "fin.ss"`
	EXCLUDE は、同じ基準が含まれている INCLUDE よりも優先されます。複数の包含句と除外句を使用できます。

`MAPPED | UNMAPPED | OTHER | ALL`	DDL 操作のスコープに基づいて INCLUDE または EXCLUDE を適用するには、MAPPED、UNMAPPED、OTHER および ALL を使用します。 ◆ MAPPED では、INCLUDE または EXCLUDE が MAPPED スコープの DDL 操作に適用されます。MAPPED フィルタリングは、他の DDL パラメータ・オプションを使用して指定されたフィルタリングの前に実行されます。 ◆ UNMAPPED では、INCLUDE または EXCLUDE が UNMAPPED スコープの DDL 操作に適用されます。 ◆ OTHER では、INCLUDE または EXCLUDE が OTHER スコープの DDL 操作に適用されます。 ◆ ALL では、INCLUDE または EXCLUDE がすべてのスコープの DDL 操作に適用されます。

表 15　DDL の包含オプションと除外オプション (続き)

オプション	説明
`OPTYPE <type>`	INCLUDE または EXCLUDE を特定のタイプの DDL 操作 (CREATE、ALTER、RENAME など) に適用するには、OPTYPE を使用します。<type> には、データベースに有効な任意の DDL コマンドを使用します。たとえば、ALTER 操作を含める場合の正しい構文は次のようになります。 `DDL INCLUDE OPTYPE ALTER`

`OBJTYPE '<type>'`	INCLUDE または EXCLUDE を特定のタイプのデータベース・オブジェクトに適用するには、OBJTYPE を使用します。<type> には、データベースに有効な任意のオブジェクト・タイプ (TABLE、INDEX、TRIGGER など) を使用します。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・ログの場合、正しいタイプはそれぞれ snapshot と 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 には OBJNAME オプションを使用しないでください (OBJNAME には owner.object が必要ですが、USER はスキーマのみを持つためです)。

`OBJNAME "<name>"`	INCLUDE または EXCLUDE をオブジェクトの完全修飾名 (owner.table_name など) に適用するには、OBJNAME を使用します。このオプションでは、二重引用符で囲まれた文字列を入力として使用します。
	ワイルドカードは、オブジェクト名にのみ使用できます。
	例 :
	`DDL INCLUDE OBJNAME "accounts.*"`
	Oracle USER オブジェクトには OBJNAME を使用しないでください (OBJNAME には owner.object が必要ですが、USER はスキーマのみを持つためです)。
	Replicat パラメータ・ファイルで OBJNAME と MAPPED を組み合せて使用する場合、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.*" です。
	`MAP fin.exp_*, TARGET fin2.*;`
	次の例では、CREATE TABLE 文はソースで次のように実行されます。
	`CREATE TABLE fin.exp_phone;`
	ターゲットでは次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たデータベース・ユーザーにマップします。

表 15　DDL の包含オプションと除外オプション (続き)

オプション	説明

	トリガー、シノニムおよび索引を作成する DDL の場合、OBJNAME の値はトリガー、シノニムまたは索引の名前ではなく、ベース・オブジェクトの名前にする必要があります。
	たとえば、次の DDL 文を含める場合、正しい値は hr.insert_trig ではなく hr.accounts です。
	`CREATE TRIGGER hr.insert_trig ON hr.accounts;`
	RENAME 操作では、OBJNAME の値を新しい表名にする必要があります。たとえば、次の DDL 文を含める場合、正しい値は hr.acct です。
	`ALTER TABLE hr.accounts RENAME TO acct;`

`INSTR '<string>'`	INCLUDE または EXCLUDE を、コマンド構文内に特定の文字列を含む (ただしコメント内には含まない) DDL 文に適用するには、INSTR を使用します。たとえば、次の例では、索引を作成する DDL は除外されます。 `DDL INCLUDE ALL EXCLUDE INSTR 'CREATE INDEX'` 文字列は一重引用符で囲みます。文字列の検索では、大 / 小文字は区別されません。 INSTR では、文字列内に含まれる一重引用符 (' ') も、NULL 値もサポートされません。

`INSTRCOMMENTS '<comment_string>'`	INCLUDE または EXCLUDE を、コメント内に特定の文字列を含む (DDL コマンド自体には含まない) DDL 文に適用するには、INSTRCOMMENTS を使用します。INSTRCOMMENTS を使用すると、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、次の例では、コメントに source を含む DDL 文は除外されます。
	`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。文字列の検索では、大 / 小文字は区別されません。INSTR と INSTRCOMMENTS を組み合せることで、同じ DDL 文のコマンド構文内とコメント内の文字列をフィルタできます。
	INSTRCOMMENTS では、文字列内に含まれる一重引用符 (' ') も、NULL 値もサポートされません。

表 15　DDL の包含オプションと除外オプション (続き)

オプション	説明

`INSTRWORDS '<word list>'`	INCLUDE または EXCLUDE を特定の語を含む DDL 文に適用するには、INSTRWORDS を使用します。
	<word list> には、一重引用符内に任意の順序で語を指定します。空白を含めるには、空白を (語がある場合は語も) 二重引用符で囲みます。二重引用符は、文を囲む場合にも使用できます。
	INSTRWORDS が有効になるには、指定された語がすべて DDL に存在する必要があります。
	例 :
	`ALTER TABLE INCLUDE INSTRWORDS 'ALTER CONSTRAINT " xyz"`
	この例は、次に一致します。
	`ALTER TABLE ADD CONSTRAINT xyz CHECK`
	および
	`ALTER TABLE DROP CONSTRAINT xyz`
	INSTRWORDS では、文字列内に含まれる一重引用符 (' ') も、NULL 値もサポートされません。

`INSTRCOMMENTSWORDS '<word list>'`	INSTRWORDS と同様に機能しますが、DDL 文内のコメントにのみ適用され、DDL 構文自体には適用されません。INSTRCOMMENTS を使用すると、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、文字列内に含まれる一重引用符 (' ') も、NULL 値もサポートされません。 INSTRWORDS と INSTRCOMMENTSWORDS を組み合せることで、同じ DDL 文のコマンド構文内とコメント内の文字列をフィルタできます。

DDL パラメータ・オプションの組合せ

次に、DDL パラメータ・オプションの組合せ方の例を示します。

```
DDL  &
INCLUDE UNMAPPED &
    OPTYPE alter &
    OBJTYPE 'table' &
    OBJNAME "users.tab*" &
INCLUDE MAPPED OBJNAME "*" &
EXCLUDE MAPPED OBJNAME "temporary.tab*"
```

この文で組み合されたフィルタ基準では、次のように指定されます。

- TABLE文またはMAP文でマップされていない(UNMAPPEDスコープ)表に対するすべてのALTER TABLE文がINCLUDE されます。
  - ただし、表が users によって所有され、その名前が tab で始まる場合のみです。
- TABLE文またはMAP文でマップされている(MAPPEDスコープ)すべての表に対するすべてのDDL操作タイプがINCLUDE されます。
- MAPPED スコープのすべての表に対するすべての DDL 操作タイプが EXCLUDE されます。
  - ただし、これらの表が temporary によって所有される場合のみです。
  - かつ、その名前が tab で始まる場合のみです。

特別なフィルタのケース

次に、フィルタ条件を作成する場合に注意する必要がある特別なケースを示します。

DDL EXCLUDE ALL

DDL EXCLUDE ALL は、DDL 操作自体のレプリケーションは行わずに、Oracle GoldenGate のオブジェクト・メタデータを最新に保つ特別な処理オプションです。Oracle GoldenGate 以外の方法を使用して DDL をターゲットに適用し、ターゲット・オブジェクトへのデータ変更を Oracle GoldenGate でレプリケートする場合に、DDL EXCLUDE ALL を使用できます。現在のメタデータはオブジェクトの変更として Oracle GoldenGate に提供されるため、Oracle GoldenGate プロセスを停止したり起動する必要がありません。DDL EXCLUDE ALL には次の特別な条件が適用されます。

- DDL EXCLUDE ALL では、INCLUDE 句を使用する必要はありません。
- DDL EXCLUDE ALL を使用する場合、WILDCARDRESOLVE パラメータを IMMEDIATE に設定すると、必要に応じて即座に DML を解決できるようになります。

DDL メタデータと操作がすべてレプリケートされないようにするには、DDL パラメータ全体を省略します。DDL トリガーによる履歴表への DDL 操作の記録は、手動で無効化しないかぎり継続されます。

暗黙的 DDL

ユーザーが生成した DDL 操作によって暗黙的 DDL 操作が生成される場合があります。たとえば、次の文では、Oracle DDL トリガーによって 2 つの異なる DDL 操作が処理されます。

```
CREATE TABLE customers (custID number, name varchar2(50), address varchar2(75), address2
varchar2(75), city varchar2(50), state (varchar2(2), zip number, contact varchar2(50),
areacode number(3), phone number(7), primary key (custID));
```

- 最初の (明示的) DDL 操作は、CREATE TABLE 文自体です。
- 2 番目の DDL 操作は、暗黙的な CREATE UNIQUE INDEX 文で、主キーの索引を作成します。この操作は、ユーザー・アプリケーションではなく、データベース・エンジンによって生成されます。

暗黙的 DDL のフィルタリングのガイドライン

*DDL パラメータ*を使用して DDL 操作をフィルタする場合、ターゲットでは明示的 DDL によって暗黙的 DDL が生成されるため、Oracle GoldenGate はデフォルトで暗黙的 DDL を除外します。たとえば、前述の例の CREATE TABLE 文が Replicat によって適用されると、ターゲット・データベースでは適切な索引が作成されます。

ただし、*DDL トリガー*を使用して DDL 操作をフィルタする場合、暗黙的 DDL は次に基づいてフィルタ・ルールで処理する必要があります。

- フィルタ・ルールで明示的 DDL を伝播対象から除外する場合、暗黙的 DDL を除外するルールも作成する必要があります。たとえば、前述の例の CREATE TABLE 文を除外しながら、CREATE UNIQUE INDEX 文は除外していない場合、ターゲット・データベースは存在しない表に対して索引を作成しようとします。
- フィルタリング・ルールで明示的 DDL の伝播が許可されている場合、暗黙的 DDL を除外する必要はありません。Oracle GoldenGate とターゲット・データベースで適切に処理されます。

Oracle GoldenGate による導出オブジェクト名の処理方法

DDL 操作には、*ベース・オブジェクト*名に加え、*導出オブジェクト*名を含めることができます。ベース・オブジェクトは、データが格納されたオブジェクトです。導出オブジェクトは、ベース・オブジェクトの一部の属性を継承し、そのオブジェクトに関連する機能を実行するオブジェクトです。ベース・オブジェクトと導出オブジェクトの両方を含む DDL 文は次のとおりです。

- RENAME および ALTER RENAME
- 索引、シノニム、またはトリガーに対する CREATE および DROP

次の DDL 文について考えてみます。

```
CREATE INDEX hr.indexPayrollDate ON TABLE hr.tabPayroll (payDate);
```

この場合、表がベース・オブジェクトです。その名前 (hr.tabPayroll) が*ベース名*で、MAPPED スコープでの TABLE または MAP によるマッピングの対象です。導出オブジェクトは索引で、その名前 (hr.indexPayrollDate) が*導出名*です。

導出名は、ベース・オブジェクトとは別に、独自の TABLE 文または MAP 文でマップできます。または、1 つの MAP 文で両方を処理できます。MAP の場合、ターゲットでの導出オブジェクト名の変換は次のように処理されます。

ベース・オブジェクトに対する MAP はあるが、導出オブジェクトにはない場合

ベース・オブジェクトに対する MAP 文はあるが、導出オブジェクトにはない場合、導出オブジェクトの*暗黙的マッピング*が行われます。DDL 文に MAPPED が含まれる場合、Replicat はベース・オブジェクトと同じターゲット所有者を導出オブジェクトに割り当てます。導出オブジェクトの名前は、ソース文と同じです。たとえば、次のとおり仮定します。

Extract (ソース)	**Replicat (ターゲット)**
`TABLE hr.tab*;`	`MAP hr.tab*, TARGET hrBackup.*;`

次のソース DDL 文があるとします。

```
CREATE INDEX hr.indexPayrollDate ON TABLE hr.tabPayroll (payDate);
```

ターゲットで Replicat によって実行される CREATE INDEX 文は次のとおりです。

```
CREATE INDEX hrBackup.indexPayrollDate ON TABLE hrBackup.tabPayroll (payDate);
```

暗黙的マッピングのルールは、導出オブジェクトにベース・オブジェクトと同じ所有者を割り当てるという一般的な方法に基づきます。導出オブジェクトの名前がソースの文で完全修飾されていない場合でも、名前が適切に変換されることが保証されます。また、ベース・オブジェクトと同じターゲット所有者が索引を所有している場合、暗黙的マッピングで導出オブジェクト名を明示的にマップする必要はありません。

ベース・オブジェクトと導出オブジェクトに対する MAP がある場合

ベース・オブジェクトに対しても、導出オブジェクトに対しても MAP 文がある場合、*明示的マッピング*が行われます。DDL 文に MAPPED が含まれる場合、Replicat は独自の TARGET 句に従って各オブジェクトの所有者と名前を変換します。たとえば、次のとおり仮定します。

Extract (ソース)	**Replicat (ターゲット)**
`TABLE hr.tab*;`	`MAP hr.tab*, TARGET hrBackup.*;`
`TABLE hr.index*;`	`MAP hr.index*, TARGET hrIndex.*;`

次のソース DDL 文があるとします。

```
CREATE INDEX hr.indexPayrollDate ON TABLE hr.tabPayroll (payDate);
```

ターゲットで Replicat によって実行される CREATE INDEX 文は次のとおりです。

```
CREATE INDEX hrIndex.indexPayrollDate ON TABLE hrBackup.tabPayroll (payDate);
```

ターゲットの索引がベース・オブジェクトとは異なる所有者によって所有される必要がある場合、またはターゲットとソースとで名前を変える必要がある場合は、明示的マッピングを使用します。

導出オブジェクトに対する MAP はあるが、ベース・オブジェクトにはない場合

導出オブジェクトに対する MAP 文はあるが、ベース・オブジェクトにはない場合、Replicat はどちらのオブジェクトに対しても名前変換を行いません。ターゲットの DDL 文は、ソースと同じです。導出オブジェクトをマップするには、次の方法があります。

- ベース・オブジェクトに対する明示的な MAP 文を使用します。
- 名前に問題がなければ、ワイルドカードを使用してベースと導出の両方のオブジェクトを同じ MAP 文でマップします。
- 名前の変換方法に応じて、各オブジェクトに対する MAP 文を作成します。

導出オブジェクトとしての新規表

次のものから作成される新しい表の Oracle GoldenGate による処理方法について、次に説明します。

- RENAME および ALTER RENAME
- CREATE TABLE AS SELECT

RENAME および ALTER TABLE RENAME

RENAME 操作と ALTER TABLE RENAME 操作では、ベース・オブジェクトは常に新しい表の名前です。次の例では、ベース・オブジェクト名は index_paydate とみなされます。

```
ALTER TABLE hr.indexPayrollDate RENAME TO index_paydate;
```

または

```
RENAME hr.indexPayrollDate TO index_paydate;
```

導出オブジェクト名は hr.indexPayrollDate です。

DDL レプリケーションに関連する名前変更の詳細は、172 ページの「名前変更を DDL 構成に反映するかどうかの制御」を参照してください。

CREATE TABLE AS SELECT

CREATE TABLE AS SELECT 文には、基になる任意の数のオブジェクトを対象とする SELECT 文と INSERT 文が含まれます。Oracle GoldenGate は、ターゲットで、AS SELECT 句に対応するデータをターゲット・データベースから取得します。AS SELECT 句のオブジェクトがターゲット・データベースに存在し、その名前がソースと同一である必要があります。

Oracle GoldenGate では、MAP 文で新しい表の名前 (CREATE TABLE <name>) のみ TARGET の指定にマップし、AS SELECT 句で取得した基になるオブジェクトの名前はマップしません。それらのオブジェクトに依存性があり、名前が TARGET の指定に変換されると、データに矛盾が生じる可能性があります。

次に、ソースの CREATE TABLE AS SELECT 文の例と、それが Oracle GoldenGate によってどのようにターゲットにレプリケートされるかを示します。

```
CREATE TABLE a.tab1 AS SELECT * FROM a.tab2;
```

Replicat の MAP 文は次のとおりです。

```
MAP a.tab*, TARGET a.x*;
```

Replicat によって適用されるターゲットの DDL 文は次のとおりです。

```
CREATE TABLE a.xtab1 AS SELECT * FROM a.tab2;
```

次のようにはなりません。

```
CREATE TABLE a.xtab1 AS SELECT * FROM a.xtab2;
```

AS SELECT * FROM 句の表名は、ソースと同じまま (tab2) です。

基になるオブジェクトのデータがソースとターゲットで一致しているようにするには、それらを Oracle GoldenGate のデータ・レプリケーション用に構成します。前述の例では、次の文を使用してこの要件を満たすことができます。

ソース	ターゲット

`TABLE a.tab*;`	`MAPEXCLUDE a.tab2`
	`MAP a.tab*, TARGET a.x*;`
	`MAP a.tab2, TARGET a.tab2;`

151 ページの「DDL の未修飾のオブジェクト名の適切な識別」も参照してください。

導出オブジェクトのマッピングの無効化

導出オブジェクトが含まれている MAP 文の TARGET 句に従ってその名前が変換されないようにするには、DDLOPTIONS パラメータに NOMAPDERIVED オプションを使用します。NOMAPDERIVED は、ベース・オブジェクトまたは導出オブジェクトの名前を含む明示的な MAP 文をオーバーライドします。導出オブジェクトが含まれているソース DDL は、ソースと同じ所有者とオブジェクト名でターゲットにレプリケートされます。

MAP 文がベース・オブジェクトのみ、または導出オブジェクトのみに対するものなのか、あるいはその両方に対するものかに基づいて、MAPDERIVED と NOMAPDERIVED を比較した結果を次の表に示します。

表 16　マッピング構成に基づくターゲットでの [NO]MAPDERIVED の結果

ベース・オブジェクト	導出オブジェクト	MAP/NOMAP DERIVED	導出オブジェクトが MAP によって変換されるか	導出オブジェクトにベース・オブジェクトの所有者が割り当てられるか
マップ対象[1]	マップ対象	MAPDERIVED	はい	いいえ
マップ対象	マップ対象外	MAPDERIVED	いいえ	はい
マップ対象外	マップ対象	MAPDERIVED	いいえ	いいえ
マップ対象外	マップ対象外	MAPDERIVED	いいえ	いいえ
マップ対象	マップ対象	NOMAPDERIVED	いいえ	いいえ
マップ対象	マップ対象外	NOMAPDERIVED	いいえ	いいえ
マップ対象外	マップ対象	NOMAPDERIVED	いいえ	いいえ
マップ対象外	マップ対象外	NOMAPDERIVED	いいえ	いいえ

[1]「マップ対象」は、MAP 文に含まれることを意味します。

次の例は、MAPDERIVED と NOMAPDERIVED を比較した結果を示しています。

次の表では、ベース名と導出名の両方が MAPDERIVED によって変換されるため、ターゲットではトリガーと表の両方が rpt によって所有されます。

表 17　導出オブジェクト名のデフォルト・マッピング (MAPDERIVED)

MAP 文	ソース DDL 文 (Extract によって取得)	ターゲット DDL 文 (Replicat によって適用)
`MAP fin.*, TARGET rpt.*;`	`CREATE TRIGGER fin.act_trig ON fin.acct;`	`CREATE TRIGGER rpt.act_trig ON rpt.acct;`

次の表では、NOMAPDERIVED の使用により変換が行われなくなるため、トリガーは fin によって所有されます。

表 18　NOMAPDERIVED 使用時の導出オブジェクト名のマッピング

MAP 文	ソース DDL 文 (Extract によって取得)	ターゲット DDL 文 (Replicat によって適用)
`MAP fin.*, TARGET rpt.*;`	`CREATE TRIGGER fin.act_trig ON fin.acct;`	`CREATE TRIGGER fin.act_trig ON rpt.acct;`

注意　RENAME 文では、新しい表名がベース表名とみなされ、古い表名が導出表名とみなされます。

DDL 文字列置換の使用

Oracle GoldenGate によって処理される際、DDL 操作内で文字列を置換できます。この機能は、データ構造に直接関連しないディレクトリ名やコメントなどを変更したりマップする場合に便利です。たとえば、ある表領域名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。文字列置換は、DDLSUBST パラメータによって制御されます。

DDLSUBST の使用に関するガイドライン

- DDLSUBST を使用して、ターゲットで列名やデータ型を別のものに変換しないでください。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、データがレプリケートされる際にエラーが発生します。同様に、DDLSUBST を使用して、ターゲットの DDL 文の所有者と表名を変更しないでください。レプリケートされた DDL 操作を別のターゲット・オブジェクトにマップする場合は、常に MAP 文を使用してください。

- DDLSUBST は、パラメータ・ファイル内の相対順序にかかわらず、常に DDL パラメータの後に実行されます。フィルタリングが最初に実行されるので、文字列置換に使用する基準と互換性のあるフィルタリング基準を使用します。たとえば、次のパラメータ文について考えてみます。

  ```
  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ので置換は行われません。fin が所有するオブジェクトは Oracle GoldenGate DDL 構成に含まれますが、sales が所有するオブジェクトは含まれません。

- 複数の DDLSUBST パラメータを使用できます。これらは、パラメータ・ファイルにリストされている順序で実行されます。
- Oracle DDL にコメントが含まれ、そのコメントに対して文字列置換を行う場合、DDLOPTIONS パラメータに REMOVECOMMENTS BEFORE オプションを使用しないでください。文字列置換が発生する前に REMOVECOMMENTS BEFORE によってコメントが削除されます。文字列置換を可能にしつつ、コメントを削除するには、REMOVECOMMENTS AFTER オプションを使用します。
- 置換には、データベースの制限以外、最大文字列サイズの制限はありません。文字列サイズがデータベースの制限を超える場合、操作を実行中の Extract または Replicat のプロセスは異常終了します。

注意 **DDLSUBST パラメータ文を作成する前に、この章の「処理における DDL の評価方法」を確認すると役立ちます。**

構文

```
DDLSUBST '<search_string>' WITH '<replace_string>'
[INCLUDE <inclusion clause> | EXCLUDE <exclusion clause>]
```

.

引数	説明
`'<search_string>'`	ソース DDL 文内の置換対象の文字列。文字列は一重引用符で囲みます。文字列内で一重引用符を表す場合は、二重引用符を使用します。
`WITH`	必須キーワード。
`'<replace_string>'`	ターゲット DDL 内の置換用文字列。文字列は一重引用符で囲みます。文字列内で一重引用符を表す場合は、二重引用符を使用します。
`INCLUDE <inclusion clause> | EXCLUDE <exclusion clause>`	文字列置換ルールが適用される DDL 操作をフィルタするには、INCLUDE 文と EXCLUDE 文を 1 つ以上使用します。次の表を参照してください。

表 19　DDL の包含オプションと除外オプション

オプション	説明

`INCLUDE | EXCLUDE`	INCLUDE および EXCLUDE を使用して、包含句または除外句の開始を示します。
	◆ 包含句には、このパラメータの対象となる DDL を識別するフィルタリング基準が含まれます。
	◆ 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が含まれます。
	包含句または除外句は、INCLUDE または EXCLUDE キーワードの後に、適用されるパラメータの有効なオプションの組合せを指定して構成する必要があります。
	
	EXCLUDE を使用する場合、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.*"`
	ただし、次のいずれかは使用できます。
	`DDL INCLUDE ALL, EXCLUDE OBJNAME "hr.*"`
	`DDL INCLUDE OBJNAME "fin.*" EXCLUDE "fin.ss"`
	EXCLUDE は、同じ基準が含まれている INCLUDE よりも優先されます。複数の包含句と除外句を使用できます。

`MAPPED | UNMAPPED | OTHER | ALL`	DDL 操作のスコープに基づいて INCLUDE または EXCLUDE を適用するには、MAPPED、UNMAPPED、OTHER および ALL を使用します。 ◆ MAPPED では、INCLUDE または EXCLUDE が MAPPED スコープの DDL 操作に適用されます。MAPPED フィルタリングは、他の DDL パラメータ・オプションを使用して指定されたフィルタリングの前に実行されます。 ◆ UNMAPPED では、INCLUDE または EXCLUDE が UNMAPPED スコープの DDL 操作に適用されます。 ◆ OTHER では、INCLUDE または EXCLUDE が OTHER スコープの DDL 操作に適用されます。 ◆ ALL では、INCLUDE または EXCLUDE がすべてのスコープの DDL 操作に適用されます。
`OPTYPE <type>`	INCLUDE または EXCLUDE を特定のタイプの DDL 操作 (CREATE、ALTER、RENAME など) に適用するには、OPTYPE を使用します。<type> には、データベースに有効な任意の DDL コマンドを使用します。たとえば、ALTER 操作を含める場合の正しい構文は次のようになります。 `DDL INCLUDE OPTYPE ALTER`

表 19　DDL の包含オプションと除外オプション (続き)

オプション	説明

`OBJTYPE '<type>'`	INCLUDE または EXCLUDE を特定のタイプのデータベース・オブジェクトに適用するには、OBJTYPE を使用します。<type> には、データベースに有効な任意のオブジェクト・タイプ (TABLE、INDEX、TRIGGER など) を使用します。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・ログの場合、正しいタイプはそれぞれ snapshot と 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 には OBJNAME オプションを使用しないでください (OBJNAME には owner.object が必要ですが、USER はスキーマのみを持つためです)。

`OBJNAME "<name>"`	INCLUDE または EXCLUDE をオブジェクトの完全修飾名 (owner.table_name など) に適用するには、OBJNAME を使用します。このオプションでは、二重引用符で囲まれた文字列を入力として使用します。
	ワイルドカードは、オブジェクト名にのみ使用できます。
	例 :
	`DDL INCLUDE OBJNAME "accounts.*"`
	Oracle USER オブジェクトには OBJNAME を使用しないでください (OBJNAME には owner.object が必要ですが、USER はスキーマのみを持つためです)。
	Replicat パラメータ・ファイルで OBJNAME と MAPPED を組み合せて使用する場合、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.*" です。
	`MAP fin.exp_*, TARGET fin2.*;`
	次の例では、CREATE TABLE 文はソースで次のように実行されます。
	`CREATE TABLE fin.exp_phone;`
	ターゲットでは次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たデータベース・ユーザーにマップします。

表 19　DDL の包含オプションと除外オプション (続き)

オプション	説明

	トリガー、シノニムおよび索引を作成する DDL の場合、OBJNAME の値はトリガー、シノニムまたは索引の名前ではなく、ベース・オブジェクトの名前にする必要があります。
	たとえば、次の DDL 文を含める場合、正しい値は hr.insert_trig ではなく hr.accounts です。
	`CREATE TRIGGER hr.insert_trig ON hr.accounts;`
	RENAME 操作では、OBJNAME の値を新しい表名にする必要があります。たとえば、次の DDL 文を含める場合、正しい値は hr.acct です。
	`ALTER TABLE hr.accounts RENAME TO acct;`

`INSTR '<string>'`	INCLUDE または EXCLUDE を、コマンド構文内に特定の文字列を含む (ただしコメント内には含まない) DDL 文に適用するには、INSTR を使用します。たとえば、次の例では、索引を作成する DDL は除外されます。 `DDL INCLUDE ALL EXCLUDE INSTR 'CREATE INDEX'` 文字列は一重引用符で囲みます。文字列の検索では、大 / 小文字は区別されません。 INSTR では、文字列内に含まれる一重引用符 (' ') も、NULL 値もサポートされません。

`INSTRCOMMENTS '<comment_string>'`	INCLUDE または EXCLUDE を、コメント内に特定の文字列を含む (DDL コマンド自体には含まない) DDL 文に適用するには、INSTRCOMMENTS を使用します。INSTRCOMMENTS を使用すると、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、次の例では、コメントに source を含む DDL 文は除外されます。
	`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。文字列の検索では、大 / 小文字は区別されません。INSTR と INSTRCOMMENTS を組み合せることで、同じ DDL 文のコマンド構文内とコメント内の文字列をフィルタできます。
	INSTRCOMMENTS では、文字列内に含まれる一重引用符 (' ') も、NULL 値もサポートされません。

表 19　DDL の包含オプションと除外オプション (続き)

オプション	説明

INSTRWORDS '<word list>'	INCLUDE または EXCLUDE を特定の語を含む DDL 文に適用するには、INSTRWORDS を使用します。
	<word list> には、一重引用符内に任意の順序で語を指定します。空白を含めるには、空白を (語がある場合は語も) 二重引用符で囲みます。二重引用符は、文を囲む場合にも使用できます。
	INSTRWORDS が有効になるには、指定された語がすべて DDL に存在する必要があります。
	例 :
	`ALTER TABLE INCLUDE INSTRWORDS 'ALTER CONSTRAINT " xyz"`
	この例は、次に一致します。
	`ALTER TABLE ADD CONSTRAINT xyz CHECK`
	および
	`ALTER TABLE DROP CONSTRAINT xyz`
	INSTRWORDS では、文字列内に含まれる一重引用符 (' ') も、NULL 値もサポートされません。

INSTRCOMMENTSWORDS '<word list>'	INSTRWORDS と同様に機能しますが、DDL 文内のコメントにのみ適用され、DDL 構文自体には適用されません。INSTRCOMMENTS を使用すると、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、文字列内に含まれる一重引用符 (' ') も、NULL 値もサポートされません。 INSTRWORDS と INSTRCOMMENTSWORDS を組み合せることで、同じ DDL 文のコマンド構文内とコメント内の文字列をフィルタできます。

例　次の例では、DDL コマンドに logfile という語が含まれる場合にのみ、新規ディレクトリが置換されます。検索文字列が複数回検出された場合、置換文字列が複数回挿入されます。

```
DDLSUBST '/file1/location1' WITH '/file2/location2'
INCLUDE INSTR'logfile'
```

例　次の例では、fin が所有する表の文字列 cust が文字列 customers に置換されます。

```
DDLSUBST 'cust' WITH 'customers'
INCLUDE ALL OBJTYPE 'table' OBJNAME "fin.*"
```

検索で大 / 小文字は区別されません。文字列内で一重引用符を表す場合は、二重引用符を使用します。

例　この例では複数の DDLSUBST パラメータを使用します。これらは、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。最終的には、文字列 a および b が c に置換されます。

```
DDLSUBST 'a' WITH 'b' INCLUDE ALL
DDLSUBST 'b' WITH 'c' INCLUDE ALL
```

Replicat によって実行される DDL の伝播の制御

Extract と Replicat は、いずれも DDL 操作を発行します。

- Extract は ALTER TABLE 文を発行してログ・グループを作成します。
- Replicat はレプリケートされた DDL 文をターゲットに適用します。

Oracle GoldenGate の DDL 操作を識別するため、次のコメントが Extract と Replicat の各 DDL 文に追加されます。

```
/* GOLDENGATE_DDL_REPLICATION */
```

DDLOPTIONS パラメータで、Replicat の DDL を伝播するかどうかを制御します。

- GETREPLICATES オプションと IGNOREREPLICATES オプションでは、*Replicat* の DDL 操作を Extract で取得するか、無視するかを制御します。デフォルトは IGNOREREPLICATES です。
- GETAPPLOPS オプションと IGNOREAPPLOPS オプションでは、*Replicat 以外のアプリケーション* (ビジネス・アプリケーション) からの DDL を取得するか、無視するかを制御します。

デフォルトでは、ローカル Replicat によってローカル・データベースに適用される DDL を Extract は無視するため、DDL はそのソースに戻されません (ただし、Extract はレプリケーション用に構成されている他のすべての DDL を取得します)。次に、デフォルトの DDLOPTIONS 構成を示します。

```
DDLOPTIONS GETAPPLOPS, IGNOREREPLICATES
```

この動作は変更可能です。次の項を参照してください。

- 「アクティブ / アクティブ (双方向) 構成での DDL の伝播」
- 「カスケード構成での DDL の伝播」

アクティブ / アクティブ (双方向) 構成での DDL の伝播

Oracle GoldenGate では、2 つのシステム間のアクティブ / アクティブ型 DDL レプリケーションがサポートされます。アクティブ / アクティブ型双方向レプリケーションでは、Oracle GoldenGate プロセスで次のように構成する必要があります。

1. 一方のシステムでビジネス・アプリケーションによって実行される DDL は、他方のシステムにレプリケートして同期を保つ必要があります。この要件を満たすには、両方のシステムの Extract パラメータ・ファイルで DDLOPTIONS 文に GETAPPLOPS オプションを含めます。
2. 一方のシステムで Replicat によって適用される DDL は、ローカルの Extract で取得して他方のシステムに戻す必要があります。この要件を満たすには、両方のシステムの Extract パラメータ・ファイルで DDLOPTIONS 文に GETREPLICATES オプションを使用します。

   注意　ループバックを防ぐため、Oracle GoldenGate の内部トークンによって実際の Replicat DDL 文自体は無視されます。Replicat の DDL を元のシステムに戻す目的は、着信 DML の受信に備えて、そのシステムの Replicat がオブジェクト・メタデータ・キャッシュを更新し、新しいメタデータを持つようにすることです。この項の図を参照してください。

3. 取得された Replicat DDL 文がリモートの Extract から届いたらオブジェクト・メタデータ・キャッシュを更新するよう、各 Replicat を構成する必要があります。この要件を満たすには、両方のシステムの Replicat パラメータ・ファイルで DDLOPTIONS 文に UPDATEMETADATA オプションを使用します。

その結果、DDLOPTIONS 文は次のようになります。

Extract (プライマリおよびセカンダリ)

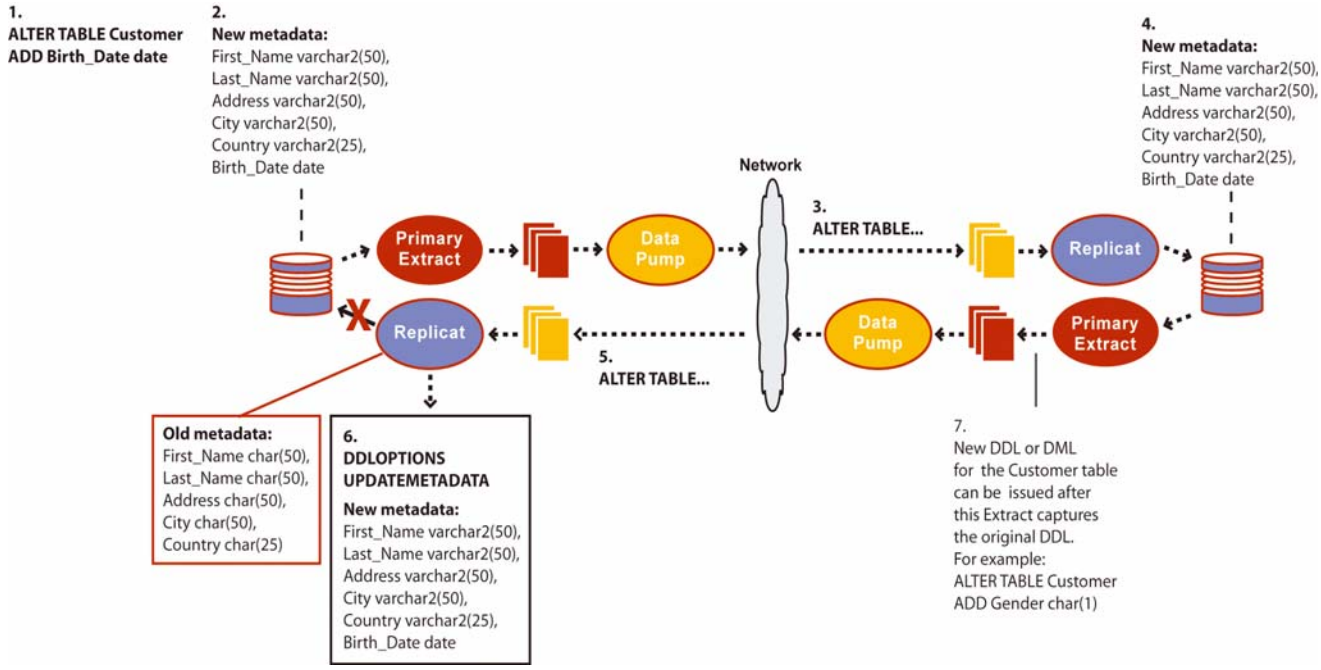
```
DDLOPTIONS GETREREPLICATES, GETAPPLOPS
```

Replicat (プライマリおよびセカンダリ)

```
DDLOPTIONS UPDATEMETADATA
```

警告 元の DDL がリモート・システムにレプリケートされてからそのシステムの Extract によって再度取得されるまでの時間を考慮して、元の DDL と同じオブジェクトに対して新しい DDL または DML を発行するようにします。これで、各操作が元のシステムの Replicat に正しい順序で届くことが保証され、メタデータの矛盾による DML エラーの発生を防げます。詳細は図を参照してください。

図 18 Replicat のオブジェクト・メタデータ・キャッシュが更新される DDL のラウンドトリップ・パス



カスケード構成での DDL の伝播

カスケード構成では、各中間システムの Extract パラメータ・ファイルで DDLOPTIONS に次の設定を使用します。この構成により、Extract は中間システムの Replicat から DDL を取得して、次のシステム・ダウンストリームにカスケードします。

```
DDLOPTIONS GETREPLICATES, IGNOREAPPLOPS
```

サプリメンタル・ログ・グループの自動追加

DDLOPTIONS パラメータに ADDTRANDATA オプションを使用すると、次のことを行えます。

- CREATE TABLE によって作成された新規表で Oracle のサプリメンタル・ロギングを自動的に有効化します。

- 列を追加または削除する ALTER TABLE の対象となる表で Oracle のサプリメンタル・ロギングを更新します。
- 名前が変更される表で Oracle のサプリメンタル・ロギングを更新します。
- 一意キーまたは主キーが追加または削除される表で Oracle のサプリメンタル・ロギングを更新します。

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 パラメータが使用されていなければターゲットにレプリケートされません。

このオプションの詳細は、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』を参照してください。

レプリケートされた DDL からのコメントの削除

DDLOPTIONS パラメータに REMOVECOMMENTS BEFORE オプションと REMOVECOMMENTS AFTER オプションを使用すると、ソース DDL で使用されていたコメントをターゲット DDL に含めないようにすることができます。文字列置換で使用できるように、コメントはデフォルトでは削除されません。

このオプションの詳細は、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』を参照してください。

名前変更を DDL 構成に反映するかどうかの制御

DDLOPTIONS パラメータに NOCROSSRENAME オプションを使用すると、Oracle GoldenGate 構成から除外されているオブジェクトの名前が構成内の既存の名前に変更されないようにするルールを適用できます。次に、名前変更が発生する場合の例を示します。

- TableA は除外され、tableB は含まれています。
- TableA の名前が tableB に変更されます。

オブジェクトの名前が Oracle GoldenGate 構成内の既存の名前に変更されると、Extract によって警告が発行されるため、ユーザーは適切な処理を実行できます (それを Oracle GoldenGate 構成内に保持するか、またはパラメータを適切に調整して除外できます)。たとえば、名前を変更したオブジェクトの構造が Oracle GoldenGate でサポートされていない場合にエラーを回避できるなど、この通知は役に立ちます。

Oracle RAC 環境では、他にもパフォーマンスの向上というメリットが NOCROSSRENAME にあります。除外オブジェクトの名前が構成内の既存の名前に変更された場合に通常であればそれらを追跡するためにノード間で必要とされる処理上のオーバーヘッドが排除されます。

NOCROSSRENAME は、次のものに対してグローバルに適用されます。

- パラメータ・ファイルの TABLE 文と TABLEEXCLUDE 文で指定されているすべてのオブジェクト
- Oracle GoldenGate DDL 構成から除外され、TABLE または TABLEEXCLUDE で指定されていないすべてのオブジェクト

DDLOPTIONS NOCROSSRENAME は、TABLEEXCLUDE パラメータに NORENAME オプションを使用した場合と同じ結果をもたらします。2 つのパラメータの相違点は、TABLEEXCLUDE NORENAME では TABLEEXCLUDE 文のオブジェクトのみが機能の対象となるため、NOCROSSRENAME より選択的な処理が可能なことです。

IDENTIFIED BY パスワードのレプリケート

DDLOPTIONS パラメータに次のオプションを使用すると、レプリケートされた {CREATE | ALTER} USER <name> IDENTIFIED BY <password> 文のパスワードの処理方法を制御できます。これらのオプションは一緒に使用する必要があります。

DEFAULTUSERPASSWORD

このオプションは Replicat で有効です。ソース文で使用されているものとは異なるパスワードを指定するには、レプリケートされた {CREATE | ALTER} USER <name> IDENTIFIED BY <password> 文に DEFAULTUSERPASSWORD を使用します。デフォルトでは、ソース・パスワードがターゲットにレプリケートされます。クリアテキスト・パスワードまたは暗号化パスワードを指定できます。Replicat は、Extract が証跡に書き込むプレースホルダを、指定されたパスワードで置き換えます。

REPLICATEPASSWORD | NOREPLICATEPASSWORD

このオプションは Extract で有効です。デフォルト (REPLICATEPASSWORD) では、Oracle GoldenGate はターゲットの CREATE 文または ALTER 文の {CREATE | ALTER} USER <name> IDENTIFIED BY <password> のソース・パスワードを使用します。ソース・パスワードをターゲットに送信しない場合、NOREPLICATEPASSWORD を使用します。

これらのオプションの詳細は、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』を参照してください。

処理における DDL の評価方法

次に、ソース・システムとターゲット・システムでの Oracle GoldenGate による DDL 文の処理方法について説明します。Oracle GoldenGate パラメータの各種基準が処理される順序を示し、Extract と Replicat がそれぞれ DDL を処理する方法の違いを説明します。

Extract

1. Extract が DDL 操作を取得します。
2. Extract は、コメントがあれば、それをメインの文から分離します。
3. Extract は、DDL パラメータを検索します。(この例では、存在するものとします。)
4. Extract は、IGNOREREPLICATES パラメータを検索します。これが存在し、Replicat がこのシステムでこの DDL 操作を生成した場合、Extract はその DDL 文を無視します。(この例では、このシステムで Replicat 操作が実行されないものとします。)
5. Extract は、文が RENAME かどうかを判断します。そうである場合、内部的に名前変更フラグが付けられます。
6. Extract は、ベース・オブジェクト名と導出オブジェクト名 (存在する場合) を取得します。
7. 文が RENAME の場合、Extract はこれを ALTER TABLE RENAME に変更します。

8. Extract は、DDLOPTIONS REMOVECOMMENTS BEFORE パラメータを検索します。存在する場合、Extract は DDL 文からコメントを削除しますが、INSTR または INSTRCOMMENTS を使用する DDL INCLUDE 句または DDL EXCLUDE 句がある場合に備えて、それらを保存します。
9. Extract は、DDL スコープ (MAPPED、UNMAPPED または OTHER) を判別します。次の場合は MAPPED です。
   - 操作とオブジェクト・タイプがマッピングでサポートされる場合。

     および
   - ベース・オブジェクト名または導出オブジェクト名、あるいはその両方 (RENAME の場合) が TABLE パラメータ内にある場合。

   次の場合は UNMAPPED です。
   - 操作とオブジェクト・タイプがマッピングでサポートされない場合。

     および
   - ベース・オブジェクト名または導出オブジェクト名、あるいはその両方 (RENAME の場合) が TABLE パラメータ内にない場合。

   これ以外の場合、操作は OTHER とみなされます。
10. Extract は、DDL パラメータに INCLUDE 句と EXCLUDE 句があるかどうかをチェックし、これらの句の DDL パラメータ基準を評価します。INCLUDE または EXCLUDE が TRUE と評価されるには、すべてのオプションが TRUE と評価される必要があります。次のようになります。
    - EXCLUDE 句が TRUE と評価される場合、Extract は DDL 操作を破棄して別の DDL 操作を評価します。この場合、処理手順は最初に戻ります。
    - INCLUDE 句が TRUE と評価される場合、または DDL パラメータに INCLUDE 句も EXCLUDE 句も含まれていない場合、Extract は DDL 操作を受け入れ、処理ロジックが続けられます。
11. Extract は、DDLSUBST パラメータを検索し、INCLUDE 句と EXCLUDE 句を評価します。これらの句の基準が最終的に TRUE になる場合、Extract は文字列置換を実行します。Extract は、パラメータ・ファイルの各 DDLSUBST 文に対して DDL 操作を評価します。TRUE と評価されたすべての DDLSUBST 文について、Extract は DDLSUBST パラメータがファイルにリストされている順序で文字列置換を実行します。
12. DDLSUBT の処理が終わると、Extract は REMOVECOMMENTS AFTER パラメータを検索します。存在する場合、Extract は DDL 文からコメントを削除します。
13. Extract は、DDLOPTIONS ADDTRANDATA を検索します。存在する場合、操作が CREATE TABLE であれば、Extract は ALTER TABLE <name> ADD SUPPLEMENTAL LOG GROUP コマンドを表に対して発行します。
14. Extract は、DDL 文を証跡に書き込みます。

Replicat

1. Replicat が証跡から DDL 操作を読み取ります。
2. Replicat は、コメントがあれば、それをメインの文から分離します。
3. Replicat は、DDLOPTIONS REMOVECOMMENTS BEFORE を検索します。存在する場合、Replicat は DDL 文からコメントを削除します。
4. Replicat は、DDL 同期スコープを評価し、DDL が名前マッピングに適しているかどうかを判断します。他はすべて、OTHER スコープです。

5. Replicat は、パラメータ・ファイルの MAP 文を評価します。(証跡から読み取られた) この DDL のソースのベース・オブジェクト名が MAP 文のいずれかに存在する場合、操作は MAPPED スコープとしてマークされます。それ以外の場合、UNMAPPED スコープとしてマークされます。
6. Replicat は、ソースのベース・オブジェクト名を、MAP 文の TARGET 句に指定されたベース・オブジェクト名で置き換えます。
7. 導出オブジェクトがある場合、Replicat は DDLOPTIONS MAPDERIVED を検索します。存在する場合、Replicat はソースの導出名を MAP 文のターゲットの導出名で置き換えます。
8. Replicat は、DDL パラメータに INCLUDE 句と EXCLUDE 句があるかどうかをチェックし、それらに含まれる DDL パラメータ基準を評価します。INCLUDE または EXCLUDE が TRUE と評価されるには、すべてのオプションが TRUE と評価される必要があります。次のようになります。
   - EXCLUDE 句が TRUE と評価される場合、Replicat は DDL 操作を破棄して別の DDL 操作の評価を開始します。この場合、処理手順は最初に戻ります。
   - INCLUDE 句が TRUE と評価される場合、または DDL パラメータに INCLUDE 句も EXCLUDE 句も含まれていない場合、Replicat は DDL 操作を受け入れ、処理ロジックが続けられます。
9. Replicat は、DDLSUBST パラメータを検索し、INCLUDE 句と EXCLUDE 句を評価します。これらの句のオプションが最終的に TRUE になる場合、Replicat は文字列置換を実行します。Replicat は、パラメータ・ファイルの各 DDLSUBST 文に対して DDL 操作を評価します。TRUE と評価されたすべての DDLSUBST 文について、Replicat は DDLSUBST パラメータがファイルにリストされている順序で文字列置換を実行します。
10. DDLSUBT の処理が終わると、Replicat は REMOVECOMMENTS AFTER パラメータを検索します。存在する場合、Replicat は DDL 文からコメントを削除します。
11. Replicat は、ターゲット・データベースで DDL 操作を実行します。
12. エラーがなければ、Replicat は次の DDL 文を処理します。エラーがある場合、Replicat は次の手順を実行します。
13. Replicat は、Replicat の DDLERROR パラメータ文の INCLUDE ルールと EXCLUDE ルールを、パラメータ・ファイルに出現する順序で分析します。Replicat は、エラー・コードに対応するルールを検出すると、指定されたエラー処理を適用します。それ以外の場合は、DEFAULT 処理を適用します。
14. エラー処理を行っても DDL 操作を続行できない場合、Replicat はルールでの指定に応じて異常終了、操作の無視または破棄のいずれかを実行します。

注意 **同じソースに対して複数のターゲットが MAP 文にある場合、ターゲットごとに処理ロジックが実行されます。**

Extract の DDL 処理エラーへの対処

メタデータが見つからないオブジェクトに関するエラー (Extract で検出) を処理するには、DDLERROR パラメータの Extract オプションを使用します。

構文

```
DDLERROR [RESTARTSKIP <num skips>] [SKIPTRIGGERERROR <num errors>]
```

条件 :

- RESTARTSKIP は、起動時に複数の DDL 操作をスキップし、Extract がエラーによって異常終了するのを防ぎます。デフォルトでは操作をスキップしないので、Extract はエラーによって異常終了します。最大 100,000 の DDL 操作をスキップできます。

スキップされた操作に関する情報を Extract レポート・ファイルに書き込むには、DDLOPTIONS パラメータに REPORT オプションを使用します。

Replicat の DDL 処理エラーへの対処

DDL がターゲット・データベースに適用される際に発生するエラーを処理するには、DDLERROR パラメータの Replicat オプションを使用します。DDLERROR オプションを使用すると、ほとんどのエラーはデフォルトの方法 (処理の停止など) で処理でき、他のエラーも特定の方法で処理できます。同じパラメータ・ファイル内で DDLERROR の複数のインスタンスを使用し、予期されるすべてのエラーを処理できます。

<error>、DEFAULT および <response> の組合せを使用して、予期される DDL エラーと予期しない DDL エラーに対する Replicat のレスポンス方法に関するルールを作成します。必ず適切な包含句と除外句を指定して、目的の DDL にルールを適用してください。その後で、必要に応じて追加オプションを使用し、エラー処理を調整します。

構文

```
DDLERROR
{<error> | DEFAULT} {<response>}
{INCLUDE <inclusion clause> | EXCLUDE <exclusion clause>}
[IGNOREMISSINGOBJECTS | ABENDONMISSINGOBJECTS]
```

引数	説明
{<error> | DEFAULT} {<response>}	◆ <error> は、この文で処理する特定の DDL エラーです。 ◆ DEFAULT では、すべての DDL エラーを対象としたグローバルなレスポンスを設定します (ただし、DDLERROR 文が明示的に指定されているものを除く)。 ◆ <response> は、次のいずれかです。 ABEND 操作をロールバックして、処理を異常終了させます。ABEND がデフォルトです。 DISCARD 原因の操作を廃棄ファイルに記録し、後続の DDL の処理は続行します。DISCARDFILE パラメータで廃棄ファイルを指定します。 IGNORE エラーを無視します。

引数	説明
	`RETRYOP MAXRETRIES <n> [RETRYDELAY <delay>]` 原因の操作を再試行します。再試行の回数を制御するには、MAXRETRIES オプションを使用します。MAXRETRIES の指定回数を超えると、Replicat は異常終了します。整数で指定します。 再試行間隔 (秒単位) を設定するには、RETRYDELAY を使用します。
`{INCLUDE <inclusion clause> | EXCLUDE <exclusion clause>}`	DDLERROR 文で特定の DDL を処理するか、または処理しないかを制御します。詳細は、次の表を参照してください。
`[IGNOREMISSINGOBJECTS | ABENDONMISSINGOBJECTS]`	ターゲット上で検出できなかったオブジェクトに対して DML が発行された場合に、Extract を異常終了させるかどうかを制御します。通常、このような状況が生じるのは、レプリケーション以外でターゲットに対して DDL が直接発行された場合、またはソース定義とターゲット定義との間に矛盾がある場合です。 IGNOREMISSINGOBJECTS を使用すると、Replicat は存在しない表に対する DML 操作をスキップします。 ABENDONMISSINGOBJECTS を使用すると、存在しない表に対する DML 操作によって、Replicat は異常終了します。

表 20　DDL の包含オプションと除外オプション

オプション	説明

`INCLUDE | EXCLUDE`	INCLUDE および EXCLUDE を使用して、包含句または除外句の開始を示します。
	◆ 包含句には、このパラメータの対象となる DDL を識別するフィルタリング基準が含まれます。
	◆ 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が含まれます。
	包含句または除外句は、INCLUDE または EXCLUDE キーワードの後に、適用されるパラメータの有効なオプションの組合せを指定して構成する必要があります。
	
	EXCLUDE を使用する場合、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.*"`
	ただし、次のいずれかは使用できます。
	`DDL INCLUDE ALL, EXCLUDE OBJNAME "hr.*"`
	`DDL INCLUDE OBJNAME "fin.*" EXCLUDE "fin.ss"`
	EXCLUDE は、同じ基準が含まれている INCLUDE よりも優先されます。複数の包含句と除外句を使用できます。

表 20　DDL の包含オプションと除外オプション (続き)

オプション	説明
MAPPED | UNMAPPED | OTHER | ALL	DDL 操作のスコープに基づいて INCLUDE または EXCLUDE を適用するには、MAPPED、UNMAPPED、OTHER および ALL を使用します。 ◆ MAPPED では、INCLUDE または EXCLUDE が MAPPED スコープの DDL 操作に適用されます。MAPPED フィルタリングは、他の DDL パラメータ・オプションを使用して指定されたフィルタリングの前に実行されます。 ◆ UNMAPPED では、INCLUDE または EXCLUDE が UNMAPPED スコープの DDL 操作に適用されます。 ◆ OTHER では、INCLUDE または EXCLUDE が OTHER スコープの DDL 操作に適用されます。 ◆ ALL では、INCLUDE または EXCLUDE がすべてのスコープの DDL 操作に適用されます。
OPTYPE <type>	INCLUDE または EXCLUDE を特定のタイプの DDL 操作 (CREATE、ALTER、RENAME など) に適用するには、OPTYPE を使用します。<type> には、データベースに有効な任意の DDL コマンドを使用します。たとえば、ALTER 操作を含める場合の正しい構文は次のようになります。 `DDL INCLUDE OPTYPE ALTER`

OBJTYPE '<type>'	INCLUDE または EXCLUDE を特定のタイプのデータベース・オブジェクトに適用するには、OBJTYPE を使用します。<type> には、データベースに有効な任意のオブジェクト・タイプ (TABLE、INDEX、TRIGGER など) を使用します。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・ログの場合、正しいタイプはそれぞれ snapshot と 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 には OBJNAME オプションを使用しないでください (OBJNAME には owner.object が必要ですが、USER はスキーマのみを持つためです)。

OBJNAME "<name>"	INCLUDE または EXCLUDE をオブジェクトの完全修飾名 (owner.table_name など) に適用するには、OBJNAME を使用します。このオプションでは、二重引用符で囲まれた文字列を入力として使用します。
	ワイルドカードは、オブジェクト名にのみ使用できます。
	例 :
	`DDL INCLUDE OBJNAME "accounts.*"`
	Oracle USER オブジェクトに OBJNAME を使用しないでください (OBJNAME には owner.object が必要ですが、USER はスキーマのみを持つためです)。
	Replicat パラメータ・ファイルで OBJNAME と MAPPED を組み合せて使用する場合、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.*" です。
	`MAP fin.exp_*, TARGET fin2.*;`

表 20　DDL の包含オプションと除外オプション (続き)

オプション	説明

	次の例では、CREATE TABLE 文はソースで次のように実行されます。
	`CREATE TABLE fin.exp_phone;`
	ターゲットでは次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たデータベース・ユーザーにマップします。
	
	トリガー、シノニムおよび索引を作成する DDL の場合、OBJNAME の値はトリガー、シノニムまたは索引の名前ではなく、ベース・オブジェクトの名前にする必要があります。
	たとえば、次の DDL 文を含める場合、正しい値は hr.insert_trig ではなく hr.accounts です。
	`CREATE TRIGGER hr.insert_trig ON hr.accounts;`
	RENAME 操作では、OBJNAME の値を新しい表名にする必要があります。たとえば、次の DDL 文を含める場合、正しい値は hr.acct です。
	`ALTER TABLE hr.accounts RENAME TO acct;`

`INSTR '<string>'`	INCLUDE または EXCLUDE を、コマンド構文内に特定の文字列を含む (ただしコメント内には含まない) DDL 文に適用するには、INSTR を使用します。たとえば、次の例では、索引を作成する DDL は除外されます。 `DDL INCLUDE ALL EXCLUDE INSTR 'CREATE INDEX'` 文字列は一重引用符で囲みます。文字列の検索では、大 / 小文字は区別されません。 INSTR では、文字列内に含まれる一重引用符 (' ') も、NULL 値もサポートされません。

表 20　DDL の包含オプションと除外オプション (続き)

オプション	説明

INSTRCOMMENTS '<comment_string>'	INCLUDE または EXCLUDE を、コメント内に特定の文字列を含む (DDL コマンド自体には含まない) DDL 文に適用するには、INSTRCOMMENTS を使用します。INSTRCOMMENTS を使用すると、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、次の例では、コメントに source を含む DDL 文は除外されます。
	`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。文字列の検索では、大 / 小文字は区別されません。INSTR と INSTRCOMMENTS を組み合せることで、同じ DDL 文のコマンド構文内とコメント内の文字列をフィルタできます。
	INSTRCOMMENTS では、文字列内に含まれる一重引用符 (' ') も、NULL 値もサポートされません。

INSTRWORDS '<word list>'	INCLUDE または EXCLUDE を特定の語を含む DDL 文に適用するには、INSTRWORDS を使用します。
	<word list> には、一重引用符内に任意の順序で語を指定します。空白を含めるには、空白を (語がある場合は語も) 二重引用符で囲みます。二重引用符は、文を囲む場合にも使用できます。
	INSTRWORDS が有効になるには、指定された語がすべて DDL に存在する必要があります。
	例 :
	`ALTER TABLE INCLUDE INSTRWORDS 'ALTER CONSTRAINT " xyz"`
	この例は、次に一致します。
	`ALTER TABLE ADD CONSTRAINT xyz CHECK`
	および
	`ALTER TABLE DROP CONSTRAINT xyz`
	INSTRWORDS では、文字列内に含まれる一重引用符 (' ') も、NULL 値もサポートされません。

INSTRCOMMENTSWORDS '<word list>'	INSTRWORDS と同様に機能しますが、DDL 文内のコメントにのみ適用され、DDL 構文自体には適用されません。INSTRCOMMENTS を使用すると、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、文字列内に含まれる一重引用符 (' ') も、NULL 値もサポートされません。 INSTRWORDS と INSTRCOMMENTSWORDS を組み合せることで、同じ DDL 文のコマンド構文内とコメント内の文字列をフィルタできます。

サンプルの DDLERROR 文

次の例の DDLERROR 文では、Replicat は 10 秒間隔で操作を 3 回試行してから、指定されたエラーを無視します。Replicat は、ワイルドカード "tab*" に一致する名前のオブジェクト (任意のユーザー、操作) に対して実行された DDL 操作にエラー処理を適用しますが、"tab1*" に一致するものは除外されます。

```
DDLERROR <error> IGNORE RETRYOP MAXRETRIES 3 RETRYDELAY 10 &
INCLUDE ALL OBJTYPE TABLE OBJNAME "tab*" EXCLUDE OBJNAME "tab1*"
```

このエラー以外のすべてのエラーを処理するには、次の DDLERROR 文を追加します。

```
DDLERROR DEFAULT ABENDS
```

この場合、DDL エラーにより Replicat は異常終了します。

複数の DDLERROR 文の使用

パラメータ・ファイル内で DDLERROR 文がリストされている順序は、それぞれ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じエラーが指定されている場合は例外です。このような場合、Replicat では最初にリストされているもののみが使用されます。たとえば、次の文では、エラーにより Replicat は異常終了します。

```
DDLERROR <error1> ABEND
DDLERROR <error1> IGNORE
```

ただし、適切な修飾子が指定されている場合は、前述の構成の方が便利です。次に例を示します。

```
DDLERROR <error1> ABEND INCLUDE OBJNAME "tab*"
DDLERROR <error1> IGNORE
```

この場合、INCLUDE 文があるので、エラーが発生した DDL 文内のオブジェクト名がワイルドカード "tab*" と一致する場合にのみ、Replicat は異常終了します。オブジェクトが他の名前の場合、Replicat はエラーの発生した操作を無視します。

DDL トリガー・エラーへの対処

ソース DDL の失敗または成功に関連する Oracle GoldenGate DDL トリガー・エラーを処理するには、params.sql 実行不可能スクリプトで次のパラメータを使用します。

- **ddl_fire_error_in_trigger**: TRUE に設定されている場合、Oracle GoldenGate DDL トリガー・エラーは、Oracle GoldenGate エラー・メッセージとデータベース・エラー・メッセージとともにソースのエンド・ユーザー・アプリケーションに表示されます。ソースの操作は失敗します。

  FALSE に設定されている場合、エラーは表示されずに、メッセージが Oracle GoldenGate ディレクトリのトリガー・トレース・ファイルに書き込まれます。ソースの操作は成功しますが、DDL はレプリケートされません。後続のデータ変更が古いターゲット・オブジェクトの構造に適合しない場合、ターゲット・アプリケーションは最終的には失敗します。デフォルトは FALSE です。

- **_ddl_cause_error**: TRUE に設定されている場合、故意にエラーを発生させ、トリガーのエラー・レスポンスをテストします。エラーを生成するため、Oracle GoldenGate では例外処理なしで 0 (ゼロ) 行を SELECT しようとします。テストが終了したら、このフラグをデフォルトの FALSE に戻します。

params.sql スクリプトは、Oracle GoldenGate のルート・ディレクトリにあります。

DDL レポート情報の表示

Oracle GoldenGate では、Extract および Replicat の各レポートの最後に、DDL 操作に関する基本的な統計がデフォルトで表示されます。拡張 DDL レポートを有効化するには、DDLOPTIONS パラメータに REPORT オプションを使用します。拡張レポートには、DDL 処理に関する次の情報が含まれます。

- Oracle GoldenGate によって処理された DDL 操作の手順ごとの履歴
- 使用中の DDL フィルタリング・パラメータと処理パラメータ

拡張 DDL レポート情報によってレポート・ファイルのサイズは大きくなりますが、この情報は特定の状況で役立ちます (トラブルシューティングや、サプリメンタル・ロギングを追加する ADDTRANDATA がいつ適用されたかを確認する場合など)。

プロセス・レポートを表示する手順

レポートを表示するには、GGSCI で VIEW REPORT コマンドを使用します。

```
VIEW REPORT <group>
```

Extract DDL レポート

Extract レポートにリストされる内容は次のとおりです。

- 取得された各 DDL 操作の構文全体、開始と終了の SCN、Oracle インスタンス、DDL 順序番号 (履歴表の SEQNO 列から取得) および操作のサイズ (バイト単位)。
- 処理基準がどのように操作に適用されたかを示す後続のエントリ (文字列置換または INCLUDE と EXCLUDE のフィルタリングなど)。
- 操作が証跡に書き込まれたか、除外されたかを示す別のエントリ。

次の例 (Extract レポートから取得) は、包含された操作と除外された操作を示しています。包含された操作にはレポート・メッセージがあり、除外された操作にはありません。

```
2011-01-20 15:11:41  GGS INFO     2100  DDL found, operation [create table myTable (
    myId number (10) not null,
    myNumber number,
    myString varchar2(100),
    myDate date,
    primary key (myId)
) ], start SCN [1186754], commit SCN [1186772] instance [test10g (1)], DDL seqno [4134].

2011-01-20 15:11:41  GGS INFO    2100  DDL operation included [INCLUDE OBJNAME myTable*],
optype [CREATE], objtype [TABLE], objname [QATEST1.MYTABLE].

2011-01-20 15:11:41  GGS INFO     2100  DDL operation written to extract trail file.

2011-01-20 15:11:42  GGS INFO     2100  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
*/ ].

2011-01-20 15:11:43  GGS INFO     2100  DDL found, operation [create table myTableTemp (
    vid varchar2(100),
    someDate date,
    primary key (vid)
```

```
) ], start SCN [1186777], commit SCN [1186795] instance [test10g (1)], DDL seqno [4137].

2011-01-20 15:11:43  GGS INFO     2100  DDL operation excluded [EXCLUDE OBJNAME
myTableTemp OPTYPE CREATE], optype [CREATE], objtype [TABLE], objname
[QATEST1.MYTABLETEMP].
```

Replicat DDL レポート

Replicat レポートにリストされる内容は次のとおりです。

- Replicat が証跡から処理した各 DDL 操作の構文全体とソースの Oracle GoldenGate SCN。ソース SCN は、特にバックアップからのリストアが存在し、Replicat の証跡内の位置が過去の時点に設定される場合に追跡目的で使用できます。
- 操作のスコープ (MAPPED、UNMAPPED、OTHER) およびターゲット DDL 文でオブジェクト名がどのようにマップされたか (該当する場合) を示す後続のエントリ。
- 処理基準がどのように適用されたかを示す別のエントリ。
- 操作が成功したか失敗したかを示し、Replicat がエラー処理ルールを適用したかどうかを示す追加のエントリ。

次の例は Replicat レポートの一部で、エラー処理を含む一連の手順を示しています。

```
2011-01-20 15:11:45  GGS INFO     2104  DDL found, operation [drop table myTableTemp ],
Source SCN [1186713.0].

2011-01-20 15:11:45  GGS INFO    2100  DDL is of mapped scope, after mapping new operation
[drop table "QATEST2"."MYTABLETEMP" ].

2011-01-20 15:11:45  GGS INFO    2100  DDL operation included [include objname myTable*],
optype [DROP], objtype [TABLE], objname [QATEST2.MYTABLETEMP].

2011-01-20 15:11:45  GGS INFO     2100  Executing DDL operation.

2011-01-20 15:11:48  GGS INFO     2105  DDL error ignored for next retry: error code
[942], filter [include objname myTableTemp], error text [ORA-00942: table or view does
not exist], retry [1].

2011-01-20 15:11:48  GGS INFO     2100  Executing DDL operation , trying again due to
RETRYOP parameter.

2011-01-20 15:11:51  GGS INFO     2105  DDL error ignored for next retry: error code
[942], filter [include objname myTableTemp], error text [ORA-00942: table or view does
not exist], retry [2].

2011-01-20 15:11:51  GGS INFO     2100  Executing DDL operation, trying again due to
RETRYOP parameter.

2011-01-20 15:11:54  GGS INFO     2105  DDL error ignored for next retry: error code
[942], filter [include objname myTableTemp], error text [ORA-00942: table or view does
not exist], retry [3].

2011-01-20 15:11:54  GGS INFO     2100  Executing DDL operation, trying again due to
RETRYOP parameter.

2011-01-20 15:11:54  GGS INFO     2105  DDL error ignored: error code [942], filter
[include objname myTableTemp], error text [ORA-00942: table or view does not exist].
```

プロセス・レポートの統計

GGSCI で SEND コマンドを使用すると、DDL 処理に関する現在の統計を Extract と Replicat の各レポートに送信できます。

```
SEND {EXTRACT | REPLICAT} <group> REPORT
```

統計には、次の合計が表示されます。

- すべての DDL 操作
- MAPPED スコープの操作
- UNMAPPED スコープの操作
- OTHER スコープの操作
- 除外された操作 (包含された操作を差し引いた数)
- エラー (Replicat のみ)
- 再試行されたエラー (Replicat のみ)
- 破棄されたエラー (Replicat のみ)
- 無視された操作 (Replicat のみ)

```
From Table QATEST1.MYTABLE:
       #                   inserts:       100
       #                   updates:         0
       #                   deletes:         0
       #                  discards:         0

DDL replication statistics:

                    Operations:        18
             Mapped operations:         4
           Unmapped operations:         0
            Default operations:         0
           Excluded operations:         0
```

DDL 履歴表のメタデータの表示

DDL 履歴表に含まれている情報を表示するには、GGSCI で DUMPDDL コマンドを使用します。この情報は独自の形式で保存されていますが、判読可能な形式で画面に出力したり、問合せ可能な一連の SQL 表にエクスポートすることができます。DDL 履歴表の情報は、Extract プロセスで使用される情報と同じです。

履歴データは DDL の *before* トリガーによって取得されるため、DDL 変更前のオブジェクトの状態を反映しています。したがって、CREATE 操作のデータはありません。

DUMPDDL は、DDL 履歴表のレコードごとに先頭の約 4000 バイトをダンプします。出力をフィルタするには、SQL 問合せを使用するか、リダイレクトされた標準出力を検索します。

メタデータの形式は文字列ベースです。完全にエスケープされ、オブジェクト名や列名では標準以外の文字 (=、?、* など) がサポートされます。

DUMPDDL を使用して DDL 履歴を表示する手順

1. GGSCI を実行します。

2. GGSCI では、履歴表の所有者としてデータベースにログインします。

```
DBLOGIN USERID <user>[, PASSWORD <password>]
```

3. DUMPDDL コマンドを発行します。

```
DUMPDDL [SHOW]
```

基本の DUMPDDL

基本の DUMPDDL コマンドでは、次の表にリストされている表にメタデータを送信します。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割り当てられた Oracle GoldenGate DDL スキーマによって所有されています (『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)。これらの表の構造を表示するには、SQL*Plus で DESC コマンドを使用してください。

表 21　DUMPDDL の出力表

表	説明
GGS_DDL_OBJECTS	DDL 操作のオブジェクトに関する情報が含まれます。SEQNO が主キーです。他のすべての GGS_DDL_ 表には、GGS_DDL_OBJECTS の外部キーである SEQNO 列が含まれます。
GGS_DDL_COLUMNS	DDL 操作のオブジェクトの列に関する情報が含まれます。
GGS_DDL_LOG_GROUPS	DDL 操作のオブジェクトのサプリメンタル・ログ・グループに関する情報が含まれます。
GGS_DDL_PARTITIONS	DDL 操作のオブジェクトのパーティションに関する情報が含まれます。
GGS_DDL_PRIMARY_KEYS	DDL 操作のオブジェクトの主キーに関する情報が含まれます。

SEQNO 列は、Extract と Replicat の各レポート・ファイルにリストされている DDL 順序番号です。DDL 履歴表を問い合せても取得できます。DDL 履歴表のデフォルト名は GGS_DDL_HIST です。

DUMPDDL SHOW

DUMPDDL に SHOW オプションを指定すると、履歴表の情報が標準出力形式で画面にダンプされます。GGS_DDL_ 出力表は生成されません。このコマンドでは、DDL 履歴表のすべてのレコードがダンプされます。

DDL 処理のトレース

Oracle GoldenGate テクニカル・サポートでサポート・ケースを開く場合、トレースを有効化するよう求められることがあります。次のパラメータで DDL トレースを制御します。

- TLTRACE で Extract トレースを制御
- TRACE と TRACE2 で Replicat トレースを制御

これらのパラメータには、DDL のトレースを DML のトレースから分離するオプションがあります。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。

DDL トリガーのトレース

Oracle GoldenGate DDL トリガーのアクティビティをトレースするには、次のツールを使用します。

- ggs_ddl_trace.log トレース・ファイル : Oracle GoldenGate では、Oracle の USER_DUMP_DEST ディレクトリにトレース・ファイルが作成されます。RAC では、各ノードにそのノードの DDL トレースを取得する独自のトレース・ファイルが存在します。トレース・ファイルは、次のように問い合せることができます。

  ```
  select value from sys.v_$parameter where name = 'user_dump_dest';
  ```

- ddl_tracelevel スクリプト : このスクリプトを編集して実行し、トレース・レベルを設定します。値 None では、致命的エラーとインストール・ログを除き、DDL トレースは生成されません。デフォルト値の 0 (ゼロ) では、最小限のトレース情報が生成されます。値が 1 または 2 の場合、トレース・ファイルには非常に多くの情報が生成されます。サポート・ケースの一環として Oracle GoldenGate テクニカル・サポートのアナリストから求められないかぎり、1 または 2 は使用しないでください。

- ddl_cleartrace スクリプト : このスクリプトを定期的に実行して、トレース・ファイルの増大によりディスク領域が過度に使用されるのを防ぎます。ファイルは削除されますが、Oracle GoldenGate によって別のファイルが作成されます。DDL トリガーは、Oracle ディレクトリの領域が少なくなるとトレース・ファイルへの書込みを停止し、領域が再度使用可能になると書込みを再開します。このスクリプトは、Oracle GoldenGate ディレクトリにあります。スクリプトを実行する前に、トレース・ファイルをバックアップしてください。

第 15 章

Teradata データベースでの DDL 同期の構成

このドキュメントについて

このドキュメントには、Teradata 環境内の Oracle GoldenGate ソリューションの設定に固有の情報が含まれています。Teradata データベースおよび Teradata のレプリケーション・ソリューションに関する基本的知識がある読者が対象です。また、次が正しく構成されていることを前提としています。

- リレー・サービス・ゲートウェイ (RSG)
- チェンジ・データ・キャプチャ (CDC)
- Teradata アクセス・モジュール (TAM)
- レプリケーション・グループ

Teradata データベースでのレプリケーションの構成方法の詳細は、Teradata のレプリケーション・ソリューションのドキュメントを参照してください。

DDL 同期の概要

Oracle GoldenGate では、DDL 操作のデータベース間の同期がサポートされます。DDL 同期は次の場合にアクティブになります。

- ビジネス・アプリケーションがソース・オブジェクトとターゲット・オブジェクトにアクティブにアクセスして更新している場合。
- Oracle GoldenGate のトランザクション・データ同期がアクティブな場合。

DDL のレプリケーションをサポートするコンポーネントとトランザクション・データ変更 (DML) のレプリケーションをサポートするコンポーネントは相互に独立しています。したがって、次の同期を行えます。

- DDL 変更のみ
- DML 変更のみ
- DDL と DML の両方

たとえば、バッチ実行を使用してターゲット・オブジェクトを現在の状態に維持する場合、DDL 同期を継続的な (オンライン) 実行として構成すれば、バッチ・ロードの実行時にターゲット・メタデータを常に最新の状態に保つことができます。Oracle GoldenGate のバッチ・ロードでは、ソース・カタログとターゲット・カタログの現在のメタデータを使用します。

Teradata での DDL サポートでサポートされるオブジェクトと操作のリストは、『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。

Oracle GoldenGate DDL サポートの制限

DDL 文の長さ

Oracle GoldenGate では、DDL 文の長さが文字ではなくバイトで測定されます。サポートされる長さは約 2MB で、これには、対象となるオブジェクトの名前やその DDL タイプなどの特性に応じてサイズが変化する可能性のある内部的なオーバーヘッドが含まれます。DDL がサポートされるサイズよりも長い場合、Extract は警告を発行してその DDL 操作を無視します。

システム構成

- Oracle GoldenGateでは、2つのシステム間でのみ一方向構成および双方向構成(アクティブ/パッシブ型とアクティブ / アクティブ型) の DDL レプリケーションがサポートされます。
- Oracle GoldenGate では、同類構成でのみ DDL 同期がサポートされます。Oracle GoldenGate DDL サポートには、次の要件があります。
  - ソースとターゲットのオブジェクト定義は同一である必要があります。
  - Replicat パラメータ・ファイルで ASSUMETARGETDEFS パラメータが使用されている必要があります。オブジェクトが DDL サポート用に構成され、SOURCEDEFS パラメータが使用されている場合、Replicat は異常終了します。ASSUMETARGETDEFS 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。

フィルタリング、マッピングおよび変換

DDL

DDL 操作は Oracle GoldenGate プロセスでは変換できません。単純な文字列置換を使用したり、次のようにスキーマ名とオブジェクト名をマップすることは可能です。

- プライマリの Extract または Replicat プロセスではソース DDL を別のターゲット・オブジェクトにマップしたりフィルタしたりできますが、データ・ポンプ Extract または VAM ソート Extract で DDL のマッピングやフィルタリングを行うことはできません。DDL は、PASSTHRU モード (マッピングまたはフィルタリング不可能) のデータ・ポンプまたは VAM ソート Extract を通じて伝播されます。
- 結果として、特定の名前のソース表に対して実行される DDL (ALTER TABLE TableA... など) は、同じ表名 (ALTER TABLE TableA) でデータ・ポンプまたは VAM ソート Extract によって処理されます。TABLE 文の指定に関係なく、そのプロセスでは ALTER TABLE TableB としてマップできません。

DML

DDL はデータ・ポンプまたは VAM ソート Extract によって変更やマップされずに渡されるので、ソース表を別のターゲット名にマップする DML 操作を実行する場合は、この制限を考慮してください。プライマリの Extract または Replicat を使用して DML のフィルタリング、マッピングおよび変換を実行し、データ・ポンプまたは VAM ソート Extract でこれらの表を PASSTHRU モードに構成します。

データ・ポンプまたは VAM ソート Extract で、名前マッピングを含んだ DML 操作を実行する必要がある場合、これらの表に対するレプリケートされた DDL についても、同様の名前マッピングを実行するように Replicat を構成する必要があります。

注意 DDL サポートを使用しない表を NOPASSTHRU モードで構成すると、データのフィルタリングとデータ・ポンプによる操作が可能になります。

データの受渡し用に表を構成する手順

1. データ・ポンプまたは VAM ソート Extract のパラメータ・ファイルで、DDL サポートを使用する表が含まれているすべての TABLE 文の前に PASSTHRU パラメータを指定します。
2. データのフィルタリング、マッピングまたは変換を行う場合は、同じパラメータ・ファイルで、DDL サポートを使用しない表が含まれている TABLE 文の前に NOPASSTHRU パラメータを指定できます。
3. データ・ポンプまたは VAM ソート Extract に対して DDL 構成パラメータ (DDL、DDLOPTIONS、DDLSUBST、DDLERROR) や DDL オプションを指定した Oracle GoldenGate トレース・パラメータを使用しないでください。

PASSTHRU 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。

SQLEXEC

- SQLEXEC のストアド・プロシージャまたは問合せの影響を受けるすべてのオブジェクトは、SQL の実行前に適切な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 の実行前に発生している必要があります。
- スタンドアロンの SQLEXEC 文の影響を受けるすべてのオブジェクトは、Oracle GoldenGate プロセスが起動する前に存在している必要があります。このため、DDL サポートは、これらのオブジェクトに対して無効にする必要があります。そうしないと、SQLEXEC のプロシージャまたは問合せが実行される前に、DDL 操作によって構造が変更されたり、オブジェクトが削除される可能性があります。

ユーザー・イグジット

DDL 操作 (DDL が実行されたオブジェクトに関する情報や DDL 文のテキスト自体など) を戻すようにするには、GET_DDL_RECORD_PROPERTIES 関数を使用します。Extract プロセスでは、ソース表のレイアウトのみ取得可能です。Replicat プロセスでは、ソースまたはターゲットのレイアウトを取得できます。

このユーザー・イグジットは取得機能のみを備えています。Oracle GoldenGate には DDL レコードを操作する関数はありません。

DDL レスポンス時間

レプリケーションで取得される DDL 文のレスポンス時間は、Teradata データベースとレプリケーション・システム (Oracle GoldenGate コンポーネントを含む) 間の同期プロトコル固有の待機時間が原因で増加する可能性があります。レスポンス時間のオーバーヘッドは、通常、1 秒を超えることはありません。取得されない DDL のレスポンス時間には、それほど大きな影響はありません。UDT や LOB が含まれている表の変更データを取得するパフォーマンス・コストは、これらのデータ型が含まれていない表と比較して、他の方法でデータをエクスポートする場合とほぼ同じです。

DDL サポートに関する構成のガイドライン

データベースの権限

Oracle GoldenGate で DDL の取得とレプリケーションをサポートするために必要なデータベース権限は、『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。

初期同期

- DDL レプリケーションを構成するには、ソース・データベースと同期しているターゲット・データベースから作業を開始します。DDL サポートは、Replicat の初期ロード方法と互換性があります。
- DDL サポートは、Teradata の FastLoad 初期同期方法とも互換性があります。この機能の詳細は、Teradata のドキュメントを参照してください。
- 初期ロードを実行する前に、DDL の抽出およびレプリケーションを無効化します。DDL 処理は、Extract および Replicat のパラメータ・ファイルの DDL パラメータによって制御されます。

プロセス・トポロジ

- Extract または Replicat (あるいはその両方) のパラレル・プロセスを使用する場合、関連する DDL と DML を同じプロセス・ストリーム内にまとめ、データの整合性を保証します。プロセスは次のように構成します。
  - 任意のオブジェクトに対するすべての DDL と DML を、同じ Extract グループおよび同じ Replicat グループで処理します。
  - 相互に関連するすべてのオブジェクトを同じプロセス・グループで処理します。

  たとえば、ReplicatA で Table1 に対する DML を処理する場合、Table1 に対する DDL も処理する必要があります。Table2 に Table1 の外部キーがある場合、その DML 操作と DDL 操作も ReplicatA で処理される必要があります。
- Extract グループで、異なる Replicat グループによって読み取られる複数の証跡に書き込む場合、Extract はすべての DDL をすべての証跡に送信します。各 Replicat グループを使用して DDL をフィルタするには、Replicat パラメータ・ファイルで DDL パラメータのフィルタ・オプションを使用します。

オブジェクト名

- Oracle GoldenGate では、マルチバイト・キャラクタや特殊な英数字 (!、$、# など) を含むオブジェクト名がサポートされます。制限が適用されるのは、オブジェクトが TABLE パラメータまたは MAP パラメータでマップされる場合です (これらのパラメータでは、一部の使用可能な特殊文字がサポートされないため)。MAP 文と TABLE 文のオブジェクトに対する DDL では、これらのパラメータの制限が継承されます。これらのパラメータの詳細は、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』の MAP および TABLE の説明を参照してください。
- Oracle GoldenGate の標準のアスタリスク・ワイルドカード (*) を使用して、DDL 同期をサポートする構成パラメータでオブジェクト名を指定できます。ワイルドカードを正しく処理するため、WILDCARDRESOLVE パラメータはデフォルトで DYNAMIC に設定されます。WILDCARDRESOLVE を他の設定にすると、DDL 操作を処理している Oracle GoldenGate プロセスが異常終了して、プロセス・レポートにエラーが書き込まれます。

- Oracle GoldenGate では、DDL 文で所有者とオブジェクト名を区切るドットの前、後または前後両方に空白を使用できます。ドットの両側に使用できる空白は 1 つのみです。たとえば、次が有効です。

```
CREATE TABLE fin . customers...
CREATE TABLE fin. customers...
CREATE TABLE fin .customers...
```

DDL スコープの理解

データベース・オブジェクトは*スコープ*に分類されます。スコープとは、オブジェクトに対する DDL 操作を Oracle GoldenGate で処理する方法を定義するカテゴリです。次のスコープがあります。

- MAPPED
- UNMAPPED
- OTHER

スコープを使用することで、DDL 操作のフィルタリング、文字列置換およびエラー処理を詳細に制御できます。

MAPPED スコープ

TABLE 文と MAP 文で指定されるオブジェクトは、*MAPPED スコープ*です。これらの文の抽出指示とレプリケーション指示は、オーバーライド・ルールが適用されないかぎり、指定したオブジェクトに対するデータ (DML) と DDL の両方に適用されます。

TABLE 文と MAP 文のオブジェクトでは、次の表にリストされている DDL 操作がサポートされます。

表 22　MAP 文と TABLE 文でマップできるオブジェクト

操作	オブジェクト[1]
CREATE	TABLE (AS SELECT を含む)
ALTER	INDEX[3]
DROP	TRIGGER
RENAME	VIEW
COMMENT ON[2]	FUNCTION
	PROCEDUREMACRO
GRANT	TABLE
REVOKE	

[1] TABLE および MAP では、これらの操作の対象となるオブジェクト名に使用される可能性のある一部の特殊文字がサポートされません。このような文字のリストは、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』の MAP パラメータと TABLE パラメータの説明を参照してください。サポートされない特殊文字が含まれるオブジェクトは、UNMAPPED スコープと OTHER スコープでサポートされます。

[2] COMMENT ON TABLE、COMMENT ON COLUMN に適用されます。

[3] 索引に対する DDL は、HASH 操作と JOIN 操作ではサポートされません。

Extract の場合、MAPPED スコープでは TABLE 文の指示に従ってオブジェクトが DDL 取得用にマークされます。Replicat の場合、MAPPED スコープでは DDL がレプリケーション用にマークされ、MAP 文の TARGET 句の所有者と名前で指定されたオブジェクトにマップされます。

次の TABLE 文と MAP 文があるとします。

Extract (ソース)	**Replicat (ターゲット)**
`TABLE fin.expen;`	`MAP fin.expen, TARGET fin2.expen2;`
`TABLE hr.tab*;`	`MAP hr.tab*, TARGET hrBackup.bak_*;`

さらに、次のソース DDL 文があるとします。

```
ALTER TABLE fin.expen ADD notes varchar2(100);
```

この例では、別の所有者と表名にマップする TARGET 句が指定された MAP 文にソース表 fin.expen があるため、ターゲットの DDL 文は次のようになります。

```
ALTER TABLE fin2.expen2 ADD notes varchar2(100);
```

同様に、この例の TABLE 文と MAP 文の 2 番目のセットに対しては、次のソースおよびターゲットの DDL 文を使用できます。

ソース : `CREATE TABLE hr.tabPayables ... ;`

ターゲット : `CREATE TABLE hrBackup.bak_tabPayables ...;`

MAPPED スコープのオブジェクトでは、DDL サポートを詳細に調整しない場合、DDL 構成パラメータからオブジェクト名を省略できます。TABLE 文と MAP 文のオブジェクト名を変更する必要がある場合、それらのオブジェクトに対する DDL に変更が自動的に適用されます。

オブジェクトが TABLE 文に含まれ、MAP 文に含まれない場合、そのオブジェクトに対する DDL のスコープは、ソースでは MAPPED ですがターゲットでは UNMAPPED です。

ALTER INDEX のマッピング

ALTER INDEX...RENAME コマンドは、別のターゲット索引名にはマップできませんが、別のターゲット所有者にはマップできます。

有効な例 :

```
ALTER INDEX src.ind RENAME TO indnew;
```

この DDL は、ワイルドカードを使用して次のようにマップできます。

```
MAP src.* TARGET tgt.*;
```

また、次のように元の索引名をソースとターゲットの指定に使用し、明示的にマップすることもできます。

```
MAP src.ind TARGET tgt.ind;
```

前述のいずれの場合も、ターゲット DDL は次のようになります。

```
ALTER INDEX tgt.ind RENAME TO indnew;
```

無効な例 :

次のような MAP 文は無効です。

```
MAP src.ind TARGET tgt.indnew;
```

この文では、古い名前が新しい名前にマップされ、ターゲット DDL は次のようになります。

```
ALTER INDEX tgt.indnew RENAME TO indnew;
```

UNMAPPED スコープ

TABLE 文または MAP 文で DDL 操作の使用がサポートされていて、そのベース・オブジェクト名がこれらのパラメータのいずれかに含まれない場合は、*UNMAPPED スコープ*です。

オブジェクト名のスコープが、ソースでは UNMAPPED (Extract の TABLE 文にない)、ターゲットでは MAPPED (Replicat の MAP 文にある) になることも、その逆になることもあります。Teradata DDL のスコープが Replicat 構成で UNMAPPED の場合、次のいずれかの方法でターゲットに適用されます。

- 必須のReplicat接続パラメータTARGETDBにDSNのみが含まれ(tdtargなど)、データベース名が含まれない場合、ソース DDL と同じ所有者 (データベース名) とオブジェクト名を使用してターゲット・オブジェクトに適用されます。
- 特定のデータベース名が TARGETDB で使用されている場合 (db@tdtarg など)、TARGETDB の所有者を使用してすべての DDL 操作がターゲットに適用されます。

OTHER スコープ

マップできない DDL 操作は、*OTHER スコープ*です。DDL のスコープが Replicat 構成で OTHER の場合、ソース DDL と同じ所有者とオブジェクト名を使用してターゲットに適用されます。

OTHER スコープには、たとえば、システム固有の参照を作成する DDL 操作 (データ・ファイル名を操作する DDL など) があります。

DDL サポートの有効化

デフォルトでは、DDL レプリケーション・サポートのステータスは次のとおりです。

- ソースでは、Oracle GoldenGate DDL サポートはデフォルトで無効ですが、Teradata TAM から Oracle GoldenGate VAM にすべての DDL が送信されます。DDL パラメータを使用して、DDL を取得するように Extract を構成する必要があります。
- ターゲットでは、レプリケートされるトランザクション・データの整合性を保持するために、DDL サポートはデフォルトで有効です。デフォルトで、Replicat は証跡に含まれるすべての DDL 操作を処理します。必要に応じて DDL パラメータを使用し、DDL 操作を無視またはフィルタするように Replicat を構成できます。

DDL レプリケーションのフィルタリング

要件に応じて特定 (またはすべて) の DDL がターゲット・データベースに適用されるように DDL 操作をフィルタするには、DDL パラメータを使用します。

オプションなしで DDL パラメータを使用すると、フィルタリングは行われず、すべての DDL 操作が次のように伝播されます。

- (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・オブジェクトに対して生成された、サポートされているすべての DDL 操作を取得し、証跡に送信します。
- (Replicat パラメータとして) Oracle GoldenGate の証跡からすべての DDL 操作をレプリケートし、ターゲットに適用します。これは、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。

オプションを指定して使用すると、DDL パラメータはフィルタリング・エージェントとして機能し、次に基づいて DDL 操作を包含または除外します。

- スコープ
- 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、あるいはその両方の文字列

パラメータ・ファイルで使用できる DDL パラメータは 1 つのみですが、複数の包含オプションと除外オプションを組み合せることで、必要なレベルまで DDL をフィルタできます。

- 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 に対しては有効ですが、データ・ポンプ Extract に対しては無効です。
- 組み合せることで、複数のフィルタ・オプションの指定は AND 文として論理的に連結されます。
- DDL 文をレプリケートするには、複数のオプションを使用して指定されたフィルタ基準がすべて満たされる必要があります。
- 複雑な DDL フィルタリング基準を使用する場合、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。

> **注意** DDL パラメータ文を作成する前に、この章の「処理における DDL の評価方法」を確認すると役立ちます。

構文

```
DDL [
{INCLUDE | EXCLUDE}
    [, MAPPED | UNMAPPED | OTHER | ALL]
    [, OPTYPE <type>]
    [, OBJTYPE '<type>']
    [, OBJNAME "<name>"]
    [, INSTR '<string>']
]
[...]
```

表 23　DDL の包含オプションと除外オプション

オプション	説明

INCLUDE | EXCLUDE	INCLUDE および EXCLUDE を使用して、包含句または除外句の開始を示します。
	◆ 包含句には、このパラメータの対象となる DDL を識別するフィルタリング基準が含まれます。
	◆ 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が含まれます。
	包含句または除外句は、INCLUDE または EXCLUDE キーワードの後に、適用されるパラメータの有効なオプションの組合せを指定して構成する必要があります。
	
	EXCLUDE を使用する場合、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.*"`
	ただし、次のいずれかは使用できます。
	`DDL INCLUDE ALL, EXCLUDE OBJNAME "hr.*"`
	`DDL INCLUDE OBJNAME "fin.*" EXCLUDE "fin.ss"`
	EXCLUDE は、同じ基準が含まれている INCLUDE よりも優先されます。複数の包含句と除外句を使用できます。

MAPPED | UNMAPPED | OTHER | ALL	DDL 操作のスコープに基づいて INCLUDE または EXCLUDE を適用するには、MAPPED、UNMAPPED、OTHER および ALL を使用します。 ◆ MAPPED では、INCLUDE または EXCLUDE が MAPPED スコープの DDL 操作に適用されます。MAPPED フィルタリングは、他の DDL パラメータ・オプションを使用して指定されたフィルタリングの前に実行されます。 ◆ UNMAPPED では、INCLUDE または EXCLUDE が UNMAPPED スコープの DDL 操作に適用されます。 ◆ OTHER では、INCLUDE または EXCLUDE が OTHER スコープの DDL 操作に適用されます。 ◆ ALL では、INCLUDE または EXCLUDE がすべてのスコープの DDL 操作に適用されます。
OPTYPE <type>	INCLUDE または EXCLUDE を特定のタイプの DDL 操作 (CREATE、ALTER、RENAME など) に適用するには、OPTYPE を使用します。<type> には、データベースに有効な任意の DDL コマンドを使用します。たとえば、ALTER 操作を含める場合の正しい構文は次のようになります。 `DDL INCLUDE OPTYPE ALTER`

表 23　DDL の包含オプションと除外オプション (続き)

オプション	説明

`OBJTYPE '<type>'`	INCLUDE または EXCLUDE を特定のタイプのデータベース・オブジェクトに適用するには、OBJTYPE を使用します。<type> には、データベースに有効な任意のオブジェクト・タイプ (TABLE、INDEX、TRIGGER など) を使用します。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・ログの場合、正しいタイプはそれぞれ snapshot と 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 には OBJNAME オプションを使用しないでください (OBJNAME には owner.object が必要ですが、USER はスキーマのみを持つためです)。

`OBJNAME "<name>"`	INCLUDE または EXCLUDE をオブジェクトの完全修飾名 (owner.table_name など) に適用するには、OBJNAME を使用します。このオプションでは、二重引用符で囲まれた文字列を入力として使用します。
	ワイルドカードは、オブジェクト名にのみ使用できます。
	例 :
	`DDL INCLUDE OBJNAME "accounts.*"`
	Oracle USER オブジェクトには OBJNAME を使用しないでください (OBJNAME には owner.object が必要ですが、USER はスキーマのみを持つためです)。
	Replicat パラメータ・ファイルで OBJNAME と MAPPED を組み合せて使用する場合、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.*" です。
	`MAP fin.exp_*, TARGET fin2.*;`
	
	次の例では、CREATE TABLE 文はソースで次のように実行されます。
	`CREATE TABLE fin.exp_phone;`
	ターゲットでは次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たデータベース・ユーザーにマップします。
	
	トリガー、および索引を作成する DDL の場合、OBJNAME の値はトリガー、または索引の名前ではなく、ベース・オブジェクトの名前にする必要があります。
	たとえば、次の DDL 文を含める場合、正しい値は hr.insert_trig ではなく hr.accounts です。
	`CREATE TRIGGER hr.insert_trig ON hr.accounts;`
	RENAME 操作では、OBJNAME の値を新しい表名にする必要があります。たとえば、次の DDL 文を含める場合、正しい値は hr.acct です。
	`ALTER TABLE hr.accounts RENAME TO acct;`

表 23　DDL の包含オプションと除外オプション (続き)

オプション	説明
INSTR '<string>'	INCLUDE または EXCLUDE を、コマンド構文内に特定の文字列を含む DDL 文に適用するには、INSTR を使用します。たとえば、次の例では、索引を作成する DDL は除外されます。 `DDL INCLUDE ALL EXCLUDE INSTR 'CREATE INDEX'` 文字列は一重引用符で囲みます。文字列の検索では、大 / 小文字は区別されません。 INSTR では、文字列内に含まれる一重引用符 (' ') も、NULL 値もサポートされません。

INSTRWORDS '<word list>'	INCLUDE または EXCLUDE を特定の語を含む DDL 文に適用するには、INSTRWORDS を使用します。
	<word list> には、一重引用符内に任意の順序で語を指定します。空白を含めるには、空白を (語がある場合は語も) 二重引用符で囲みます。二重引用符は、文を囲む場合にも使用できます。
	INSTRWORDS が有効になるには、指定された語がすべて DDL に存在する必要があります。
	例 :
	`ALTER TABLE INCLUDE INSTRWORDS 'ALTER CONSTRAINT " xyz"`
	この例は、次に一致します。
	`ALTER TABLE ADD CONSTRAINT xyz CHECK`
	および
	`ALTER TABLE DROP CONSTRAINT xyz`
	INSTRWORDS では、文字列内に含まれる一重引用符 (' ') も、NULL 値もサポートされません。

DDL パラメータ・オプションの組合せ

次に、DDL パラメータ・オプションの組合せ方の例を示します。

```
DDL  &
INCLUDE UNMAPPED &
    OPTYPE alter &
    OBJTYPE 'table' &
    OBJNAME "users.tab*" &
INCLUDE MAPPED OBJNAME "*" &
EXCLUDE MAPPED OBJNAME "temporary.tab*"
```

この文で組み合されたフィルタ基準では、次のように指定されます。

- TABLE 文または MAP 文でマップされていない (UNMAPPED スコープ) 表に対するすべての ALTER TABLE 文が INCLUDE されます。
  - ただし、表が users によって所有され、その名前が tab で始まる場合のみです。
- TABLE 文または MAP 文でマップされている (MAPPED スコープ) すべての表に対するすべての DDL 操作タイプが INCLUDE されます。

- MAPPED スコープのすべての表に対するすべての DDL 操作タイプが EXCLUDE されます。
  - ただし、これらの表が temporary によって所有される場合のみです。
  - かつ、その名前が tab で始まる場合のみです。

DDL EXCLUDE ALL

DDL EXCLUDE ALL は、DDL 操作自体のレプリケーションは行わずに、Oracle GoldenGate のオブジェクト・メタデータを最新に保つ特別な処理オプションです。Oracle GoldenGate 以外の方法を使用して DDL をターゲットに適用し、ターゲット・オブジェクトへのデータ変更を Oracle GoldenGate でレプリケートする場合に、DDL EXCLUDE ALL を使用できます。現在のメタデータはオブジェクトの変更として Oracle GoldenGate に提供されるため、Oracle GoldenGate プロセスを停止したり起動する必要がありません。DDL EXCLUDE ALL には次の特別な条件が適用されます。

- DDL EXCLUDE ALL では、INCLUDE 句を使用する必要はありません。
- DDL EXCLUDE ALL を使用する場合、WILDCARDRESOLVE パラメータを IMMEDIATE に設定すると、必要に応じて即座に DML を解決できるようになります。

DDL メタデータと操作がすべてレプリケートされないようにするには、DDL パラメータ全体を省略します。

Oracle GoldenGate による導出オブジェクト名の処理方法

DDL 操作には、*ベース・オブジェクト*名に加え、*導出オブジェクト*名を含めることができます。ベース・オブジェクトは、データが格納されたオブジェクトです。導出オブジェクトは、ベース・オブジェクトの一部の属性を継承し、そのオブジェクトに関連する機能を実行するオブジェクトです。ベース・オブジェクトと導出オブジェクトの両方を含む DDL 文は次のとおりです。

- RENAME
- 索引またはトリガーに対する CREATE および DROP

次の DDL 文について考えてみます。

```
CREATE INDEX hr.indexPayrollDate ON TABLE hr.tabPayroll (payDate);
```

この場合、表がベース・オブジェクトです。その名前 (hr.tabPayroll) が*ベース名*で、MAPPED スコープでの TABLE または MAP によるマッピングの対象です。導出オブジェクトは索引で、その名前 (hr.indexPayrollDate) が*導出名*です。

導出名は、ベース・オブジェクトとは別に、独自の TABLE 文または MAP 文でマップできます。または、1 つの MAP 文で両方を処理できます。MAP の場合、ターゲットでの導出オブジェクト名の変換は次のように処理されます。

ベース・オブジェクトに対する MAP はあるが、導出オブジェクトにはない場合

ベース・オブジェクトに対する MAP 文はあるが、導出オブジェクトにはない場合、導出オブジェクトの*暗黙的マッピング*が行われます。DDL 文に MAPPED が含まれる場合、Replicat はベース・オブジェクトと同じターゲット所有者を導出オブジェクトに割り当てます。導出オブジェクトの名前は、ソース文と同じです。たとえば、次のとおり仮定します。

Extract (ソース)	**Replicat (ターゲット)**
`TABLE hr.tab*;`	`MAP hr.tab*, TARGET hrBackup.*;`

次のソース DDL 文があるとします。

```
CREATE INDEX hr.indexPayrollDate ON TABLE hr.tabPayroll (payDate);
```

ターゲットで Replicat によって実行される CREATE INDEX 文は次のとおりです。

```
CREATE INDEX hrBackup.indexPayrollDate ON TABLE hrBackup.tabPayroll (payDate);
```

暗黙的マッピングのルールは、導出オブジェクトにベース・オブジェクトと同じ所有者を割り当てるという一般的な方法に基づきます。また、ベース・オブジェクトと同じターゲット所有者が索引を所有している場合、暗黙的マッピングで導出オブジェクト名を明示的にマップする必要はありません。

ベース・オブジェクトと導出オブジェクトに対する MAP がある場合

ベース・オブジェクトに対しても、導出オブジェクトに対しても MAP 文がある場合、*明示的マッピング*が行われます。DDL 文に MAPPED が含まれる場合、Replicat は独自の TARGET 句に従って各オブジェクトの所有者と名前を変換します。たとえば、次のとおり仮定します。

Extract (ソース)	**Replicat (ターゲット)**
`TABLE hr.tab*;`	`MAP hr.tab*, TARGET hrBackup.*;`
`TABLE hr.index*;`	`MAP hr.index*, TARGET hrIndex.*;`

次のソース DDL 文があるとします。

```
CREATE INDEX hr.indexPayrollDate ON TABLE hr.tabPayroll (payDate);
```

ターゲットで Replicat によって実行される CREATE INDEX 文は次のとおりです。

```
CREATE INDEX hrIndex.indexPayrollDate ON TABLE hrBackup.tabPayroll (payDate);
```

ターゲットの索引がベース・オブジェクトとは異なる所有者によって所有される必要がある場合、またはターゲットとソースとで名前を変える必要がある場合は、明示的マッピングを使用します。

導出オブジェクトに対する MAP はあるが、ベース・オブジェクトにはない場合

導出オブジェクトに対する MAP 文はあるが、ベース・オブジェクトにはない場合、Replicat はどちらのオブジェクトに対しても名前変換を行いません。ターゲットの DDL 文は、ソースと同じです。導出オブジェクトをマップするには、次の方法があります。

- ベース・オブジェクトに対する明示的な MAP 文を使用します。
- 名前に問題がなければ、ワイルドカードを使用してベースと導出の両方のオブジェクトを同じ MAP 文でマップします。
- 名前の変換方法に応じて、各オブジェクトに対する MAP 文を作成します。

導出オブジェクトとしての新規表

次のものから作成される新しい表の Oracle GoldenGate による処理方法について、次に説明します。

- RENAME
- CREATE TABLE AS SELECT

RENAME

RENAME 操作では、ベース・オブジェクトは常に新しい表の名前です。次の例では、ベース・オブジェクト名は index_paydate とみなされます。

```
RENAME hr.indexPayrollDate TO index_paydate;
```

導出オブジェクト名は hr.indexPayrollDate です。

DDL レプリケーションに関連する名前変更の詳細は、207 ページの「名前変更を DDL 構成に反映するかどうかの制御」を参照してください。

CREATE TABLE AS SELECT

CREATE TABLE AS SELECT 文には、基になる任意の数のオブジェクトを対象とする SELECT 文と INSERT 文が含まれます。Oracle GoldenGate は、ターゲットで、AS SELECT 句に対応するデータをターゲット・データベースから取得します。AS SELECT 句のオブジェクトがターゲット・データベースに存在し、その名前がソースと同一である必要があります。

Oracle GoldenGate では、MAP 文で新しい表の名前 (CREATE TABLE <name>) のみ TARGET の指定にマップし、AS SELECT 句で取得した基になるオブジェクトの名前はマップしません。それらのオブジェクトに依存性があり、名前が TARGET の指定に変換されると、データに矛盾が生じる可能性があります。

次に、ソースの CREATE TABLE AS SELECT 文の例と、それが Oracle GoldenGate によってどのようにターゲットにレプリケートされるかを示します。

```
CREATE TABLE a.tab1 AS SELECT * FROM a.tab2;
```

Replicat の MAP 文は次のとおりです。

```
MAP a.tab*, TARGET a.x*;
```

Replicat によって適用されるターゲットの DDL 文は次のとおりです。

```
CREATE TABLE a.xtab1 AS SELECT * FROM a.tab2;
```

次のようにはなりません。

```
CREATE TABLE a.xtab1 AS SELECT * FROM a.xtab2;
```

AS SELECT * FROM 句の表名は、ソースと同じまま (tab2) です。

基になるオブジェクトのデータがソースとターゲットで一致しているようにするには、それらを Oracle GoldenGate のデータ・レプリケーション用に構成します。前述の例では、次の文を使用してこの要件を満たすことができます。

ソース	ターゲット

`TABLE a.tab*;`	`MAPEXCLUDE a.tab2`
	`MAP a.tab*, TARGET a.x*;`
	`MAP a.tab2, TARGET a.tab2;`

導出オブジェクトのマッピングの無効化

導出オブジェクトが含まれている MAP 文の TARGET 句に従ってその名前が変換されないようにするには、DDLOPTIONS パラメータに NOMAPDERIVED オプションを使用します。NOMAPDERIVED は、ベース・オブジェクトまたは導出オブジェクトの名前を含む明示的な MAP 文をオーバーライドします。導出オブジェクトが含まれているソース DDL は、ソースと同じ所有者とオブジェクト名でターゲットにレプリケートされます。

MAP 文がベース・オブジェクトのみ、または導出オブジェクトのみに対するものなのか、あるいはその両方に対するものかに基づいて、MAPDERIVED と NOMAPDERIVED を比較した結果を次の表に示します。

表 24　マッピング構成に基づくターゲットでの [NO]MAPDERIVED の結果

ベース・オブジェクト	導出オブジェクト	MAP/NOMAP DERIVED	導出オブジェクトが MAP によって変換されるか	導出オブジェクトにベース・オブジェクトの所有者が割り当てられるか
マップ対象[1]	マップ対象	MAPDERIVED	はい	いいえ
マップ対象	マップ対象外	MAPDERIVED	いいえ	はい
マップ対象外	マップ対象	MAPDERIVED	いいえ	いいえ
マップ対象外	マップ対象外	MAPDERIVED	いいえ	いいえ
マップ対象	マップ対象	NOMAPDERIVED	いいえ	いいえ
マップ対象	マップ対象外	NOMAPDERIVED	いいえ	いいえ
マップ対象外	マップ対象	NOMAPDERIVED	いいえ	いいえ
マップ対象外	マップ対象外	NOMAPDERIVED	いいえ	いいえ

[1]「マップ対象」は、MAP 文に含まれることを意味します。

次の例は、MAPDERIVED と NOMAPDERIVED を比較した結果を示しています。

次の表では、ベース名と導出名の両方が MAPDERIVED によって変換されるため、ターゲットではトリガーと表の両方が rpt によって所有されます。

表 25　導出オブジェクト名のデフォルト・マッピング (MAPDERIVED)

MAP 文	ソース DDL 文 (Extract によって取得)	ターゲット DDL 文 (Replicat によって適用)
`MAP fin.*, TARGET rpt.*;`	`CREATE TRIGGER fin.act_trig ON fin.acct;`	`CREATE TRIGGER rpt.act_trig ON rpt.acct;`

次の表では、NOMAPDERIVED の使用により変換が行われなくなるため、トリガーは fin によって所有されます。

表 26　NOMAPDERIVED 使用時の導出オブジェクト名のマッピング

MAP 文	ソース DDL 文 (Extract によって取得)	ターゲット DDL 文 (Replicat によって適用)
`MAP fin.*, TARGET rpt.*;`	`CREATE TRIGGER fin.act_trig ON fin.acct;`	`CREATE TRIGGER fin.act_trig ON rpt.acct;`

注意　RENAME 文では、新しい表名がベース表名とみなされ、古い表名が導出表名とみなされます。

DDL 文字列置換の使用

Oracle GoldenGate によって処理される際、DDL 操作内で文字列を置換できます。この機能は、データ構造に直接関連しないディレクトリ名などを変更したりマップする場合に便利です。文字列置換は、DDLSUBST パラメータによって制御されます。

DDLSUBST の使用に関するガイドライン

- DDLSUBST を使用して、ターゲットで列名やデータ型を別のものに変換しないでください。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、データがレプリケートされる際にエラーが発生します。同様に、DDLSUBST を使用して、ターゲットの DDL 文の所有者と表名を変更しないでください。レプリケートされた DDL 操作を別のターゲット・オブジェクトにマップする場合は、常に MAP 文を使用してください。
- DDLSUBST は、パラメータ・ファイル内の相対順序にかかわらず、常に DDL パラメータの後に実行されます。フィルタリングが最初に実行されるので、文字列置換に使用する基準と互換性のあるフィルタリング基準を使用します。たとえば、次のパラメータ文について考えてみます。

  ```
  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ので置換は行われません。fin が所有するオブジェクトは Oracle GoldenGate DDL 構成に含まれますが、sales が所有するオブジェクトは含まれません。
- 複数の DDLSUBST パラメータを使用できます。これらは、パラメータ・ファイルにリストされている順序で実行されます。

- 置換には、データベースの制限以外、最大文字列サイズの制限はありません。文字列サイズがデータベースの制限を超える場合、操作を実行中の Extract または Replicat のプロセスは異常終了します。

注意 DDLSUBST パラメータ文を作成する前に、この章の「処理における DDL の評価方法」を確認すると役立ちます。

構文

```
DDLSUBST '<search_string>' WITH '<replace_string>'
[INCLUDE <inclusion clause> | EXCLUDE <exclusion clause>]
```

引数	説明
`'<search_string>'`	ソース DDL 文内の置換対象の文字列。文字列は一重引用符で囲みます。文字列内で一重引用符を表す場合は、二重引用符を使用します。
`WITH`	必須キーワード。
`'<replace_string>'`	ターゲット DDL 内の置換用文字列。文字列は一重引用符で囲みます。文字列内で一重引用符を表す場合は、二重引用符を使用します。
`INCLUDE <inclusion clause> | EXCLUDE <exclusion clause>`	文字列置換ルールが適用される DDL 操作をフィルタするには、INCLUDE 文と EXCLUDE 文を 1 つ以上使用します。次の表を参照してください。

表 27　DDL の包含オプションと除外オプション

オプション	説明

`INCLUDE | EXCLUDE`	INCLUDE および EXCLUDE を使用して、包含句または除外句の開始を示します。
	◆ 包含句には、このパラメータの対象となる DDL を識別するフィルタリング基準が含まれます。
	◆ 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が含まれます。
	包含句または除外句は、INCLUDE または EXCLUDE キーワードの後に、適用されるパラメータの有効なオプションの組合せを指定して構成する必要があります。
	
	EXCLUDE を使用する場合、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.*"`
	ただし、次のいずれかは使用できます。
	`DDL INCLUDE ALL, EXCLUDE OBJNAME "hr.*"`
	`DDL INCLUDE OBJNAME "fin.*" EXCLUDE "fin.ss"`
	EXCLUDE は、同じ基準が含まれている INCLUDE よりも優先されます。複数の包含句と除外句を使用できます。

`MAPPED | UNMAPPED | OTHER | ALL`	DDL 操作のスコープに基づいて INCLUDE または EXCLUDE を適用するには、MAPPED、UNMAPPED、OTHER および ALL を使用します。 ◆ MAPPED では、INCLUDE または EXCLUDE が MAPPED スコープの DDL 操作に適用されます。MAPPED フィルタリングは、他の DDL パラメータ・オプションを使用して指定されたフィルタリングの前に実行されます。 ◆ UNMAPPED では、INCLUDE または EXCLUDE が UNMAPPED スコープの DDL 操作に適用されます。 ◆ OTHER では、INCLUDE または EXCLUDE が OTHER スコープの DDL 操作に適用されます。 ◆ ALL では、INCLUDE または EXCLUDE がすべてのスコープの DDL 操作に適用されます。
`OPTYPE <type>`	INCLUDE または EXCLUDE を特定のタイプの DDL 操作 (CREATE、ALTER、RENAME など) に適用するには、OPTYPE を使用します。<type> には、データベースに有効な任意の DDL コマンドを使用します。たとえば、ALTER 操作を含める場合の正しい構文は次のようになります。 `DDL INCLUDE OPTYPE ALTER`

表 27　DDL の包含オプションと除外オプション (続き)

オプション	説明

`OBJTYPE '<type>'`	INCLUDE または EXCLUDE を特定のタイプのデータベース・オブジェクトに適用するには、OBJTYPE を使用します。<type> には、データベースに有効な任意のオブジェクト・タイプ (TABLE、INDEX、TRIGGER など) を使用します。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・ログの場合、正しいタイプはそれぞれ snapshot と 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 には OBJNAME オプションを使用しないでください (OBJNAME には owner.object が必要ですが、USER はスキーマのみを持つためです)。

`OBJNAME "<name>"`	INCLUDE または EXCLUDE をオブジェクトの完全修飾名 (owner.table_name など) に適用するには、OBJNAME を使用します。このオプションでは、二重引用符で囲まれた文字列を入力として使用します。
	ワイルドカードは、オブジェクト名にのみ使用できます。
	例 :
	`DDL INCLUDE OBJNAME "accounts.*"`
	Oracle USER オブジェクトには OBJNAME を使用しないでください (OBJNAME には owner.object が必要ですが、USER はスキーマのみを持つためです)。
	Replicat パラメータ・ファイルで OBJNAME と MAPPED を組み合せて使用する場合、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.*" です。
	`MAP fin.exp_*, TARGET fin2.*;`
	次の例では、CREATE TABLE 文はソースで次のように実行されます。
	`CREATE TABLE fin.exp_phone;`
	ターゲットでは次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たデータベース・ユーザーにマップします。
	トリガー、および索引を作成する DDL の場合、OBJNAME の値はトリガー、または索引の名前ではなく、ベース・オブジェクトの名前にする必要があります。
	たとえば、次の DDL 文を含める場合、正しい値は hr.insert_trig ではなく hr.accounts です。
	`CREATE TRIGGER hr.insert_trig ON hr.accounts;`
	RENAME 操作では、OBJNAME の値を新しい表名にする必要があります。たとえば、次の DDL 文を含める場合、正しい値は hr.acct です。
	`ALTER TABLE hr.accounts RENAME TO acct;`

表 27　DDL の包含オプションと除外オプション (続き)

オプション	説明
INSTR '<string>'	INCLUDE または EXCLUDE を、コマンド構文内に特定の文字列を含む DDL 文に適用するには、INSTR を使用します。たとえば、次の例では、索引を作成する DDL は除外されます。 `DDL INCLUDE ALL EXCLUDE INSTR 'CREATE INDEX'` 文字列は一重引用符で囲みます。文字列の検索では、大 / 小文字は区別されません。 INSTR では、文字列内に含まれる一重引用符 (' ') も、NULL 値もサポートされません。

INSTRWORDS '<word list>'	INCLUDE または EXCLUDE を特定の語を含む DDL 文に適用するには、INSTRWORDS を使用します。
	<word list> には、一重引用符内に任意の順序で語を指定します。空白を含めるには、空白を (語がある場合は語も) 二重引用符で囲みます。二重引用符は、文を囲む場合にも使用できます。
	INSTRWORDS が有効になるには、指定された語がすべて DDL に存在する必要があります。
	例 :
	`ALTER TABLE INCLUDE INSTRWORDS 'ALTER CONSTRAINT " xyz"`
	この例は、次に一致します。
	`ALTER TABLE ADD CONSTRAINT xyz CHECK`
	および
	`ALTER TABLE DROP CONSTRAINT xyz`
	INSTRWORDS では、文字列内に含まれる一重引用符 (' ') も、NULL 値もサポートされません。

例　次の例では、fin が所有する表の文字列 cust が文字列 customers に置換されます。

```
DDLSUBST 'cust' WITH 'customers'
INCLUDE ALL OBJTYPE 'table' OBJNAME "fin.*"
```

検索で大 / 小文字は区別されません。文字列内で一重引用符を表す場合は、二重引用符を使用します。

例　この例では複数の DDLSUBST パラメータを使用します。これらは、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。最終的には、文字列 a および b が c に置換されます。

```
DDLSUBST 'a' WITH 'b' INCLUDE ALL
DDLSUBST 'b' WITH 'c' INCLUDE ALL
```

名前変更を DDL 構成に反映するかどうかの制御

DDLOPTIONS パラメータに NOCROSSRENAME オプションを使用すると、Oracle GoldenGate 構成から除外されているオブジェクトの名前が構成内の既存の名前に変更されないようにするルールを適用できます。次に、名前変更が発生する場合の例を示します。

- TableA は除外され、tableB は含まれています。
- TableA の名前が tableB に変更されます。

オブジェクトの名前が Oracle GoldenGate 構成内の既存の名前に変更されると、Extract によって警告が発行されるため、ユーザーは適切な処理を実行できます (それを Oracle GoldenGate 構成内に保持するか、またはパラメータを適切に調整して除外できます)。たとえば、名前を変更したオブジェクトの構造が Oracle GoldenGate でサポートされていない場合にエラーを回避できるなど、この通知は役に立ちます。

NOCROSSRENAME は、次のものに対してグローバルに適用されます。

- パラメータ・ファイルの TABLE 文と TABLEEXCLUDE 文で指定されているすべてのオブジェクト
- Oracle GoldenGate DDL 構成から除外され、TABLE または TABLEEXCLUDE で指定されていないすべてのオブジェクト

DDLOPTIONS NOCROSSRENAME は、TABLEEXCLUDE パラメータに NORENAME オプションを使用した場合と同じ結果をもたらします。2 つのパラメータの相違点は、TABLEEXCLUDE NORENAME では TABLEEXCLUDE 文のオブジェクトのみが機能の対象となるため、NOCROSSRENAME より選択的な処理が可能なことです。

処理における DDL の評価方法

次に、ソース・システムとターゲット・システムでの Oracle GoldenGate による DDL 文の処理方法について説明します。Oracle GoldenGate パラメータの各種基準が処理される順序を示し、Extract と Replicat がそれぞれ DDL を処理する方法の違いを説明します。

Extract

1. Extract が DDL 操作を取得します。
2. Extract は、DDL パラメータを検索します。(この例では、存在するものとします。)
3. Extract は、ベース・オブジェクト名と導出オブジェクト名 (存在する場合) を取得します。
4. Extract は、DDL スコープ (MAPPED、UNMAPPED または OTHER) を判別します。次の場合は MAPPED です。
   - 操作とオブジェクト・タイプがマッピングでサポートされる場合。

     および

   - ベース・オブジェクト名または導出オブジェクト名、あるいはその両方 (RENAME の場合) が TABLE パラメータ内にある場合。

   次の場合は UNMAPPED です。

   - 操作とオブジェクト・タイプがマッピングでサポートされない場合。

     および

- ベース・オブジェクト名または導出オブジェクト名、あるいはその両方 (RENAME の場合) が TABLE パラメータ内にない場合。

これ以外の場合、操作は OTHER とみなされます。

5. Extract は、DDL パラメータに INCLUDE 句と EXCLUDE 句があるかどうかをチェックし、これらの句の DDL パラメータ基準を評価します。INCLUDE または EXCLUDE が TRUE と評価されるには、すべてのオプションが TRUE と評価される必要があります。次のようになります。
   - EXCLUDE 句が TRUE と評価される場合、Extract は DDL 操作を破棄して別の DDL 操作を評価します。この場合、処理手順は最初に戻ります。
   - INCLUDE 句が TRUE と評価される場合、または DDL パラメータに INCLUDE 句も EXCLUDE 句も含まれていない場合、Extract は DDL 操作を受け入れ、処理ロジックが続けられます。
6. Extract は、DDLSUBST パラメータを検索し、INCLUDE 句と EXCLUDE 句を評価します。これらの句の基準が最終的に TRUE になる場合、Extract は文字列置換を実行します。Extract は、パラメータ・ファイルの各 DDLSUBST 文に対して DDL 操作を評価します。TRUE と評価されたすべての DDLSUBST 文について、Extract は DDLSUBST パラメータがファイルにリストされている順序で文字列置換を実行します。
7. Extract は、DDL 文を証跡に書き込みます。

Replicat

1. Replicat が証跡から DDL 操作を読み取ります。
2. Replicat は、DDL 同期スコープを評価し、DDL が名前マッピングに適しているかどうかを判断します。他はすべて、OTHER スコープです。
3. Replicat は、パラメータ・ファイルの MAP 文を評価します。(証跡から読み取られた) この DDL のソースのベース・オブジェクト名が MAP 文のいずれかに存在する場合、操作は MAPPED スコープとしてマークされます。それ以外の場合、UNMAPPED スコープとしてマークされます。
4. Replicat は、ソースのベース・オブジェクト名を、MAP 文の TARGET 句に指定されたベース・オブジェクト名で置き換えます。
5. 導出オブジェクトがある場合、Replicat は DDLOPTIONS MAPDERIVED を検索します。存在する場合、Replicat はソースの導出名を MAP 文のターゲットの導出名で置き換えます。
6. Replicat は、DDL パラメータに INCLUDE 句と EXCLUDE 句があるかどうかをチェックし、それらに含まれる DDL パラメータ基準を評価します。INCLUDE または EXCLUDE が TRUE と評価されるには、すべてのオプションが TRUE と評価される必要があります。次のようになります。
   - EXCLUDE 句が TRUE と評価される場合、Replicat は DDL 操作を破棄して別の DDL 操作の評価を開始します。この場合、処理手順は最初に戻ります。
   - INCLUDE 句が TRUE と評価される場合、または DDL パラメータに INCLUDE 句も EXCLUDE 句も含まれていない場合、Replicat は DDL 操作を受け入れ、処理ロジックが続けられます。
7. Replicat は、DDLSUBST パラメータを検索し、INCLUDE 句と EXCLUDE 句を評価します。これらの句のオプションが最終的に TRUE になる場合、Replicat は文字列置換を実行します。Replicat は、パラメータ・ファイルの各 DDLSUBST 文に対して DDL 操作を評価します。TRUE と評価されたすべての DDLSUBST 文について、Replicat は DDLSUBST パラメータがファイルにリストされている順序で文字列置換を実行します。
8. Replicat は、ターゲット・データベースで DDL 操作を実行します。
9. エラーがなければ、Replicat は次の DDL 文を処理します。エラーがある場合、Replicat は次の手順を実行します。

10. Replicat は、Replicat の DDLERROR パラメータ文の INCLUDE ルールと EXCLUDE ルールを、パラメータ・ファイルに出現する順序で分析します。Replicat は、エラー・コードに対応するルールを検出すると、指定されたエラー処理を適用します。それ以外の場合は、DEFAULT 処理を適用します。
11. エラー処理を行っても DDL 操作を続行できない場合、Replicat はルールでの指定に応じて異常終了、操作の無視または破棄のいずれかを実行します。

注意 **同じソースに対して複数のターゲットが MAP 文にある場合、ターゲットごとに処理ロジックが実行されます。**

Extract の DDL 処理エラーへの対処

メタデータが見つからないオブジェクトに関するエラー (Extract で検出) を処理するには、DDLERROR パラメータの Extract オプションを使用します。

構文

```
DDLERROR [RESTARTSKIP <num skips>] [SKIPTRIGGERERROR <num errors>]
```

条件 :

- RESTARTSKIP は、起動時に複数の DDL 操作をスキップし、Extract がエラーによって異常終了するのを防ぎます。デフォルトでは操作をスキップしないので、Extract はエラーによって異常終了します。最大 100,000 の DDL 操作をスキップできます。

スキップされた操作に関する情報を Extract レポート・ファイルに書き込むには、DDLOPTIONS パラメータに REPORT オプションを使用します。

Replicat の DDL 処理エラーへの対処

DDL がターゲット・データベースに適用される際に発生するエラーを処理するには、DDLERROR パラメータの Replicat オプションを使用します。DDLERROR オプションを使用すると、ほとんどのエラーはデフォルトの方法 (処理の停止など) で処理でき、他のエラーも特定の方法で処理できます。同じパラメータ・ファイル内で DDLERROR の複数のインスタンスを使用し、予期されるすべてのエラーを処理できます。

<error>、DEFAULT および <response> の組合せを使用して、予期される DDL エラーと予期しない DDL エラーに対する Replicat のレスポンス方法に関するルールを作成します。必ず適切な包含句と除外句を指定して、目的の DDL にルールを適用してください。その後で、必要に応じて追加オプションを使用し、エラー処理を調整します。

構文

```
DDLERROR
{<error> | DEFAULT} {<response>}
{INCLUDE <inclusion clause> | EXCLUDE <exclusion clause>}
[IGNOREMISSINGOBJECTS | ABENDONMISSINGOBJECTS]
```

引数	説明
{<error> | DEFAULT} {<response>}	◆ <error> は、この文で処理する特定の DDL エラーです。 ◆ DEFAULT では、すべての DDL エラーを対象としたグローバルなレスポンスを設定します (ただし、DDLERROR 文が明示的に指定されているものを除く)。 ◆ <response> は、次のいずれかです。 ABEND 操作をロールバックして、処理を異常終了させます。ABEND がデフォルトです。 DISCARD 原因の操作を廃棄ファイルに記録し、後続の DDL の処理は続行します。DISCARDFILE パラメータで廃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>] 原因の操作を再試行します。再試行の回数を制御するには、MAXRETRIES オプションを使用します。MAXRETRIES の指定回数を超えると、Replicat は異常終了します。整数で指定します。 再試行間隔 (秒単位) を設定するには、RETRYDELAY を使用します。
{INCLUDE <inclusion clause> | EXCLUDE <exclusion clause>}	DDLERROR 文で特定の DDL を処理するか、または処理しないかを制御します。詳細は、次の表を参照してください。
[IGNOREMISSINGOBJECTS | ABENDONMISSINGOBJECTS]	ターゲット上で検出できなかったオブジェクトに対して DML が発行された場合に、Extract を異常終了させるかどうかを制御します。通常、このような状況が生じるのは、レプリケーション以外でターゲットに対して DDL が直接発行された場合、またはソース定義とターゲット定義との間に矛盾がある場合です。 IGNOREMISSINGOBJECTS を使用すると、Replicat は存在しない表に対する DML 操作をスキップします。 ABENDONMISSINGOBJECTS を使用すると、存在しない表に対する DML 操作によって、Replicat は異常終了します。

表 28　DDL の包含オプションと除外オプション

オプション	説明

INCLUDE | EXCLUDE	INCLUDE および EXCLUDE を使用して、包含句または除外句の開始を示します。
	◆ 包含句には、このパラメータの対象となる DDL を識別するフィルタリング基準が含まれます。
	◆ 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が含まれます。
	包含句または除外句は、INCLUDE または EXCLUDE キーワードの後に、適用されるパラメータの有効なオプションの組合せを指定して構成する必要があります。
	
	EXCLUDE を使用する場合、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.*"`
	ただし、次のいずれかは使用できます。
	`DDL INCLUDE ALL, EXCLUDE OBJNAME "hr.*"`
	`DDL INCLUDE OBJNAME "fin.*" EXCLUDE "fin.ss"`
	EXCLUDE は、同じ基準が含まれている INCLUDE よりも優先されます。複数の包含句と除外句を使用できます。

MAPPED | UNMAPPED | OTHER | ALL	DDL 操作のスコープに基づいて INCLUDE または EXCLUDE を適用するには、MAPPED、UNMAPPED、OTHER および ALL を使用します。 ◆ MAPPED では、INCLUDE または EXCLUDE が MAPPED スコープの DDL 操作に適用されます。MAPPED フィルタリングは、他の DDL パラメータ・オプションを使用して指定されたフィルタリングの前に実行されます。 ◆ UNMAPPED では、INCLUDE または EXCLUDE が UNMAPPED スコープの DDL 操作に適用されます。 ◆ OTHER では、INCLUDE または EXCLUDE が OTHER スコープの DDL 操作に適用されます。 ◆ ALL では、INCLUDE または EXCLUDE がすべてのスコープの DDL 操作に適用されます。
OPTYPE <type>	INCLUDE または EXCLUDE を特定のタイプの DDL 操作 (CREATE、ALTER、RENAME など) に適用するには、OPTYPE を使用します。<type> には、データベースに有効な任意の DDL コマンドを使用します。たとえば、ALTER 操作を含める場合の正しい構文は次のようになります。 `DDL INCLUDE OPTYPE ALTER`

表 28　DDL の包含オプションと除外オプション (続き)

オプション	説明

`OBJTYPE '<type>'`	INCLUDE または EXCLUDE を特定のタイプのデータベース・オブジェクトに適用するには、OBJTYPE を使用します。<type> には、データベースに有効な任意のオブジェクト・タイプ (TABLE、INDEX、TRIGGER など) を使用します。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・ログの場合、正しいタイプはそれぞれ snapshot と 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 には OBJNAME オプションを使用しないでください (OBJNAME には owner.object が必要ですが、USER はスキーマのみを持つためです)。

`OBJNAME "<name>"`	INCLUDE または EXCLUDE をオブジェクトの完全修飾名 (owner.table_name など) に適用するには、OBJNAME を使用します。このオプションでは、二重引用符で囲まれた文字列を入力として使用します。
	ワイルドカードは、オブジェクト名にのみ使用できます。
	例 :
	`DDL INCLUDE OBJNAME "accounts.*"`
	Oracle USER オブジェクトには OBJNAME を使用しないでください (OBJNAME には owner.object が必要ですが、USER はスキーマのみを持つためです)。
	Replicat パラメータ・ファイルで OBJNAME と MAPPED を組み合せて使用する場合、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.*" です。
	`MAP fin.exp_*, TARGET fin2.*;`
	次の例では、CREATE TABLE 文はソースで次のように実行されます。
	`CREATE TABLE fin.exp_phone;`
	ターゲットでは次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たデータベース・ユーザーにマップします。
	トリガー、および索引を作成する DDL の場合、OBJNAME の値はトリガー、または索引の名前ではなく、ベース・オブジェクトの名前にする必要があります。
	たとえば、次の DDL 文を含める場合、正しい値は hr.insert_trig ではなく hr.accounts です。
	`CREATE TRIGGER hr.insert_trig ON hr.accounts;`
	RENAME 操作では、OBJNAME の値を新しい表名にする必要があります。たとえば、次の DDL 文を含める場合、正しい値は hr.acct です。
	`ALTER TABLE hr.accounts RENAME TO acct;`

表 28　DDL の包含オプションと除外オプション (続き)

オプション	説明
`INSTR '<string>'`	INCLUDE または EXCLUDE を、コマンド構文内に特定の文字列を含む DDL 文に適用するには、INSTR を使用します。たとえば、次の例では、索引を作成する DDL は除外されます。 `DDL INCLUDE ALL EXCLUDE INSTR 'CREATE INDEX'` 文字列は一重引用符で囲みます。文字列の検索では、大 / 小文字は区別されません。 INSTR では、文字列内に含まれる一重引用符 (' ') も、NULL 値もサポートされません。

`INSTRWORDS '<word list>'`	INCLUDE または EXCLUDE を特定の語を含む DDL 文に適用するには、INSTRWORDS を使用します。
	<word list> には、一重引用符内に任意の順序で語を指定します。空白を含めるには、空白を (語がある場合は語も) 二重引用符で囲みます。二重引用符は、文を囲む場合にも使用できます。
	INSTRWORDS が有効になるには、指定された語がすべて DDL に存在する必要があります。
	例 :
	`ALTER TABLE INCLUDE INSTRWORDS 'ALTER CONSTRAINT " xyz"`
	この例は、次に一致します。
	`ALTER TABLE ADD CONSTRAINT xyz CHECK`
	および
	`ALTER TABLE DROP CONSTRAINT xyz`
	INSTRWORDS では、文字列内に含まれる一重引用符 (' ') も、NULL 値もサポートされません。

サンプルの DDLERROR 文

次の例の DDLERROR 文では、Replicat は 10 秒間隔で操作を 3 回試行してから、指定されたエラーを無視します。Replicat は、ワイルドカード "tab*" に一致する名前のオブジェクト (任意のユーザー、操作) に対して実行された DDL 操作にエラー処理を適用しますが、"tab1*" に一致するものは除外されます。

```
DDLERROR <error> IGNORE RETRYOP MAXRETRIES 3 RETRYDELAY 10 &
INCLUDE ALL OBJTYPE TABLE OBJNAME "tab*" EXCLUDE OBJNAME "tab1*"
```

このエラー以外のすべてのエラーを処理するには、次の DDLERROR 文を追加します。

```
DDLERROR DEFAULT ABENDS
```

この場合、DDL エラーにより Replicat は異常終了します。

複数の DDLERROR 文の使用

パラメータ・ファイル内で DDLERROR 文がリストされている順序は、それぞれ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じエラーが指定されている場合は例外です。このような場合、Replicat では最初にリストされているもののみが使用されます。たとえば、次の文では、エラーにより Replicat は異常終了します。

```
DDLERROR <error1> ABEND
DDLERROR <error1> IGNORE
```

ただし、適切な修飾子が指定されている場合は、前述の構成の方が便利です。次に例を示します。

```
DDLERROR <error1> ABEND INCLUDE OBJNAME "tab*"
DDLERROR <error1> IGNORE
```

この場合、INCLUDE 文があるので、エラーが発生した DDL 文内のオブジェクト名がワイルドカード "tab*" と一致する場合にのみ、Replicat は異常終了します。オブジェクトが他の名前の場合、Replicat はエラーの発生した操作を無視します。

DDL レポート情報の表示

Oracle GoldenGate では、Extract および Replicat の各レポートの最後に、DDL 操作に関する基本的な統計がデフォルトで表示されます。拡張 DDL レポートを有効化するには、DDLOPTIONS パラメータに REPORT オプションを使用します。拡張レポートには、DDL 処理に関する次の情報が含まれます。

- Oracle GoldenGate によって処理された DDL 操作の手順ごとの履歴
- 使用中の DDL フィルタリング・パラメータと処理パラメータ

拡張 DDL レポート情報によってレポート・ファイルのサイズは大きくなりますが、この情報は特定の状況で役立ちます (トラブルシューティングや、

プロセス・レポートを表示する手順

レポートを表示するには、GGSCI で VIEW REPORT コマンドを使用します。

```
VIEW REPORT <group>
```

Extract DDL レポート

Extract レポートにリストされる内容は次のとおりです。

- 取得された各 DDL 操作の構文全体、その Oracle GoldenGate CSN 番号、Teradata 順序番号および操作のサイズ (バイト単位)。
- 処理基準がどのように操作に適用されたかを示す後続のエントリ (文字列置換または INCLUDE と EXCLUDE のフィルタリングなど)。
- 操作が証跡に書き込まれたか、除外されたかを示す別のエントリ。

次に、Extract レポート・ファイルから取得した例を示します。

```
2011-01-21 18:41:40  GGS INFO       2100  DDL found, operation [DROP TABLE
"SMIJATOVDBS"."src13_tabtable_9" ; (size 59)], start CSN [2500FF3F0200363A], DDL seqno
[00000025000000000000381500000021].
2011-01-21 18:41:40  GGS INFO      2100  DDL operation included [include mapped objname
"*"], optype [DROP], objtype [TABLE], objowner [SMIJATOVDBS], objname
[SRC13_TABTABLE_9].
2011-01-21 18:41:40  GGS INFO       2100  DDL operation written to extract trail file.
```

Replicat DDL レポート

Replicat レポートにリストされる内容は次のとおりです。

- Replicat が証跡から処理した各 DDL 操作の構文全体。
- 操作のスコープ (MAPPED、UNMAPPED、OTHER) およびターゲット DDL 文でオブジェクト名がどのようにマップされたか (該当する場合) を示す後続のエントリ。
- 処理基準がどのように適用されたかを示す別のエントリ。
- 操作が成功したか失敗したかを示し、Replicat がエラー処理ルールを適用したかどうかを示す追加のエントリ。

次に、Replicat パラメータ・ファイルから取得した例を示します。

```
2011-01-21 18:41:44  GGS INFO       2104  DDL found, operation [DROP TABLE
"SMIJATOVDBS"."src13_tabtable_9" ; (size 59)].

2011-01-21 18:41:44  GGS INFO       2100  DDL is of mapped scope, after mapping new
operation [DROP TABLE "SMIJATOVDBT"."SRC13_TABTABLE_9" ; (size 59)].

2011-01-21 18:41:44  GGS INFO       2100  Executing DDL operation.


2011-01-21 18:41:44  GGS INFO       2105  DDL operation successful.
```

プロセス・レポートの統計

GGSCI で SEND コマンドを使用すると、DDL 処理に関する現在の統計を Extract と Replicat の各レポートに送信できます。

```
SEND {EXTRACT | REPLICAT} <group> REPORT
```

統計には、次の合計が表示されます。

- すべての DDL 操作
- MAPPED スコープの操作
- UNMAPPED スコープの操作
- OTHER スコープの操作
- 除外された操作 (包含された操作を差し引いた数)
- エラー (Replicat のみ)
- 再試行されたエラー (Replicat のみ)
- 破棄されたエラー (Replicat のみ)
- 無視された操作 (Replicat のみ)

```
From Table QATEST1.MYTABLE:
       #                   inserts:       100
       #                   updates:         0
       #                   deletes:         0
       #                  discards:         0

DDL replication statistics:

                    Operations:        18
             Mapped operations:         4
           Unmapped operations:         0
            Default operations:         0
           Excluded operations:         0
```

DDL 処理のトレース

Oracle GoldenGate テクニカル・サポートでサポート・ケースを開く場合、トレースを有効化するよう求められることがあります。次のパラメータで DDL トレースを制御します。

- TLTRACE で Extract トレースを制御
- TRACE と TRACE2 で Replicat トレースを制御

これらのパラメータには、DDL のトレースを DML のトレースから分離するオプションがあります。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。

第 16 章

初期データ・ロードの実行

初期データ・ロード方法の概要

Oracle GoldenGate を使用して、次の処理を実行できます。

- スタンドアロンのバッチ・ロードを実行して、移行などの目的でデータベース表にデータを移入できます。
- Oracle GoldenGate との変更同期に備えて、初期同期の実行の一環としてデータベース表にデータをロードできます。

初期ロードは、アクティブなソース・データベースから実行できます。ユーザーおよびアプリケーションは、ロードの実行中もデータにアクセスしてその内容を更新できます。ターゲット・ロードが完了するまでソース表へのアクセスを遅延する場合、静止されたソース・データベースから初期ロードを実行できます。

サポートされるロード方法

Oracle GoldenGate を使用して、次のいずれかの方法でデータをロードできます。

- 220 ページの「データベース・ユーティリティを使用したデータのロード」: ユーティリティによって初期ロードを実行します。
- 221 ページの「ファイルから Replicat へのデータのロード」: Extract は抽出ファイルにレコードを書き込み、Replicat はそれらのレコードをターゲット表に適用します。これは最も低速な初期ロード方法です。
- 226 ページの「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」: Extract は、抽出ファイルに外部 ASCII 形式でレコードを書き込みます。これらのファイルは、バルク・ロード・ユーティリティによってターゲット表に入力するためのデータ・ファイルとして使用されます。Replicat は、実行ファイルと制御ファイルを作成します。
- 231 ページの「Oracle GoldenGate ダイレクト・ロードを使用したデータのロード」: Extract は、Collector プロセスまたはファイルを使用することなく、TCP/IP を通じて Replicat と直接通信します。Replicat は、データベース・エンジンを通じてデータを適用します。
- 235 ページの「ダイレクト・バルク・ロードを使用した SQL*Loader へのデータのロード」: Extract は、外部 ASCII 形式のレコードを抽出して Replicat に直接配信します。Replicat は、それらのレコードを Oracle の SQL*Loader バルク・ロード・ユーティリティに配信します。これは、Oracle GoldenGate で Oracle データをロードする場合の最も高速な方法です。
- 239 ページの「Teradata ロード・ユーティリティを使用したデータのロード」: これは、2 つの Teradata データベースを同期する場合に推奨される方法です。推奨ユーティリティは、MultiLoad です。

初期ロードでのパラレル処理の使用

データベース・ユーティリティで実行される方法以外のすべての初期ロード方法では、Oracle GoldenGate のパラレル・プロセスを使用することで、大規模データベースをより高速にロードできます。

パラレル処理を使用する手順

1. この章の指示に従って、使用するパラレル・プロセスの各セットに対して初期ロード Extract および初期ロード Replicat を作成します。
2. TABLE パラメータおよび MAP パラメータを使用して、Extract プロセスと Replicat プロセスのペアごとに異なる表セットを指定します。または、TABLE の SQLPREDICATE オプションを使用して、異なる Extract プロセス間でサイズの大きい表の行を分割します。

初期ロードの前提条件

DDL 処理の無効化

初期ロードを実行する前に、DDL の抽出およびレプリケーションを無効化します。DDL 処理は、Extract および Replicat のパラメータ・ファイルの DDL パラメータによって制御されます。DDL サポートの詳細は、141 ページを参照してください。

ターゲット表の準備

次に、ロードを高速化してエラーを回避するために役立つ推奨事項を示します。

- **データ :** ターゲット表が空であることを確認します。それ以外の場合、既存の行とロードされた行の間で重複行エラーまたは競合が発生する可能性があります。
- **制約 :** 外部キー制約およびチェック制約を無効化します。外部キー制約ではエラーが発生する可能性があり、チェック制約ではロード・プロセスが低速化する可能性があります。制約は、ロードが正常に終了した後に再度有効化できます。
- **索引 :** ターゲット表から索引を削除します。索引は、挿入には必要ありません。索引によって、ロード・プロセスの速度が大幅に低下します。表に挿入される行ごとに、データベースではその表に対するすべての索引が更新されます。索引は、ロードが終了した後に再度追加できます。

  注意 1 次索引は、DB2 for z/OS のターゲット表にアクセスするすべてのアプリケーションに必要です。1 次索引以外の他のすべての索引は、ターゲット表から削除できます。

- **キー :** HANDLECOLLISIONS 機能を使用して増分データ変更とロードとを調整するには、各ターゲット表に主キーまたは一意キーが含まれている必要があります。アプリケーションを通じてキーを作成できない場合、TABLE パラメータおよび MAP パラメータの KEYCOLS オプションを使用して、Oracle GoldenGate 用の代替キーとなる列を指定します。キーは、処理対象の行の識別に役立ちます。キーを作成できない場合、ロードのためにソース・データベースを静止する必要があります。

Manager プロセスの構成

ソース・システムとターゲット・システムで、Manager プロセスを構成して起動します。1 つの Manager を、複数の初期ロード・プロセスと変更同期プロセスに使用できます。詳細は、23 ページの「Manager プロセスの構成」を参照してください。

データ定義ファイルの作成

データ定義ファイルは、ソース・データベースとターゲット・データベースに異なる定義が存在する場合に必要です。Oracle GoldenGate は、このファイルを使用して、データをターゲット・データベースで必要とされる形式に変換します。詳細は、第 11 章を参照してください。

変更同期グループの作成

注意　静止されたソース・データベースからロードを実行し、続けて継続的な変更同期を実行*しない*場合、これらのグループは省略できます。

初期ロード中のトランザクション変更の取得およびレプリケーションに備えるため、オンラインの Extract グループおよび Replicat グループを作成します。これらのグループは、ロード手順の実行中に起動できます。このドキュメントで、使用するレプリケーション構成のタイプに適した指示を参照してください。

初期ロードの指示で要求されるまで、Extract グループまたは Replicat グループは起動しないでください。変更同期によって、ロードが適用されている間にトランザクション変更が追跡され、その後ターゲット表がそれらの変更に応じて調整されます。

注意　Extract が新しい Oracle GoldenGate 構成で初めて起動する場合、すべてのオープン・トランザクションはスキップされます。Extract の起動後に開始されたトランザクションのみが取得されます。

ソース・データベースを初期ロード中もアクティブな状態のまま維持する場合、Replicat のパラメータ・ファイルに HANDLECOLLISIONS パラメータを含めます。それ以外の場合、このパラメータは使用しません。HANDLECOLLISIONS は、初期ロードと進行中の変更レプリケーションが重複する時間に発生する衝突を処理します。このパラメータは、行がすでに存在する場合の挿入操作や、行が存在しない場合の更新操作および削除操作を調整します。使用方法は次のとおりです。

- パラメータ・ファイルですべての表に対してグローバルに使用
- 表のグループに対してオンとオフを切り替えて使用
- 特定の表ペアのエラー処理を有効化または無効化するために MAP 文内で使用

このパラメータの詳細は、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』を参照してください。

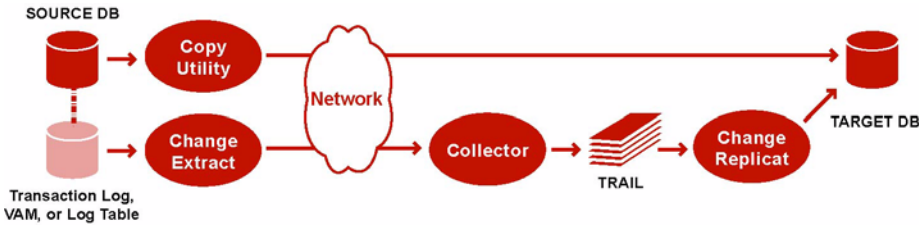
プロセス・グループ間でのパラメータの共有

変更同期パラメータ・ファイルで使用するパラメータの一部は、初期ロード Extract および初期ロード Replicat のパラメータ・ファイルでも必要です。これらのパラメータは、あるパラメータ・ファイルから別のパラメータ・ファイルにコピーできます。または、これらのパラメータを中央ファイルに格納し、各パラメータ・ファイルで OBEY パラメータを使用して取得できます。別の方法として、共有パラメータ用の Oracle GoldenGate マクロを作成し、そのマクロを MACRO パラメータを使用して各パラメータ・ファイルからコールできます。

OBEY の使用の詳細は、33 ページを参照してください。

マクロの詳細は、272 ページを参照してください。

データベース・ユーティリティを使用したデータのロード



データベース・コピー・ユーティリティを使用してターゲット・データを構築するには、データベース・ユーティリティでデータの静的コピーを作成および適用しながら、変更同期 Extract グループを起動して進行中のデータ変更を抽出します。コピーが終了したら、変更同期 Replicat グループを起動して、コピーの適用中に変更された行を再同期します。これ以降、Extract と Replicat の両方が継続的に実行され、データ同期が維持されます。この方法では、初期ロード用の特別な Extract プロセスまたは Replicat プロセスを使用しません。

データベース・ユーティリティを使用してデータをロードする手順

1. 218 ページの「初期ロードの前提条件」の要件を満たしていることを確認します。
2. ソース・システムとターゲット・システムで、GGSCI を実行して Manager プロセスを起動します。

   ```
   START MANAGER
   ```

   注意 Windows クラスタでは、クラスタ・アドミニストレータで Manager リソースを起動します。

3. ソース・システムで、変更の抽出を開始します。

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

   条件： <group name> は、Extract グループの名前です。

4. (Oracle で順序をレプリケートする場合)update.Sequence に対する EXECUTE 権限を持つユーザーとして DBLOGIN コマンドを発行します。

   ```
   GGSCI> DBLOGIN USERID DBLOGINuser, PASSWORD password
   ```

5. (Oracle で順序をレプリケートする場合) 次のコマンドを発行して各ソース順序を更新し、REDO を生成します。この REDO から、Replicat がターゲット上の順序の初期同期を実行します。順序 (所有者ではない) の名前は、任意の文字またはすべて文字をアスタリスク・ワイルドカードで表すことができます。

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

6. ソース・システムで、コピーの作成を開始します。
7. コピーが終了するまで待機し、完了した時刻を記録します。
8. Replicat のパラメータ・ファイルを表示して、HANDLECOLLISIONS パラメータがリストされていることを確認します。ない場合は、EDIT PARAMS コマンドを使用してパラメータを追加します。

   ```
   VIEW PARAMS <group name>
   EDIT PARAMS <group name>
   ```

条件： <group name> は、Replicat グループの名前です。

9. ターゲット・システムで、変更のレプリケーションを開始します。

```
START REPLICAT <group name>
```

条件： <group name> は、Replicat グループの名前です。

10. ターゲット・システムで、次のコマンドを発行して変更のレプリケーションのステータスを確認します。

```
INFO REPLICAT <group name>
```

11. 初期ロード中に生成されたすべての変更データが変更のレプリケーションにより適用されたことを確認するまで、INFO REPLICAT コマンドを発行し続けます。前に記録した完了時刻を参照してください。たとえば、コピーが 12:05 に停止した場合、変更のレプリケーションによってその時刻までデータが適用されていることを確認します。

12. ターゲット・システムで、次のコマンドを発行して HANDLECOLLISIONS パラメータをオフにし、初期ロードのエラー処理を無効化します。

```
SEND REPLICAT <Replicat group name>, NOHANDLECOLLISIONS
```

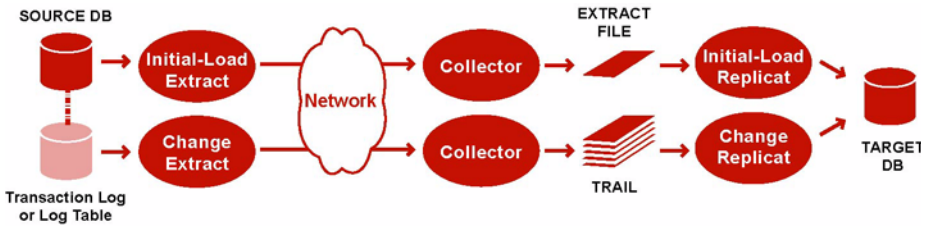
13. ターゲット・システムで、Replicat のパラメータ・ファイルを編集して HANDLECOLLISIONS パラメータを削除します。これによって、次回 Replicat が起動したときに HANDLECOLLISIONS が再度有効化されることを防止します。

```
EDIT PARAMS <Replicat group name>
```

14. パラメータ・ファイルを保存して閉じます。

これ以降、Oracle GoldenGate によって継続的にデータ変更が同期されます。

ファイルから Replicat へのデータのロード



Replicat を使用してターゲット・データを構築するには、初期ロード Extract を使用してソース表からソース・レコードを抽出し、それらのレコードを正規形式で抽出ファイルに書き込みます。初期ロード Replicat は、データベース・インタフェースを使用してこのファイルからデータをロードします。ロード中、変更同期グループは、増分変更を抽出してレプリケートします。その後、これらの変更は、ロードの結果に応じて調整されます。

ロード中、レコードは、1 つずつターゲット・データベースに適用されるため、この方法は他の初期ロード方法と比較して非常に低速です。この方法では、ソース・システムとターゲット・システムのいずれかでデータ変換を実行できます。

ファイルから Replicat にデータをロードする手順

1. 218 ページの「初期ロードの前提条件」の要件を満たしていることを確認します。
2. ソース・システムとターゲット・システムで、GGSCI を実行して Manager を起動します。

   ```
   START MANAGER
   ```

   注意 Windows クラスタでは、クラスタ・アドミニストレータで Manager リソースを起動します。

3. ソース・システムで、次のコマンドを発行して初期ロード Extract のパラメータ・ファイルを作成します。

   ```
   EDIT PARAMS <initial-load Extract name>
   ```

4. 表 29 にリストされている順序でパラメータを入力します。パラメータ文ごとに新規行を開始します。

表 29 ファイルから Replicat にデータをロードするための初期ロード Extract のパラメータ

パラメータ	説明
`SOURCEISTABLE`	Extract をソース表からレコードを直接抽出する初期ロード・プロセスとして指定します。

`[SOURCEDB <dsn>,]`
`[USERID <user id> [, PASSWORD <pw>]]`
◆ SOURCEDB では、データソース名を指定します (接続情報で必要な場合)。Oracle では必要ありません。
◆ USERID では、必要に応じてデータベース資格証明を指定します。Oracle では、次のようなホスト文字列を含めることができます。
`USERID ggs@ora1.ora, PASSWORD ggs123`
NonStop SQL/MX または DB2 では、PASSWORD は必要ありません。
	データベース接続情報を指定します。これらのパラメータでは、オペレーティング・システム・レベルでの認証も可能です。『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』を参照してください。

`RMTHOST <hostname>,` `MGRPORT <portnumber>`	Manager が稼働しているターゲット・システムおよびポートを指定します。

`RMTFILE <path name>,`
`[MAXFILES <number>, MEGABYTES <n>]`
◆ <path name> は、ファイルの相対名または完全修飾名です。
◆ MAXFILES では、必要に応じてエージングされる一連のファイルを作成します。ファイルがオペレーティング・システムのファイル・サイズ制限を超える可能性がある場合に使用します。
◆ MEGABYTES では、各ファイルのサイズを指定します。
	ロード・データを書き込む抽出ファイルを指定します。Oracle GoldenGate では、ロード中にこのファイルが作成されます。チェックポイントは、RMTFILE では保持されません。
	注意 : Solaris システムでは、抽出ファイルが Replicat によって処理される場合、そのサイズは 2GB 以下にする必要があります。MAXFILES オプションおよび MEGABYTES オプションを使用して、サイズを制御します。

表 29　ファイルから Replicat にデータをロードするための初期ロード Extract のパラメータ (続き)

パラメータ	説明
TABLE <owner>.<table>; ◆ <owner> は、スキーマ名です。 ◆ <table> は、表の名前またはワイルドカードで定義された表のグループの名前です。ワイルドカードの指定から表を除外するには、TABLEEXCLUDE パラメータを使用します。	初期データ抽出用の 1 つ以上のソース表を指定します。

5. 『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』にリストされている Extract の適切なオプション・パラメータを入力します。
6. パラメータ・ファイルを保存して閉じます。
7. ターゲット・システムで、次のコマンドを発行して初期ロード Replicat のパラメータ・ファイルを作成します。

   ```
   EDIT PARAMS <initial-load Replicat name>
   ```

8. 表 30 にリストされている順序でパラメータを入力します。パラメータ文ごとに新規行を開始します。

表 30　ファイルから Replicat にデータをロードするための初期ロード Replicat のパラメータ

パラメータ	説明
SPECIALRUN	初期ロード Replicat をチェックポイントを使用しない 1 回かぎりの実行として実装します。
END RUNTIME	ロードの完了時に終了するように初期ロード Replicat に指示します。
[TARGETDB <dsn>,] [USERID <user id> [, PASSWORD <pw>]] ◆ TARGETDB では、データソース名を指定します (接続情報で必要な場合)。Oracle では必要ありません。 ◆ USERID では、必要に応じてデータベース資格証明を指定します。Oracle では、次のようなホスト文字列を含めることができます。 USERID ggs@ora1.ora, PASSWORD ggs123	データベース接続情報を指定します。これらのパラメータでは、オペレーティング・システム・レベルでの認証も可能です。『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』を参照してください。
EXTFILE <path name> | EXTTRAIL <path name> ◆ <path name> は、ファイルまたは証跡の相対名または完全修飾名です。 ◆ EXTTRAIL は、Extract のパラメータ・ファイルで RMTFILE パラメータの MAXFILES オプションを使用している場合にのみ使用します。	Extract パラメータの RMTFILE で指定されている抽出ファイルを指定します。

表 30　ファイルから Replicat にデータをロードするための初期ロード Replicat のパラメータ (続き)

パラメータ	説明
{SOURCEDEFS <file name>} | ASSUMETARGETDEFS ◆ SOURCEDEFS は、ソース表とターゲット表に異なる定義が含まれる場合に使用します。DEFGEN によって生成されたソース定義ファイルの相対名または完全修飾名を指定します。 ◆ ASSUMETARGETDEFS は、ソース表とターゲット表に同じ定義が含まれる場合に使用します。	データ定義の解釈方法を指定します。 データ定義ファイルの詳細は、第 11 章を参照してください。
MAP <owner>.<table>, TARGET <owner>.<table>; ◆ <owner> は、スキーマ名です。 ◆ <table> は、表の名前または複数の表を示すワイルドカード定義です。ワイルドカードの指定から表を除外するには、MAPEXCLUDE パラメータを使用します。	ソースとターゲットの 1 つ以上の表どうしの関係を指定します。

9. 『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』にリストされている Replicat の適切なオプション・パラメータを入力します。
10. ファイルを保存して閉じます。
11. ソース・システムで、変更の抽出を開始します。

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

12. (Oracle で順序をレプリケートする場合)update.Sequence に対する EXECUTE 権限を持つユーザーとして DBLOGIN コマンドを発行します。

    ```
    GGSCI> DBLOGIN USERID DBLOGINuser, PASSWORD password
    ```

13. (Oracle で順序をレプリケートする場合) 次のコマンドを発行して各ソース順序を更新し、REDO を生成します。この REDO から、Replicat がターゲット上の順序の初期同期を実行します。順序 (所有者ではない) の名前は、任意の文字またはすべて文字をアスタリスク・ワイルドカードで表すことができます。

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

14. ソース・システムの Oracle GoldenGate がインストールされているディレクトリから、初期ロード Extract を起動します。

    UNIX および Linux:

    ```
    $ /<GGS directory>/extract paramfile dirprm/<initial-load Extract name>.prm
    reportfile <path name>
    ```

    Windows:

    ```
    C:\> <GGS directory>\extract paramfile dirprm\<initial-load Extract name>.prm
    reportfile <path name>
    ```

条件 : <initial-load Extract name> はパラメータ・ファイルの作成時に使用した初期ロード Extract の名前であり、<path name> は Extract レポート・ファイルの相対名または完全修飾名です。

15. オペレーティング・システムの標準的なファイル表示方法を使用して Extract レポート・ファイルを表示し、初期抽出の進行状況および結果を確認します。
16. 初期抽出が終了するまで待機します。
17. ターゲット・システムで、初期ロード Replicat を起動します。

    UNIX および Linux:

    ```
    $ /<GGS directory>/replicat paramfile dirprm/<initial-load Replicat name>.prm
    reportfile <path name>
    ```

    Windows:

    ```
    C:\> <GGS directory>\replicat paramfile dirprm\<initial-load Replicat name>.prm
    reportfile <path name>
    ```

    条件 : <initial-load Replicat name> はパラメータ・ファイルの作成時に使用した初期ロード Replicat の名前であり、<path name> は Replicat レポート・ファイルの相対名または完全修飾名です。
18. 初期ロード Replicat の実行が終了したら、オペレーティング・システムの標準的なファイル表示方法を使用して Replicat レポート・ファイルを表示し、結果を確認します。
19. ターゲット・システムで、変更のレプリケーションを開始します。

    ```
    START REPLICAT <Replicat group name>
    ```

20. ターゲット・システムで、次のコマンドを発行して変更のレプリケーションのステータスを確認します。

    ```
    INFO REPLICAT <Replicat group name>
    ```

21. 初期ロード中に生成されたすべての変更データが Replicat により適用されたことを確認するまで、INFO REPLICAT コマンドを発行し続けます。たとえば、初期ロード Extract が 12:05 に停止した場合、Replicat によってその時刻までデータが適用されていることを確認します。
22. ターゲット・システムで、次のコマンドを発行して HANDLECOLLISIONS パラメータをオフにし、初期ロードのエラー処理を無効化します。

    ```
    SEND REPLICAT <Replicat group name>, NOHANDLECOLLISIONS
    ```

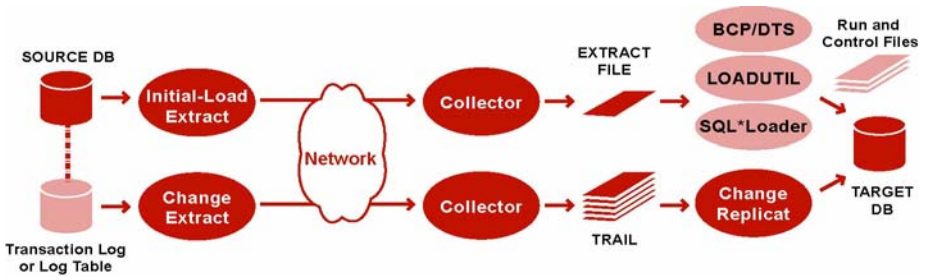
23. ターゲット・システムで、Replicat のパラメータ・ファイルを編集して HANDLECOLLISIONS パラメータを削除します。これによって、次回 Replicat が起動したときに HANDLECOLLISIONS が再度有効化されることを防止します。

    ```
    EDIT PARAMS <Replicat group name>
    ```

24. パラメータ・ファイルを保存して閉じます。

これ以降、Oracle GoldenGate によって継続的にデータ変更が同期されます。

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



データベースのバルク・ロード・ユーティリティを使用するには、初期ロード Extract を使用してソース表からソース・レコードを抽出し、それらのレコードを外部 ASCII 形式で抽出ファイルに書き込みます。ファイルを読み取ることができるのは、Oracle SQL*Loader、Microsoft 社の BCP、DTS または SQL Server Integration Services(SSIS) ユーティリティ、あるいは IBM 社のロード・ユーティリティ (LOADUTIL) です。ロード中、変更同期グループは、増分変更を抽出してレプリケートします。その後、これらの変更は、ロードの結果に応じて調整されます。ロード手順の一環として、Oracle GoldenGate では、初期ロード Replicat の使用により、データベース・ユーティリティに必要な実行ファイルと制御ファイルが作成されます。

制御ファイルは動的に生成され、変換ルールで事前構成できないため、すべてのデータ変換はソース・システムで初期ロード Extract によって実行される必要があります。

ファイルからデータベース・ユーティリティにデータをロードする手順

1. 218 ページの「初期ロードの前提条件」を満たしていることを確認します。
2. ソース・システムとターゲット・システムで、GGSCI を実行して Manager を起動します。

   ```
   START MANAGER
   ```

3. ソース・システムで、次のコマンドを発行して初期ロード Extract のパラメータ・ファイルを作成します。

   ```
   EDIT PARAMS <initial-load Extract name>
   ```

4. 表 31 にリストされている順序でパラメータを入力します。パラメータ文ごとに新規行を開始します。

表 31　ファイルからデータベース・ユーティリティにロードするための初期ロード Extract のパラメータ

パラメータ	説明
SOURCEISTABLE	Extract をソース表からレコードを直接抽出する初期ロード・プロセスとして指定します。

表 31　ファイルからデータベース・ユーティリティにロードするための初期ロード Extract のパラメータ (続き)

パラメータ	説明

`[SOURCEDB <dsn>,]`
`[USERID <user id> [, PASSWORD <pw>]]`
◆ SOURCEDB では、データソース名を指定します (接続情報で必要な場合)。Oracle では必要ありません。
◆ USERID では、必要に応じてデータベース資格証明を指定します。Oracle では、次のようなホスト文字列を含めることができます。
`USERID ggs@ora1.ora, PASSWORD ggs123`
NonStop SQL/MX または DB2 では、PASSWORD は必要ありません。
	データベース接続情報を指定します。これらのパラメータでは、オペレーティング・システム・レベルでの認証も可能です。『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』を参照してください。

`RMTHOST <hostname>,`
`MGRPORT <portnumber>`
`[, PARAMS - E -d <defs file>]`
◆ -E は、ASCII を EBCDIC に変換します。
◆ -d <defs file> では、ソース定義ファイルを指定します。
	Manager が稼働しているターゲット・システムおよびポートを指定します。
	PARAMS 句は、IBM 社のロード・ユーティリティでロードする場合に必要です (Oracle GoldenGate でソース定義ファイルを参照する必要があるため)。

`RMTFILE <path name>,`
`[MAXFILES <number>, MEGABYTES <n>]`
◆ <path name> は、ファイルの相対名または完全修飾名です。
◆ MAXFILES では、必要に応じてエージングされる一連のファイルを作成します。ファイルがオペレーティング・システムのファイル・サイズ制限を超える可能性がある場合に使用します。
◆ MEGABYTES では、各ファイルのサイズを指定します。
	ロード・データを書き込む抽出ファイルを指定します。Oracle GoldenGate では、ロード中にこのファイルが作成されます。チェックポイントは、RMTFILE では保持されません。

`FORMATASCII, {BCP | SQLLOADER}` ◆ BCP は、BCP、DTS または SSIS に使用します。 ◆ SQLLOADER は、Oracle SQL*Loader または IBM 社のロード・ユーティリティに使用します。	出力をデフォルトの正規形式ではなく ASCII テキストとしてフォーマットするように指定します。制限およびオプションの詳細は、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』を参照してください。
`TABLE <owner>.<table>;` ◆ <owner> は、スキーマ名です。 ◆ <table> は、表の名前またはワイルドカードで定義された表のグループの名前です。ワイルドカードの指定から表を除外するには、TABLEEXCLUDE パラメータを使用します。	初期データ抽出用の 1 つ以上のソース表を指定します。

5. 『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』にリストされている Extract の適切なオプション・パラメータを入力します。
6. パラメータ・ファイルを保存して閉じます。

7. ターゲット・システムで、次のコマンドを発行して初期ロード Replicat のパラメータ・ファイルを作成します。

   ```
   EDIT PARAMS <initial-load Replicat name>
   ```

8. 表 32 にリストされている順序でパラメータを入力します。パラメータ文ごとに新規行を開始します。

表 32　ファイルからデータベース・ユーティリティにロードするための初期ロード Replicat のパラメータ

パラメータ	説明
`GENLOADFILES <template file>`	データベース・ユーティリティの実行ファイルと制御ファイルを生成します。このパラメータの使用の詳細は、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』を参照してください。

`[TARGETDB <dsn>,]`
`[USERID <user id> [, PASSWORD <pw>]]`
◆ TARGETDB では、データソース名を指定します (接続情報で必要な場合)。Oracle では必要ありません。
◆ USERID では、必要に応じてデータベース資格証明を指定します。Oracle では、次のようなホスト文字列を含めることができます。
`USERID ggs@ora1.ora, PASSWORD ggs123`
	データベース接続情報を指定します。これらのパラメータでは、オペレーティング・システム・レベルでの認証も可能です。『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』を参照してください。

`EXTFILE <path name> |`
`EXTTRAIL <path name>`
◆ <path name> は、ファイルの相対名または完全修飾名です。
◆ EXTTRAIL は、Extract のパラメータ・ファイルで RMTFILE パラメータの MAXFILES オプションを使用している場合にのみ使用します。
	Extract パラメータの RMTFILE で指定されている抽出ファイルを指定します。

`{SOURCEDEFS <path name>} |`
`ASSUMETARGETDEFS`
◆ SOURCEDEFS は、ソース表とターゲット表に異なる定義が含まれる場合に使用します。DEFGEN によって生成されたソース定義ファイルの相対名または完全修飾名を指定します。
◆ ASSUMETARGETDEFS は、ソース表とターゲット表に同じ定義が含まれる場合に使用します。
	データ定義の解釈方法を指定します。データ定義ファイルの詳細は、第 11 章を参照してください。

表 32　ファイルからデータベース・ユーティリティにロードするための初期ロード Replicat のパラメータ (続き)

パラメータ	説明
MAP <owner>.<table>, TARGET <owner>.<table>; ◆ <owner> は、スキーマ名です。 ◆ <table> は、表の名前または複数の表を示すワイルドカード定義です。ワイルドカードの指定から表を除外するには、MAPEXCLUDE パラメータを使用します。	ソースとターゲットの 1 つ以上の表どうしの関係を指定します。

9. 『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』にリストされている Replicat の適切なオプション・パラメータを入力します。
10. パラメータ・ファイルを保存して閉じます。
11. ソース・システムで、変更の抽出を開始します。

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

12. (Oracle で順序をレプリケートする場合)update.Sequence に対する EXECUTE 権限を持つユーザーとして DBLOGIN コマンドを発行します。

    ```
    GGSCI> DBLOGIN USERID DBLOGINuser, PASSWORD password
    ```

13. (Oracle で順序をレプリケートする場合) 次のコマンドを発行して各ソース順序を更新し、REDO を生成します。この REDO から、Replicat がターゲット上の順序の初期同期を実行します。順序 (所有者ではない) の名前は、任意の文字またはすべて文字をアスタリスク・ワイルドカードで表すことができます。

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

14. ソース・システムの Oracle GoldenGate がインストールされているディレクトリから、初期ロード Extract を起動します。

    UNIX および Linux:

    ```
    $ /<GGS directory>/extract paramfile dirprm/<initial-load Extract name>.prm
    reportfile <path name>
    ```

    Windows:

    ```
    C:\> <GGS directory>\extract paramfile dirprm\<initial-load Extract name>.prm
    reportfile <path name>
    ```

    条件 : <initial-load Extract name> はパラメータ・ファイルの作成時に使用した初期ロード Extract の名前であり、<path name> は Extract レポート・ファイルの相対名または完全修飾名です。
15. オペレーティング・システムの標準的なファイル表示方法を使用して Extract レポート・ファイルを表示し、初期抽出の進行状況および結果を確認します。
16. 初期抽出が終了するまで待機します。
17. ターゲット・システムで、初期ロード Replicat を起動します。

UNIX および Linux:

```
$ /<GGS directory>/replicat paramfile dirprm/<initial-load Replicat name>.prm
reportfile <path name>
```

Windows:

```
C:\> <GGS directory>\replicat paramfile dirprm\<initial-load Replicat name>.prm
reportfile <path name>
```

条件 : <initial-load Replicat name> はパラメータ・ファイルの作成時に使用した初期ロード Replicat の名前であり、<path name> は Replicat レポート・ファイルの相対名または完全修飾名です。

18. 初期ロード Replicat の実行が終了したら、オペレーティング・システムの標準的なファイル表示方法を使用して Replicat レポート・ファイルを表示し、結果を確認します。
19. ASCII 形式の抽出ファイルと、初期ロード Replicat によって作成された実行ファイルおよび制御ファイルを使用して、データベース・ユーティリティでデータをロードします。
20. ターゲット表へのロードが完了するまで待機します。
21. ターゲット・システムで、変更のレプリケーションを開始します。

    ```
    START REPLICAT <Replicat group name>
    ```

22. ターゲット・システムで、次のコマンドを発行して変更のレプリケーションのステータスを確認します。

    ```
    INFO REPLICAT <group name>
    ```

23. 初期ロード中に生成されたすべての変更データが Replicat により適用されたことを確認するまで、INFO REPLICAT コマンドを発行し続けます。たとえば、初期ロード Extract が 12:05 に停止した場合、Replicat によってその時刻までデータが適用されていることを確認します。
24. ターゲット・システムで、次のコマンドを発行して HANDLECOLLISIONS パラメータをオフにし、初期ロードのエラー処理を無効化します。

    ```
    SEND REPLICAT <Replicat group name>, NOHANDLECOLLISIONS
    ```

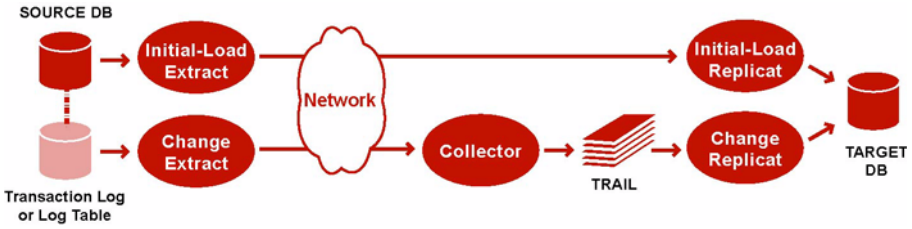
25. ターゲット・システムで、Replicat のパラメータ・ファイルを編集して HANDLECOLLISIONS パラメータを削除します。これによって、次回 Replicat が起動したときに HANDLECOLLISIONS が再度有効化されることを防止します。

    ```
    EDIT PARAMS <Replicat group name>
    ```

26. パラメータ・ファイルを保存して閉じます。

これ以降、Oracle GoldenGate によって継続的にデータ変更が同期されます。

Oracle GoldenGate ダイレクト・ロードを使用したデータのロード



Oracle GoldenGate ダイレクト・ロードを使用するには、Oracle GoldenGate の初期ロード Extract を実行してソース・レコードを抽出し、それらのレコードを初期ロード Replicat タスクに直接送信します。タスクは、Manager プロセスによって動的に起動されるため、Collector プロセスやファイルを使用する必要はありません。初期ロード Replicat タスクによって、ターゲット・データベースに対するロードがサイズの大きいブロック単位で行われます。変換およびマッピングは、Extract または Replicat、あるいはその両方で実行できます。ロード中、変更同期グループは、増分変更を抽出してレプリケートします。その後、これらの変更は、ロードの結果に応じて調整されます。

注意 この方法では、LOB データまたは LONG データの抽出はサポートされません。別の方法として、221 ページの「ファイルから Replicat へのデータのロード」または 226 ページの「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」を参照してください。

Replicat で使用するポートを制御するには、Manager パラメータ・ファイルで DYNAMICPORTLIST パラメータを指定します。Replicat などのプロセスが起動されると、Manager は、DYNAMICPORTLIST で定義されているポートを最初に検索します。ポートがリストされていない場合、Manager は、使用可能なポートが見つかるまでその独自のポート番号を増分して、ポート番号を選択します。

Oracle GoldenGate のダイレクト・ロードでは、LOB、LONG、ユーザー定義型 (UDT)、その他の大規模データ型 (サイズが 4KB 超) を含む列がある表はサポートされません。

Oracle GoldenGate ダイレクト・ロードを使用してデータをロードする手順

1. 218 ページの「初期ロードの前提条件」を満たしていることを確認します。
2. ソース・システムとターゲット・システムで、GGSCI を実行して Manager を起動します。

```
START MANAGER
```

注意 Windows クラスタでは、クラスタ・アドミニストレータで Manager リソースを起動します。

3. ソースで、次のコマンドを発行して初期ロード Extract を作成します。

```
ADD EXTRACT <initial-load Extract name>, SOURCEISTABLE
```

条件 :

- <initial-load Extract name> は、初期ロード Extract の名前 (最大 8 文字) です。
- SOURCEISTABLE では、ソース表から完全なレコードを直接読み取る初期ロード・プロセスとして Extract を指定します。その他の ADD EXTRACT サービス・オプションまたはデータソース引数を使用しないでください。

4. ソース・システムで、次のコマンドを発行して初期ロード Extract のパラメータ・ファイルを作成します。

   ```
   EDIT PARAMS <initial-load Extract name>
   ```

5. 表 33 にリストされている順序でパラメータを入力します。パラメータ文ごとに新規行を開始します。

表 33　Oracle GoldenGate ダイレクト・ロードのための初期ロード Extract のパラメータ

パラメータ	説明
`EXTRACT <initial-load Extract name>`	ステップ 3 で作成した初期ロード Extract を指定します。

`[SOURCEDB <dsn>,]`
`[USERID <user id> [, PASSWORD <pw>]]`
◆ SOURCEDB では、データソース名を指定します (接続情報で必要な場合)。Oracle では必要ありません。
◆ USERID では、必要に応じてデータベース資格証明を指定します。Oracle では、次のようなホスト文字列を含めることができます。
`USERID ggs@ora1.ora, PASSWORD ggs123`
NonStop SQL/MX または DB2 では、PASSWORD は必要ありません。
	データベース接続情報を指定します。これらのパラメータでは、オペレーティング・システム・レベルでの認証も可能です。『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』を参照してください。

`RMTHOST <hostname>,` `MGRPORT <portnumber>`	Manager が稼働しているターゲット・システムおよびポートを指定します。

`RMTTASK replicat,`
`GROUP <initial-load Replicat name>`
◆ <initial-load Replicat name> は、初期ロード Replicat グループの名前です。
	ターゲット・システムの Manager に、初期ロード Replicat を 1 回かぎりのタスクとして動的に起動するように指示します。

`TABLE <owner>.<table>;` ◆ <owner> は、スキーマ名です。 ◆ <table> は、表の名前またはワイルドカードで定義された表のグループの名前です。ワイルドカードの指定から表を除外するには、TABLEEXCLUDE パラメータを使用します。	初期データ抽出用の 1 つ以上のソース表を指定します。

6. 『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』にリストされている Extract の適切なオプション・パラメータを入力します。
7. ファイルを保存して閉じます。
8. ターゲット・システムで、次のコマンドを発行して初期ロード Replicat タスクを作成します。

   ```
   ADD REPLICAT <initial-load Replicat name>, SPECIALRUN
   ```

条件 :

- <initial-load Replicat name> は、初期ロード Replicat タスクの名前です。
- SPECIALRUN では、初期ロード Replicat を継続的なプロセスではなく 1 回かぎりの実行として指定します。

9. ターゲット・システムで、次のコマンドを発行して初期ロード Replicat のパラメータ・ファイルを作成します。

```
EDIT PARAMS <initial-load Replicat name>
```

10. 表 34 にリストされている順序でパラメータを入力します。パラメータ文ごとに新規行を開始します。

表 34　Oracle GoldenGate ダイレクト・ロードのための初期ロード Replicat のパラメータ

パラメータ	説明
`REPLICAT <initial-load Replicat name>`	Manager で起動する初期ロード Replicat タスクを指定します。ステップ 8 で初期ロード Replicat を作成したときに指定した名前を使用します。

`[TARGETDB <dsn>,]`
`[USERID <user id> [, PASSWORD <pw>]]`
◆ TARGETDB では、データ ソース名を指定します (接続情報で必要な場合)。Oracle では必要ありません。
◆ USERID では、必要に応じてデータベース資格証明を指定します。Oracle では、次のようなホスト文字列を含めることができます。
`USERID ggs@ora1.ora, PASSWORD ggs123`
	データベース接続情報を指定します。これらのパラメータでは、オペレーティング・システム・レベルでの認証も可能です。『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』を参照してください。

`{SOURCEDEFS <full_pathname>} |`
`ASSUMETARGETDEFS`
◆ SOURCEDEFS は、ソース表とターゲット表に異なる定義が含まれる場合に使用します。DEFGEN によって生成されたソース定義ファイルを指定します。
◆ ASSUMETARGETDEFS は、ソース表とターゲット表に同じ定義が含まれる場合に使用します。
	データ定義の解釈方法を指定します。データ定義ファイルの詳細は、第 11 章を参照してください。

`MAP <owner>.<table>,`
`TARGET <owner>.<table>;`
◆ <owner> は、スキーマ名です。
◆ <table> は、表の名前または複数の表を示すワイルドカード定義です。ワイルドカードの指定から表を除外するには、MAPEXCLUDE パラメータを使用します。
	ソースとターゲットの 1 つ以上の表どうしの関係を指定します。

11. 『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』にリストされている Replicat の適切なオプション・パラメータを入力します。
12. パラメータ・ファイルを保存して閉じます。
13. ソース・システムで、変更の抽出を開始します。

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

14. (Oracle で順序をレプリケートする場合)update.Sequence に対する EXECUTE 権限を持つユーザーとして DBLOGIN コマンドを発行します。

```
GGSCI> DBLOGIN USERID DBLOGINuser, PASSWORD password
```

15. (Oracle で順序をレプリケートする場合) 次のコマンドを発行して各ソース順序を更新し、REDO を生成します。この REDO から、Replicat がターゲット上の順序の初期同期を実行します。順序 (所有者ではない) の名前は、任意の文字またはすべて文字をアスタリスク・ワイルドカードで表すことができます。

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

16. ソース・システムで、初期ロード Extract を起動します。

```
START EXTRACT <initial-load Extract name>
```

> **注意** 初期ロード Replicat は起動しないでください。このプロセスは、Manager プロセスによって自動的に起動され、ロードの完了時に終了されます。

17. ターゲット・システムで、次のコマンドを発行してロードが終了したかどうかを確認します。ロードが終了するまで待機してから、次の手順に進みます。

```
VIEW REPORT <initial-load Extract name>
```

18. ターゲット・システムで、変更のレプリケーションを開始します。

```
START REPLICAT <Replicat group name>
```

19. ターゲット・システムで、次のコマンドを発行して変更のレプリケーションのステータスを確認します。

```
INFO REPLICAT <Replicat group name>
```

20. 初期ロード中に生成されたすべての変更データが Replicat により適用されたことを確認するまで、INFO REPLICAT コマンドを発行し続けます。たとえば、初期ロード Extract が 12:05 に停止した場合、Replicat によってその時刻までデータが適用されていることを確認します。
21. ターゲット・システムで、次のコマンドを発行して HANDLECOLLISIONS パラメータをオフにし、初期ロードのエラー処理を無効化します。

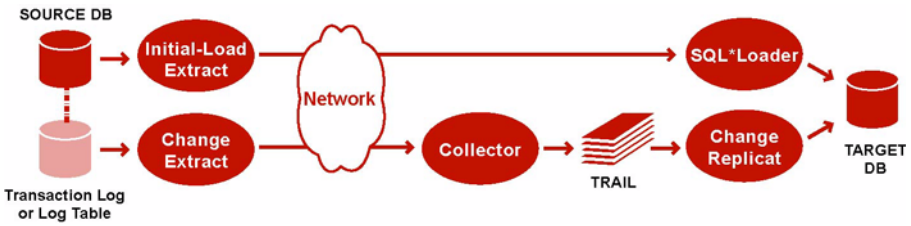
```
SEND REPLICAT <Replicat group name>, NOHANDLECOLLISIONS
```

22. ターゲット・システムで、Replicat のパラメータ・ファイルを編集して HANDLECOLLISIONS パラメータを削除します。これによって、次回 Replicat が起動したときに HANDLECOLLISIONS が再度有効化されることを防止します。

```
EDIT PARAMS <Replicat group name>
```

23. パラメータ・ファイルを保存して閉じます。これ以降、Oracle GoldenGate によって継続的にデータ変更が同期されます。

ダイレクト・バルク・ロードを使用した SQL*Loader へのデータのロード



Oracle SQL*Loader ユーティリティを使用してターゲット・データを構築するには、Oracle GoldenGate の初期ロード Extract を実行してソース・レコードを抽出し、それらのレコードを初期ロード Replicat タスクに直接送信します。タスクは、Manager プロセスによって動的に起動されるプロセスであり、Collector プロセスやファイルを使用する必要はありません。初期ロード Replicat タスクは、SQL*Loader の API とのインタフェースになり、ダイレクト・パス・バルク・ロードとしてデータをロードします。データのマッピングおよび変換は、初期ロード Extract または初期ロード Replicat、あるいはその両方によって実行できます。ロード中、変更同期グループは、増分変更を抽出してレプリケートします。その後、これらの変更は、ロードの結果に応じて調整されます。

Replicat で使用するポートを制御するには、Manager パラメータ・ファイルで DYNAMICPORTLIST パラメータを指定します。Replicat などのプロセスが起動されると、Manager は、DYNAMICPORTLIST で定義されているポートを最初に検索します。ポートがリストされていない場合、Manager は、使用可能なポートが見つかるまでその独自のポート番号を増分して、ポート番号を選択します。

制限事項 :

- この方法は、Oracle SQL*Loader でのみ動作します。他のデータベースでは使用しないでください。
- この方法では、LOB データまたは LONG データの抽出はサポートされません。別の方法として、221 ページの「ファイルから Replicat へのデータのロード」または 226 ページの「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」を参照してください。
- この方法では、サイズにかかわらず、LOB を含むマテリアライズド・ビューがサポートされません。また、データ暗号化もサポートされません。

ダイレクト・バルク・ロードを使用して SQL*Loader にデータをロードする手順

1. 218 ページの「初期ロードの前提条件」の要件を満たしていることを確認します。
2. (Oracle 9i 以上) ターゲットの Oracle データベースの Replicat データベース・ユーザーに LOCK ANY TABLE を付与します。
3. ソース・システムとターゲット・システムで、GGSCI を実行して Manager を起動します。

   ```
   START MANAGER
   ```

4. ソース・システムで、次のコマンドを発行して初期ロード Extract を作成します。

   ```
   ADD EXTRACT <initial-load Extract name>, SOURCEISTABLE
   ```

   条件 :

   - <initial-load Extract name> は、初期ロード Extract の名前 (最大 8 文字) です。
   - SOURCEISTABLE では、ソース表から完全なレコードを直接読み取る初期ロード・プロセスとして Extract を指定します。その他の ADD EXTRACT サービス・オプションまたはデータソース引数を使用しないでください。

5. ソース・システムで、次のコマンドを発行して初期ロード Extract のパラメータ・ファイルを作成します。

   ```
   EDIT PARAMS <initial-load Extract name>
   ```

6. 表 35 にリストされている順序でパラメータを入力します。パラメータ文ごとに新規行を開始します。

表 35　SQL*Loader へのダイレクト・バルク・ロードのための初期ロード Extract のパラメータ

パラメータ	説明
`EXTRACT <initial-load Extract name>`	ステップ 4 で作成した初期ロード Extract を指定します。

`[SOURCEDB <dsn>,]`
`[USERID <user id> [, PASSWORD <pw>]]`
◆ SOURCEDB では、データソース名を指定します (接続情報で必要な場合)。Oracle では必要ありません。
◆ USERID では、必要に応じてデータベース資格証明を指定します。Oracle では、次のようなホスト文字列を含めることができます。
`USERID ggs@ora1.ora, PASSWORD ggs123`
NonStop SQL/MX または DB2 では、PASSWORD は必要ありません。
	データベース接続情報を指定します。これらのパラメータでは、オペレーティング・システム・レベルでの認証も可能です。『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』を参照してください。

`RMTHOST <hostname>,` `MGRPORT <portnumber>`	Manager が稼働しているターゲット・システムおよびポートを指定します。

`RMTTASK replicat,`
`GROUP <initial-load Replicat name>`
◆ <initial-load Replicat name> は、初期ロード Replicat グループの名前です。
	ターゲット・システムの Manager に、初期ロード Replicat を 1 回かぎりのタスクとして動的に起動するように指示します。

`TABLE <owner>.<table>;` ◆ <owner> は、スキーマ名です。 ◆ <table> は、表の名前またはワイルドカードで定義された表のグループの名前です。ワイルドカードの指定から表を除外するには、TABLEEXCLUDE パラメータを使用します。	初期データ抽出用の 1 つ以上の表を指定します。

7. 適切なオプション・パラメータを入力します。
8. ファイルを保存して閉じます。

9. ターゲット・システムで、次のコマンドを発行して初期ロード Replicat を作成します。

```
ADD REPLICAT <initial-load Replicat name>, SPECIALRUN
```

条件 :

- <initial-load Replicat name> は、初期ロード Replicat タスクの名前です。
- SPECIALRUN では、初期ロード Replicat を継続的なプロセスではなく 1 回かぎりのタスクとして指定します。

10. ターゲット・システムで、次のコマンドを発行して初期ロード Replicat のパラメータ・ファイルを作成します。

```
EDIT PARAMS <initial-load Replicat name>
```

11. 表 36 にリストされている順序でパラメータを入力します。パラメータ文ごとに新規行を開始します。

表 36　SQL*Loader へのダイレクト・ロードのための初期ロード Replicat のパラメータ

パラメータ	説明
`REPLICAT <initial-load Replicat name>`	Manager で起動する初期ロード Replicat タスクを指定します。ステップ 9 で初期ロード Replicat を作成したときに指定した名前を使用します。
`USERID <user>, PASSWORD <password>`	ターゲットの Oracle データベースに接続するために初期ロード Replicat で使用するユーザー ID およびパスワードを指定します。次のようなホスト文字列を含めることができます。 `USERID ggs@ora1.ora, PASSWORD ggs123` このパラメータでは、オペレーティング・システム・レベルでの認証も可能です。『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』を参照してください。
`BULKLOAD`	Replicat に、Oracle SQL*Loader インタフェースと直接通信するように指定します。
`{SOURCEDEFS <full_pathname>} | ASSUMETARGETDEFS` ◆ SOURCEDEFS は、ソース表とターゲット表に異なる定義が含まれる場合に使用します。DEFGEN によって生成されたソース定義ファイルを指定します。 ◆ ASSUMETARGETDEFS は、ソース表とターゲット表に同じ定義が含まれる場合に使用します。	データ定義の解釈方法を指定します。データ定義ファイルの詳細は、第 11 章を参照してください。

表 36　SQL*Loader へのダイレクト・ロードのための初期ロード Replicat のパラメータ (続き)

パラメータ	説明
MAP <owner>.<table>, TARGET <owner>.<table>; ◆ <owner> は、スキーマ名です。 ◆ <table> は、表の名前または複数の表を示すワイルドカード定義です。ワイルドカードの指定から表を除外するには、MAPEXCLUDE パラメータを使用します。	ソースとターゲットの 1 つ以上の表どうしの関係を指定します。

12. 『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』にリストされている Replicat の適切なオプション・パラメータを入力します。
13. パラメータ・ファイルを保存して閉じます。
14. ソース・システムで、変更の抽出を開始します。

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

15. (Oracle で順序をレプリケートする場合)update.Sequence に対する EXECUTE 権限を持つユーザーとして DBLOGIN コマンドを発行します。

```
GGSCI> DBLOGIN USERID DBLOGINuser, PASSWORD password
```

16. (Oracle で順序をレプリケートする場合) 次のコマンドを発行して各ソース順序を更新し、REDO を生成します。この REDO から、Replicat がターゲット上の順序の初期同期を実行します。順序 (所有者ではない) の名前は、任意の文字またはすべて文字をアスタリスク・ワイルドカードで表すことができます。

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

17. ソース・システムで、初期ロード Extract を起動します。

```
START EXTRACT <initial-load Extract name>
```

警告　初期ロード Replicat は起動しないでください。このプロセスは、Manager プロセスによって自動的に起動され、ロードの完了時に終了されます。

18. ターゲット・システムで、次のコマンドを発行してロードが終了したことを確認します。ロードが終了するまで待機してから、次の手順に進みます。

```
VIEW REPORT <initial-load Extract name>
```

19. ターゲット・システムで、変更のレプリケーションを開始します。

```
START REPLICAT <Replicat group name>
```

20. ターゲット・システムで、次のコマンドを発行して変更のレプリケーションのステータスを確認します。

```
INFO REPLICAT <Replicat group name>
```

21. 初期ロード中に生成されたすべての変更データが Replicat により適用されたことを確認するまで、INFO REPLICAT コマンドを発行し続けます。たとえば、初期ロード Extract が 12:05 に停止した場合、Replicat によってその時刻までデータが適用されていることを確認します。
22. ターゲット・システムで、次のコマンドを発行して HANDLECOLLISIONS パラメータをオフにし、初期ロードのエラー処理を無効化します。

    ```
    SEND REPLICAT <Replicat group name>, NOHANDLECOLLISIONS
    ```

23. ターゲット・システムで、Replicat のパラメータ・ファイルを編集して HANDLECOLLISIONS パラメータを削除します。これによって、次回 Replicat が起動したときに HANDLECOLLISIONS が再度有効化されることを防止します。

    ```
    EDIT PARAMS <Replicat group name>
    ```

24. パラメータ・ファイルを保存して閉じます。

これ以降、Oracle GoldenGate によって継続的にデータ変更が同期されます。

Teradata ロード・ユーティリティを使用したデータのロード

2 つの Teradata データベースを同期する場合に推奨される方法は、任意の Teradata データ・ロード・ユーティリティを使用することです。推奨ユーティリティは、MultiLoad です。

この手順では、Extract および Replicat の変更同期グループが使用できる状態で、Teradata レプリケーション用に適切に構成されている必要があります。詳細は、第 12 章を参照してください。

複数の Extract グループおよび Replicat グループを使用する場合、すべてのグループで必要に応じて各手順を実行してください。

Teradata ロード・ユーティリティを使用してデータをロードする手順

1. ユーティリティで必要とされるスクリプトを作成します。
2. プライマリ Extract グループを起動します。

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

3. データ・ポンプを起動します (使用する場合)。

    ```
    START EXTRACT <data pump group name>
    ```

4. Replicat のパラメータ・ファイルを編集します。

    ```
    EDIT PARAMS <Replicat group name>
    ```

5. Replicat のパラメータ・ファイルに次のパラメータを追加します。

    ```
    END RUNTIME
    HANDLECOLLISIONS
    ```

    - END RUNTIME では、Replicat に対し、Replicat の起動時以降のタイムスタンプを持つ Oracle GoldenGate の証跡レコードを読み取った時点で正常に終了するように指示します。
    - HANDLECOLLISIONS では、Replicat に対し、トランザクション変更とコピー結果の衝突によって発生するエラーを解決する手段として、重複レコードを上書きし、欠落レコードを無視するように指示します。

6. Replicat のパラメータ・ファイルを保存して閉じます。
7. ロード・ユーティリティを起動します。
8. ターゲットでロードが完了したら、Replicat プロセスを起動します。
9. 各 Replicat プロセスが停止したら、パラメータ・ファイルから HANDLECOLLISIONS パラメータと END RUNTIME パラメータを削除します。
10. Replicat プロセスを再起動します。これで 2 つのデータベースは同期し、Oracle GoldenGate のレプリケーションを通じて最新状態に維持されます。

第 17 章

データのマッピングおよび操作

データのマッピングおよび操作の概要

異なるソース表とターゲット表のデータを統合する場合、次の方法を使用できます。

- レコードおよび列の選択
- 操作の選択および変換
- 異なる列のマッピング
- トランザクション履歴の使用
- データのテストおよび変換
- トークンの使用

Oracle GoldenGate で実行されるデータの選択、マッピングおよび操作は、すべて TABLE パラメータおよび MAP パラメータのオプションを使用して行われます。TABLE は Extract のパラメータ・ファイルで使用し、MAP は Replicat のパラメータ・ファイルで使用します。

異なるデータ構造を持つ表どうしのマッピングおよび変換には、ソース定義ファイルまたはターゲット定義ファイル、あるいは (一定の場合) その両方が必要です。ソース定義ファイルまたはターゲット定義ファイルの作成方法の詳細は、115 ページの第 11 章を参照してください。

サポートの制限

- Oracle GoldenGate の一部の機能および動作では、データのフィルタリングおよび操作がサポートされません。該当する場合、この制限についての記載があります。
- ラージ・オブジェクトのサイズが 4K を超えると、Oracle GoldenGate では、そのデータが Oracle GoldenGate 証跡内のセグメントに格納されます。最初の 4K はベース・セグメントに格納され、残りは一連の 2K セグメントに格納されます。Oracle GoldenGate では、このサイズのラージ・オブジェクトのフィルタリング、列マッピングまたは操作はサポートされません。Oracle GoldenGate の機能をすべて使用できるのは、4K 以下のオブジェクトです。

データのマッピングおよび変換の実行場所の決定

列のマッピングおよび変換は、ソース・システム、ターゲット・システムまたは中間システムで実行できます。ただし、Windows または UNIX システムから NonStop ターゲットに対してレプリケーションを行う場合は例外で、これらの機能は常にソースで実行する必要があります (NonStop の Replicat では、2 つの部分からなる表名およびデータ型を、NonStop プラットフォームで使用される 3 つの部分からなる名前に変換できないため)。Extract が NonStop の名前およびターゲットのデータ型を使用して証跡データをフォーマットできるように、マッピングと変換はソースの Windows または UNIX シス

テムで実行する必要があります。

ソース・システムで追加のオーバーヘッドが発生するのを防止するため、ほとんどの Oracle GoldenGate ユーザーはターゲット・システムまたは中間システムでマッピングや変換を実行するように選択します。ただし、複数のソースに対して 1 つのターゲットがある場合、状況によってはソースでマッピングと変換を実行することが推奨されます。アプリケーションでレイアウト変更が発生するたびにターゲットにコピーする必要のある個別のソース定義ファイルをソース・データベースごとに管理するのではなく、ターゲット表から生成された 1 つのターゲット定義ファイルを使用できます。

NonStop システムのデータに含まれる異常値の処理

Windows または UNIX システムと NonStop Server のソースまたはターゲットとの間でデータを移動する場合、NonStop Server 上で稼働している一部のアプリケーションではトランザクション操作が順序を無視して TMF 監査証跡に送信され、その後で Replicat に送信されるため、特定の異常値が発生する可能性があります。このような異常値が原因でレコードの重複エラーまたは欠落エラーが発生し、Replicat が異常終了することがあります (IDX 医療アプリケーションや一部の BASE25 銀行アプリケーションで高い頻度で発生)。これらの異常値を解決するには、Replicat パラメータの FILTERDUPS を使用します。

行の選択

抽出またはレプリケートする行を指定するには、次のパラメータの FILTER 句および WHERE 句を使用します。

Extract	Replicat
TABLE	MAP

WHERE 句では基本的な WHERE 演算子を使用できますが、FILTER 句では、Oracle GoldenGate 列変換関数をすべて使用できるため、WHERE 句より多くの機能が提供されます。

FILTER 句を使用した行の選択

FILTER 句を使用して、基本演算子または 1 つ以上の Oracle GoldenGate 列変換関数により、数値に基づいて行を選択します。

注意 文字列に基づいて列をフィルタするには、Oracle GoldenGate 文字列関数のいずれかを使用するか、WHERE 句を使用します。

構文

```
TABLE <table spec>,
, FILTER (
[, ON INSERT | ON UPDATE| ON DELETE]
[, IGNORE INSERT | IGNORE UPDATE | IGNORE DELETE]
, <filter clause>);
```

または

構文

```
MAP <table spec>, TARGET <table spec>,
, FILTER (
[, ON INSERT | ON UPDATE| ON DELETE]
[, IGNORE INSERT | IGNORE UPDATE | IGNORE DELETE]
[, RAISEERROR <error_num>]
, <filter clause>);
```

FILTER 句の有効な要素は、次のとおりです。

- Oracle GoldenGate 列変換関数。これらの関数は、Oracle GoldenGate に組み込まれており、テストの実行、データの操作、値の取得などを行うことができます。Oracle GoldenGate 変換関数の詳細は、256 ページの「データのテストおよび変換」を参照してください。
- 数値
- 数値を含む列
- 数値を戻す関数
- 算術演算子 :

  +(加算)
  -(減算)
  *(乗算)
  /(除算)
  \(剰余)

- 比較演算子 :

  >(より大きい)
  >=(以上)
  <(より小さい)
  <=(以下)
  =(等しい)
  <>(等しくない)

  比較から導出される結果は、0 (ゼロ)(FALSE に相当) または 0 以外 (TRUE に相当) です。

- カッコ (式内の結果のグループ化用)
- 結合演算子 : AND、OR

次の FILTER オプションを使用して、フィルタ句の影響を受ける SQL 操作を指定します。これらのオプションは、任意に結合できます。

ON INSERT | ON UPDATE | ON DELETE

IGNORE INSERT | IGNORE UPDATE | IGNORE DELETE

MAP パラメータの FILTER の RAISEERROR オプションを使用して、フィルタに失敗したときのユーザー定義のエラーを生成します。このオプションは、失敗に応じてイベントを起動する必要がある場合に役立ちます。たとえば、フィルタが競合解決構成の一部である場合、RAISEERROR を使用すれば、REPERROR 句でレスポンスを起動できます (レコードの破棄など)。

FILTER 句内のテキスト文字列は、次の例に示すとおり、二重引用符で囲む必要があります。

例 1 次の例では、@COMPUTE 関数をコールして、価格と数量の積が 10,000 を超えるレコードを抽出します。

```
MAP sales.tcustord, TARGET sales.tord,
FILTER (@COMPUTE (product_price*product_amount) > 10000);
```

例 2 次の例では、@STRFIND 関数を使用して、文字列 "JOE" が含まれているレコードを抽出します。

```
TABLE act.tcustord, FILTER (@STRFIND (name, "joe") > 0);
```

例 3 次の例では、amount 列が 50 を超えているレコードを選択し、更新時および削除時にフィルタを実行します。

```
TABLE act.tcustord, FILTER (ON UPDATE, ON DELETE, amount > 50);
```

例 4 @RANGE 関数を使用して、個別の TABLE 文または MAP 文の複数の FILTER 句間で処理ワークロードを分割できます。たとえば、次の例では、acct ソース表の ID 列に基づいて、2 つの範囲 (2 つの Replicat プロセス間) にレプリケーション・ワークロードを分割します。

(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, 2, 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, 2, ID));
```

例 5 Replicat のパラメータ・ファイルの一部を示した表 37 に示すように、1 つの MAP 文または TABLE 文内に複数の FILTER 句を結合できます。Oracle GoldenGate では、フィルタはいずれか 1 つが失敗するか、すべてが成功するまでリスト順に実行されます。1 つのフィルタが失敗すると、フィルタ全体が失敗します。

表 37 複数の FILTER 文の使用

パラメータ・ファイル	説明
`REPERROR (9999, EXCEPTION)`	*1.* 指定したエラーに対して例外を起動します。
`MAP owner.srctab,` `TARGET owner.targtab,`	*2.* MAP 文を開始します。

`SQLEXEC (ID check, ON UPDATE,`
`QUERY " SELECT COUNT FROM targtab "`
`"WHERE PKCOL = :P1 ",`
`PARAMS (P1 = PKCOL)),`
	3. SQLEXEC オプションを使用して問合せを実行し、更新が発生するたびに count 列の現在の値を取得します。

`FILTER (balance > 15000),`	*4.* FILTER 句を使用して、残高が 15000 を超える行を選択します。
`FILTER (ON UPDATE, BEFORE.COUNT = CHECK.COUNT)`	*5.* 別の FILTER 句を使用して、更新前の count ソース列の値がターゲットの更新適用前のターゲット列の値と一致することを保証します。
`;`	*6.* セミコロンで MAP 文を終了します。

表 37　複数の FILTER 文の使用 (続き)

パラメータ・ファイル	説明

`MAP owner.srctab,`
`TARGET owner.targexc,`
`EXCEPTIONSONLY,`
`COLMAP (USEDEFAULTS,`
`ERRTYPE = "UPDATE FILTER FAILED");`
	7. 例外 MAP 文を指定します。エラー 9999 の REPERROR 句によって、targexc に対する例外マップが実行されます。

WHERE 句を使用した行の選択

WHERE 句で表 38 の任意の要素を使用し、条件文に基づいて行の選択または除外 (あるいはその両方) を行います。各 WHERE 句は、カッコで囲む必要があります。

表 38　使用可能な WHERE 演算子

要素	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"`
組込みの列テスト	`@NULL`、`@PRESENT`、`@ABSENT`(行の列が NULL、存在または欠落)。これらのテストは、Oracle GoldenGate に組み込まれています。246 ページの「FILTER および WHERE を使用して行を選択する場合の考慮事項」を参照してください。
比較演算子	`=, <>, >, <, >=, <=`
結合演算子	`AND, OR`
グループ化用カッコ	複数の要素の論理グループ化用に開きカッコと閉じカッコ ( ) を使用します。

算術演算子および浮動小数点データ型は、WHERE ではサポートされません。より複雑な選択条件を使用するには、FILTER 句を使用するか、ユーザー・イグジット・ルーチンを使用します (278 ページの「ユーザー・イグジットを使用した Oracle GoldenGate 機能の拡張」を参照)。

構文

```
TABLE <table spec>, WHERE (<WHERE clause>);
```

または

```
MAP <table spec>, TARGET <table spec>, WHERE (<WHERE clause>);
```

FILTER および WHERE を使用して行を選択する場合の考慮事項

適切な選択句を作成するために、次の推奨事項が役立ちます。

フィルタでのデータ使用可能性の確保

データベースで*圧縮更新* (トランザクション・ログに*変更された*列の値のみが存在する更新) が使用されている場合、欠落している列が選択基準によって参照されると、エラーが発生する可能性があります。Oracle GoldenGate は、このような行操作を無視し、それらを廃棄ファイルに出力して警告を発行します。

欠落列エラーを回避するには、次のように選択条件を作成します。

- 可能であれば、選択基準として主キー列のみを使用します。
- TABLE パラメータの FETCHCOLS オプションまたは FETCHCOLSEXCEPT オプションを使用して、必要な列値を使用可能にします。これらのオプションでは、値がログに存在しない場合に、データベースに問い合せてそれらの値をフェッチします。FILTER 句または WHERE 句が実行される前に値を取得するため、TABLE 文の FETCHBEFOREFILTER オプションを FILTER 句または WHERE 句の前に指定します。次に例を示します。

  ```
  TABLE demo_src.people, FETCHBEFOREFILTER, FETCHCOLS (age), &
  FILTER (age > 50);
  ```

  値をフェッチするよりも、必要な列に対してサプリメンタル・ロギングを有効化する方がより効率的な場合もあります。

- 最初に列が存在するかどうかをテストし、次に列の値をテストします。列が存在するかどうかをテストするには、次の構文を使用します。

  ```
  <column_name> {= | <>} {@PRESENT | @ABSENT}
  ```

  次の例では、AMOUNT 列が 10,000 を超えている場合にすべてのレコードが戻されます。AMOUNT が存在しない場合、レコードは破棄されません。

  ```
  WHERE (amount = @PRESENT AND amount > 10000)
  ```

列値の比較

比較で使用される要素が確実に一致するように、次を比較します。

- 文字の列とリテラル文字列。
- 数値の列と数値 (符号と小数点を含む)。
- 日時の列とリテラル文字列 (アプリケーションによって取得される列の書式を使用)。

変更前の値の取得

更新操作では、ソース列の*変更前*の値 (更新発生前の値) を確認すると役立つことがあります。変更前の値を使用する理由は、次のとおりです。

- レプリケートされたソース列の変更前の値とターゲット列の現在の値(更新の適用前)を比較することで、操作がターゲット・レコードの適切なバージョンに適用されることが保証されます。値が一致しない場合、ターゲット表が破損しているか、Oracle GoldenGate 以外のユーザーまたはアプリケーションによって変更された可能性があります。更新を適用するかわりに、エラーを生成して操作を例外表に記録するようにこれらの比較を構成することで、問題を検出および解決できます。次に例を示します。

  ```
  TABLE hr.people, FETCHBEFOREFILTER, FETCHCOLS (name), &
  FILTER (ON UPDATES, BEFORE.name = "Doe");
  ```

- デルタ計算で変更前の値を使用できます。たとえば、表に Balance 列が含まれる場合、新しい残高から元の残高を差し引いて特定のトランザクションの正味の結果を計算できます。次に例を示します。

```
MAP owner.src, TARGET owner.targ, &
COLMAP (PK1 = PK1, delta = balance  BEFORE.balance);
```

変更前の値を参照する手順

1. 次の例および前述の例のとおり、選択句で列名の前に BEFORE キーワード、続けてドット (.) を接頭辞として追加します。

```
BEFORE.<column_name>
```

2. GETUPDATEBEFORES パラメータを Extract パラメータ・ファイルで使用して変更前イメージを抽出するか、または Replicat パラメータ・ファイルで使用して変更前イメージをレプリケートします。このパラメータを使用するには、すべての列がトランザクション・ログに存在している必要があります。データベースで圧縮更新を使用している場合、BEFORE 接頭辞を使用すると、列の欠落状態が発生し、列がレコードに存在しない場合と同じように列マップが実行されます。列値の使用可能性を確保するには、246 ページの「フィルタでのデータ使用可能性の確保」を参照してください。

NULL 値のテスト

列が NULL 値であるかどうかを評価するには、次の構文を使用します。

```
<column> {= | <>} @NULL
```

次の例では、列が NULL の場合に TRUE が戻され、他のすべての場合 (列がレコードから欠落している場合を含む) に FALSE が戻されます。

```
WHERE (amount = @NULL)
```

次の例では、列がレコードに存在し、NULL ではない場合にのみ TRUE が戻されます。

```
WHERE (amount = @PRESENT AND amount <> @NULL)
```

列の選択

Oracle GoldenGate によって抽出するソース表の列を制御するには、TABLE パラメータの COLS オプションおよび COLSEXCEPT オプションを使用します。COLS では抽出する列を選択し、COLSEXCEPT では、COLSEXCEPT で指定した列以外のすべての列を選択します。

抽出する列を制限すると、ターゲット表にソース表と同じ列が含まれない場合や、列に機密情報 (個人識別番号その他の固有のビジネス情報など) が含まれる場合に役立ちます。

SQL 操作の選択および変換

レプリケートする SQL 文のタイプの選択

デフォルトでは、Oracle GoldenGate によって挿入操作、更新操作および削除操作が同期されます。Extract または Replicat のパラメータ・ファイルで次のパラメータを使用して、処理する操作の種類 (挿入のみ、または挿入と更新のみなど) を制御できます。

GETINSERTS | IGNOREINSERTS

GETUPDATES | IGNOREUPDATES

GETDELETES | IGNOREDELETES

操作タイプ間の変換

SQL 操作を別のタイプに変換するには、次の方法を使用します。次のパラメータは、すべて Replicat のパラメータ・ファイルで使用します。

- INSERTUPDATES を使用して、ソースの更新操作をターゲット表への挿入操作に変換します。これは、ターゲット表にトランザクション履歴を保持する場合に便利です。トランザクション・ログ・レコードには、変更された値だけでなく、表の列値もすべて含まれている必要があります。一部のデータベースでは、トランザクション・ログに行の値がすべて記録されず、変更された値のみが記録されます。
- INSERTDELETES を使用して、ソースのすべての削除操作をターゲット表への挿入操作に変換します。これは、ソース・データベースに存在していたすべてのレコードの履歴を保持する場合に便利です。
- UPDATEDELETES を使用して、ソースの削除操作をターゲットの更新操作に変換します。

列のマッピング

Oracle GoldenGate では、表レベルおよびグローバル・レベルの列マッピングが提供されます。

表レベルの列マッピングの使用

MAP および TABLE パラメータの COLMAP オプションを使用して、次の処理を実行します。

- 異なる名前を持つターゲット列にソース列を明示的にマップします。
- 明示的な列マッピングが不要な場合、デフォルトの列マッピングを指定します。

COLMAP では、ソース列からターゲット列に対してデータを選択、マップ、変換および移動するための指示を提供します。COLMAP 文内で、Oracle GoldenGate 列変換関数のいずれかを使用して、マップされた列のデータを変換できます。

データ定義の指定

COLMAP を使用する場合、状況によってはデータ定義ファイルを作成する必要があります。この必要性を判断するには、ソースおよびターゲットの列構造が Oracle GoldenGate の定義に従って同一であるかどうかを検討する必要があります。

ソース列とターゲット列の構造が同一であるためには、次の要件を満たす必要があります。

- 列名が同一であること (該当する場合は大 / 小文字の区別も含む)。
- データ型が同一であること。
- 列の長さが同一であること。
- 文字の列の列長さセマンティクス (バイトか文字か) が同じであること。
- 各表の順序が同じであること。

たとえば、ソースの Oracle データベースのセマンティクスがバイトとして構成されているのに対し、ターゲットのセマンティクスが文字として構成されている場合 (またはその逆)、表の構造が同一であってもソース定義ファイルが必要です。また別の例として、次に示すように、名前列の順序以外は同一であるソース表とターゲット表の組合せでも、ソース定義ファイルが必要です。

ソース

```
CREATE TABLE emp
( employee_id    NUMBER(6)
, first_name     VARCHAR2(20)
, last_name      VARCHAR2(25)
, phone_number   VARCHAR2(20)
, hire_date      DATE  DEFAULT SYSDATE
```

ターゲット

```
CREATE TABLE emp
( employee_id    NUMBER(6)
, last_name      VARCHAR2(25)
, first_name     VARCHAR2(20)
, phone_number   VARCHAR2(20)
, hire_date      DATE  DEFAULT SYSDATE
```

構造的に*同一ではない*ソース表およびターゲット表で COLMAP を使用する場合、次の要件を満たす必要があります。

- Oracle GoldenGate 構成および使用中のデータベースに応じて、ソース表またはターゲット表、あるいはその両方のデータ定義を生成します。
- 使用する予定のシステムに定義ファイルを転送します。
- SOURCEDEFS パラメータを使用して、定義ファイルを指定します。

115 ページの「データ定義ファイルの作成」を参照してください。

構造的に*同一*のソース表およびターゲット表で COLMAP を使用し、変換などの他の関数でのみ COLMAP を使用する場合、ソース定義ファイルは必要ありません。定義ファイルを使用しない場合、かわりに ASSUMETARGETDEFS パラメータを使用する必要があります。『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』を参照してください。

COLMAP の使用可能性

COLMAP オプションは、次のパラメータで使用できます。

Extract	Replicat
TABLE	MAP

構文

```
TABLE <table spec>, TARGET <table spec>, &
COLMAP ([USEDEFAULTS, ] <target column> = <source expression>);
```

または

```
MAP <table spec>, TARGET <table spec>, &
COLMAP ([USEDEFAULTS, ] <target column> = <source expression>);
```

表 39　TABLE および MAP の引数

引数	説明
`TARGET <table spec>`	ターゲットの所有者および表。MAP では必須です。COLMAP を使用する場合は、TABLE でも必須です。
`<target column>`	データをマップするターゲット列の名前。
`<source expression>`	マップするデータを表す次のいずれかの要素。 ◆ 数値定数 (123 など) ◆ 引用符で囲まれた文字列定数 ("ABCD" など)。 ◆ ソース列の名前 (ORD_DATE など)。 ◆ Oracle GoldenGate 列変換関数を使用した式。次に例を示します。 @STREXT (COL1, 1, 3)
`USEDEFAULTS`	ソース列とターゲット列の名前が同じである場合に、それらの列を自動的にマップするデフォルトのマッピング・ルールを適用します。USEDEFAULTS を使用すると、ソース列の名前が同じであるかどうかにかかわらず、すべてのターゲット列を明示的にマップする必要がなくなります。データ型の変換は、DEFGEN で作成されたデータ定義ファイルに基づいて自動的に行われます。 明示マッピングまたは USEDEFAULTS を使用しますが、この両方を同じ列セットに対して使用することはできません。 デフォルトの列マッピングの詳細は、253 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。 TABLE および MAP 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。

例　次の列マッピングの例では、ソース表 ACCTBL とターゲット表 ACCTTAB のデフォルト列マッピングと明示的な列マッピングの使用について示します。両方の表の大半の列は同じで、次の点が異なります。

- ソース表には CUST_NAME 列がありますが、ターゲット表には NAME 列があります。
- ソース表の 10 桁の PHONE_NO 列は、ターゲット表の個別の AREA_CODE 列、PHONE_PREFIX 列および PHONE_NUMBER 列に対応します。
- ソース表の個別の YY 列、MM 列および DD 列は、ターゲット表の単一の TRANSACTION_DATE 列に対応します。

これらの相違点に対処するため、USEDEFAULTS を使用して同じ列を自動的にマップし、異なる列には明示マッピングおよび変換関数を使用します。

表 40　サンプルの列マッピング

パラメータ文	説明
`MAP sales.acctbl,` `TARGET sales.accttab,`	*1.* ソース表 acctbl をターゲット表 accttab にマップします。
`COLMAP (`	*2.* COLMAP 文を開始します。
`USEDEFAULTS,`	*3.* ターゲット列の名前が同一の場合、ソース列をそのまま移動します。
`name = cust_name,`	*4.* ソース列 cust_name をターゲット列 name にマップします。

`transaction_date =`
`@DATE("YYYY-MM-DD", "YY",`
`YEAR, "MM", MONTH, "DD", DAY),`
	5. @DATE 列変換関数を使用して、ソースの日付列からターゲット列 transaction_date にトランザクション日付を変換します。

`area_code =`
`@STREXT (phone_no, 1, 3),`
`phone_prefix =`
`@STREXT (phone_no, 4, 6),`
`phone_number =`
`@STREXT (phone_no, 7, 10))`
`;`
	6. @STREXT 列変換関数を使用して、ソース列 phone_no を別々のターゲット列 area_code、phone_prefix および phone_number に変換します。

グローバル列マッピングの使用

COLMATCH パラメータを使用して、列マッピングのグローバル・ルールを作成します。COLMATCH では、同じデータセットに対して異なる列名を持つ類似した構造の表どうしをマップできます。このタイプの列をマップする場合、個別の TABLE 文または MAP 文の COLMAP 句で表レベルのマッピングを使用するより、COLMATCH の方が簡単です。

構文

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

表 41　COLMATCH のオプション

引数	説明
`NAMES <target column> =` `<source column>`	列名に基づいてマップします。

表 41　COLMATCH のオプション (続き)

引数	説明
PREFIX <prefix>	指定した名前接頭辞を無視します。
SUFFIX <suffix>	指定した名前接尾辞を無視します。
RESET	前に定義した COLMATCH ルールを、後続の TABLE 文または MAP 文で無効化します。

例　次に、COLMATCH を使用する場合の例を示します。ソース表とターゲット表は、表名と列名が多少異なる以外は同一です。

表 42　COLMATCH のサンプル表

ソース・データベース		ターゲット・データベース	
ACCT 表	**ORD 表**	**ACCOUNT 表**	**ORDER 表**
CUST_CODE	CUST_CODE	CUSTOMER_CODE	CUSTOMER_CODE
CUST_NAME	CUST_NAME	CUSTOMER_NAME	CUSTOMER_NAME
CUST_ADDR	ORDER_ID	CUSTOMER_ADDRESS	ORDER_ID
PHONE	ORDER_AMT	PHONE	ORDER_AMT
S_REP	S_REP	REP	REP
S_REPCODE	S_REPCODE	REPCODE	REPCODE

この例でソース列をターゲット列にマップし、同時に他の表で後続のマップを処理するには、次の構文を使用します。

```
COLMATCH NAMES customer_code = cust_code
COLMATCH NAMES customer_name = cust_name
COLMATCH NAMES customer_address = cust_addr
COLMATCH PREFIX S_
MAP sales.acct, TARGET sales.account, COLMAP (USEDEFAULTS);
MAP sales.ord, TARGET sales.order, COLMAP (USEDEFAULTS);
COLMATCH RESET
MAP sales.reg, TARGET sales.reg;
MAP sales.price, TARGET sales.price;
```

この例のルールに基づいて、次の処理が発生します。

- データは、ソースの acct 表および ord 表の cust_code 列から、ターゲットの account 表および order 表の customer_code 列にマップされます。
- S_ 接頭辞は無視されます。
- phone 列や order_amt 列などの同じ名前を持つ列は、明示的なルールを必要とすることなく、USEDEFAULTS によって自動的にマップされます。「デフォルトの列マッピングの使用」を参照してください。
- 前述のグローバル列マッピングは、reg 表および price 表では無効化されます。これらの表のソース列とターゲット列は、名前がすべて同一であるため、自動的にマップされます。

デフォルトの列マッピングの使用

COLMATCH または COLMAP を使用した明示的な列マッピングが存在しない場合、Oracle GoldenGate では、デフォルトで次のルールに従ってソース列とターゲット列がマップされます。

- 同じ名前を持つ列は、データ型に互換性があれば相互にマップされます。
- オブジェクト名の大 / 小文字が区別されないデータベースでは、名前の比較で列名が大文字に変更されます。大 / 小文字の区別に対応して構成されるデータベースでは、Oracle GoldenGate によって、デフォルトのマッピングで列を評価する際に大 / 小文字の区別が考慮されます。
- ソース列の名前および大 / 小文字がターゲット列のものと正確に一致することが検出された場合、2 つの列はマップされます。大 / 小文字が一致しないことが検出された場合、大 / 小文字とは無関係に、名前がターゲット列と最初に一致した適切なソース列を使用してマップが作成されます。
- どのソース列にも対応していないターゲット列には、データベースによって指定されたデフォルト値が割り当てられます。

デフォルトのマッピングを実行できない場合、ターゲット列は、デフォルトで表 43 に示されているいずれかの値になります。

表 43　一致しないターゲット列のデフォルト値

列タイプ	値
数値	0 (ゼロ)
文字または VARCHAR	空白
日付または日時	現在の日時
NULL 値を使用できる列	NULL

データ型のマッピング

次の例では、Oracle GoldenGate によるデータ型のマップ方法について説明します。

数値の列

数値の列は、ターゲット列の型およびスケールと一致するように変換されます。ターゲット列のスケールがソースのスケールより小さい場合、数値の右側が切り捨てられます。ターゲット列のスケールがソースのスケールより大きい場合、数値の右側に 0 (ゼロ) が埋め込まれます。

英数字の列

文字ベースの列では、文字ベースのデータ型 (VARCHAR、GROUP など) および日時型に加え、引用符で囲んだ文字列リテラルを使用できます。ターゲット列のスケールがソースのスケールより小さい場合、列の右側が切り捨てられます。ターゲット列のスケールがソースのスケールより大きい場合、列の右側に空白が埋め込まれます。

日時の列

日時 (DATE、TIME および TIMESTAMP) の列では、日時および文字の列に加え、文字列リテラルを使用できます。文字の列を日時の列にマップする場合、その列が Oracle GoldenGate の外部 SQL 書式である YYYY-MM-DD:HH:MI:SS.FFFFFF に準拠していることを確認します。

必要な精度は、データ型およびターゲット・プラットフォームに応じて異なります。ターゲット列のスケールがソースのスケールより小さい場合、データの右側が切り捨てられます。ターゲット列のスケールがソースのスケールより大きい場合、列の右側が現在の日時の値で拡張されます。

3.02.0.02 より前のバージョンの Teradata ODBC ドライバを使用する Teradata ターゲット・データベースに日時の列をマップする場合、次の Replicat パラメータを使用できます

USEDATEPREFIX	DATE データ型の値に接頭辞として DATE リテラルを追加します。
USETIMEPREFIX	TIME データ型の値に接頭辞として TIME リテラルを追加します。
USETIMESTAMPPREFIX	TIMESTAMP データ型の値に接頭辞として TIMESTAMP リテラルを追加します。

出力できない文字の処理

Oracle GoldenGate では、出力できない文字を処理するために、次のパラメータが提供されています。

出力できない文字のグローバル置換

出力できない文字をグローバルに修正するには、Replicat のパラメータ・ファイルで REPLACEBADCHAR パラメータおよび REPLACEBADNUM パラメータを使用します。これらのパラメータでは、列マッピング中に出力できない文字または数値が検出された場合に常に置換する値を指定します。使用する場合、これらのパラメータは、パラメータ・ファイルの MAP エントリの前に配置する必要があります。

空白から NULL への変換の制御

(Oracle のみ) 定義の異なるソース表とターゲット表を同期する場合、Oracle GoldenGate では、空白のみを含む列がターゲット表で NULL 値に変換されます。この動作は、SPACESTONULL パラメータによって制御されます。この動作を防ぐには、NOSPACESTONULL パラメータを使用し、Oracle GoldenGate によって単一の空白文字がターゲット列に書き込まれるようにします。

文字の列にある末尾の空白の切捨て

CHAR 列を VARCHAR 列にマップする場合に末尾の空白を切り捨てるかどうかを制御するには、TRIMSPACES パラメータおよび NOTRIMSPACES パラメータを使用します。これらのパラメータは、パラメータ・ファイルのルート・レベルで使用すれば、異なる TABLE 文または MAP 文 (あるいは文のグループ) で切捨て機能の有効化と無効化を切り替えることができ、個別の TABLE 文または MAP 文内で使用すれば、グローバル設定をオーバーライドできます。デフォルトでは、末尾の空白は切り捨てられます。

文字の列にあるバイナリ・データの制御

Oracle GoldenGate では、文字の列として定義されているソース列またはターゲット列に入力されたバイナリ文字は保持されます。マルチバイト・キャラクタまたは任意のタイプの出力できないバイナリ文字 (NULL、改行、シェル・コマンドなど) も保持されます。

トランザクション履歴の使用

Oracle GoldenGate では、ターゲット・レコードに対する変更の履歴を保持して、各変更の原因となった操作に関する情報をマップできます。この履歴は、各レコードの最新バージョンのみを含むのではなく、表に対して実行されたすべての操作の個別レコードを含むトランザクションベースのレポート・システムを作成する場合に役立ちます。

たとえば、CUSTOMER という名前のターゲット表に対して行われた次の一連の操作では、ID "Dave" の履歴は残りません。最後の操作でレコードを削除するため、Dave の口座の履歴や最後の残高を知ることはできません。

図 19 CUSTOMER 表の操作履歴

順序	操作	ID	残高
1	Insert	Dave	1000
2	Update	Dave	900
3	Update	Dave	1250
4	Delete	Dave	1250

一連のレコードとしてこの履歴を保持すると、多くの場面で役立ちます。たとえば、トランザクションの正味の効果を生成できます。

トランザクション・レポートを実装する手順

1. 変更前の値を取得するように Extract を準備するため、Extract のパラメータ・ファイルで GETUPDATEBEFORES パラメータを使用します。変更前の値 (または変更前イメージ) は、更新が実行される前に列に存在する値です。変更前イメージによって、Oracle GoldenGate でトランザクション・レコードを作成できます。
2. 挿入としてすべての操作を適用するように Replicat を準備するため、Replicat のパラメータ・ファイルで INSERTALLRECORDS パラメータを使用します。表に対する各操作は、その表内の新規レコードになります。
3. トランザクション履歴をマップするため、@GETENV 列変換関数の GGHEADER オプションの戻り値を使用します。TABLE パラメータまたは MAP パラメータの COLMAP 文に、ソース式として変換関数を含めます。

例 255 ページの図 19 に示されている一連のサンプル・トランザクションを使用して、データベースの最新状態ではなく、よりトランザクション指向の強い顧客ビューを生成するために、次のパラメータ構成を作成できます。

プロセス	パラメータ文
Extract	`GETUPDATEBEFORES` `TABLE account.customer;`

Replicat
```
INSERTALLRECORDS
MAP sales.customer, TARGET sales.custhist, &
COLMAP (TS = @GETENV ("GGHEADER", "COMMITTIMESTAMP"), &
BEFORE_AFTER = @GETENV ("GGHEADER", "BEFOREAFTERINDICATOR"), &
OP_TYPE = @GETENV ("GGHEADER", "OPTYPE"), &
ID = ID, &
BALANCE = BALANCE);
```

注意 この例は、Oracle GoldenGate プロセスの完全なパラメータ・ファイルを表していません。

この構成によって、各トランザクションの正味合計と、トランザクションの時刻および顧客 ID を戻す次のような問合せが可能になります。

```
SELECT AFTER.ID, AFTER.TS, AFTER.BALANCE - BEFORE.BALANCE
FROM CUSTHIST AFTER, CUSTHIST BEFORE
WHERE AFTER.ID = BEFORE.ID AND AFTER.TS = BEFORE.TS AND
AFTER.BEFORE_AFTER = 'A' AND BEFORE.BEFORE_AFTER = 'B';
```

データのテストおよび変換

データのテストおよび変換は、Extract または Replicat によって実行できます。これらの機能は、Oracle GoldenGate の組込みの列変換関数を TABLE 文または MAP 文の COLMAP 句内で使用することで実装します。これらの変換関数では、次の処理を実行できます。

- 日付の変換。
- 列値の存在確認のテスト。
- 算術演算の実行。
- 数値および文字列の操作。
- NULL 値、無効なデータおよび欠落データの処理。
- テストの実行。

この章では、データ操作に関連するいくつかの Oracle GoldenGate 関数の概要について説明します。詳細なリファレンス情報は、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』を参照してください。

Oracle GoldenGate 関数で提供される機能の範囲外のロジックを使用する必要がある場合は、Oracle GoldenGate ユーザー・イグジットを実装して独自の関数をコールできます。ユーザー・イグジットの詳細は、278 ページを参照してください。

Oracle GoldenGate 変換関数の一般的な構文は次のとおりです。

構文
```
@<function name> (<argument>)
```

構文の要素	説明
`@<function name>`	Oracle GoldenGate 関数の名前。関数名には、接頭辞 @ が付きます (@COMPUTE や @DATE など)。
`<argument>`	関数の引数として、次の要素を指定できます。

構文の要素	説明	
	引数の要素	**例**
	数値定数	`123`
	文字列定数	`"ABCD"` (引用符が必要)
	ソース列の名前	`PHONE_NO`
	算術式	`COL2 * 100`
	比較式	`COL3 > 100 AND COL4 > 0`
	他の Oracle GoldenGate 関数	`AMOUNT = @IF (@COLTEST (AMT, MISSING, INVALID), 0, AMT)`

操作対象または評価対象の列のタイプに適した関数を使用してください。たとえば、数値関数は数値を比較する場合にのみ使用できます。文字列値を比較するには、Oracle GoldenGate 文字比較関数のいずれかを使用します。

例 次の文は、数値関数の @IF を使用して文字列値を比較しているため、失敗します。

```
@IF (SR_AREA = "Help Desk", "TRUE", "FALSE")
```

次の文は、数値を比較しているため、成功します。

```
@IF (SR_AREA = 20, "TRUE", "FALSE")
```

258 ページの「数値および文字列の操作」を参照してください。

注意 **引数解析のエラーは、レコードが処理されるまで検出されないことがあります。プロセスを起動する前に構文を確認してください。**

日付の変換

@DATE、@DATEDIF および @DATENOW の各関数を使用して、日時の取得、日時の計算および日時の変換を行います。

例 次の例では、注文が処理された時間を計算します。

```
ORDER_FILLED = @DATE (
    "YYYY-MM-DD:HH:MI:SS",
    "JTS",
    @DATE ("JTS",
    "YYMMDDHHMISS",
    ORDER_TAKEN_TIME) +
    ORDER_MINUTES * 60 * 1000000)
```

算術演算の実行

算術式の結果を戻すには、@COMPUTE 関数を使用します。関数から戻される値の形式は、文字列です。算術式では、次の要素を組み合せることができます。

- 数値
- 数値を含む列の名前
- 数値を戻す関数
- 算術演算子 :

  +(加算)
  -(減算)
  *(乗算)
  /(除算)
  \ (剰余)

- 比較演算子 :

  > (より大きい)
  >=(以上)
  <(より小さい)
  <=(以下)
  =(等しい)
  <>(等しくない)

  比較から導出される結果は、0 (ゼロ)(FALSE に相当) または 0 以外 (TRUE に相当) です。

- カッコ (式内の結果のグループ化用)
- 結合演算子の AND、OR。Oracle GoldenGate では、結合式の必要な部分のみが評価されます。文が FALSE になると、式の残り部分は無視されます。これは、欠落または NULL の可能性があるフィールドを評価する場合に役立ちます。たとえば、COL1 の値が 25 で、COL2 の値が 10 の場合、次の式を使用できます。

  `@COMPUTE (COL1 > 0 AND COL2 < 3)` は、**0 を戻します。**
  `@COMPUTE (COL1 < 0 AND COL2 < 3)` は、**0 を戻します。**COL2 < 3 は、評価されません。
  `@COMPUTE ((COL1 + COL2)/5)` は、**7 を戻します。**

@COMPUTE の省略

@COMPUTE キーワードは、式を関数の引数として渡す場合には必要ありません。

例

```
@STRNUM ((AMOUNT1 + AMOUNT2), LEFT)
```

次の式は、前述の式と同じ結果を戻します。

```
@STRNUM ((@COMPUTE (AMOUNT1 + AMOUNT2), LEFT)
```

数値および文字列の操作

数値および文字列を変換するため、Oracle GoldenGate では次の関数が提供されています。

表 44　数値および文字のための変換関数

用途	変換関数
バイナリ文字列または文字列の数値への変換	@NUMBIN @NUMSTR
数値の文字列への変換	@STRNUM
文字列の比較	@STRCMP @STRNCMP
文字列の連結	@STRCAT @STRNCAT
文字列からの抽出	@STREXT @STRFIND
文字列の長さの返却	@STRLEN
ある文字列と別の文字列の置換	@STRSUB
文字列の大文字への変換	@STRUP
先頭または末尾 (あるいはその両方) の空白の切捨て	@STRLTRIM @STRRTRIM @STRTRIM

NULL 値、無効なデータおよび欠落データの処理

列データが欠落しているか、無効であるか、または NULL である場合、Oracle GoldenGate 変換関数では、それぞれに対応する値が戻されます。

例　BALANCE は 1000 でも、AMOUNT が NULL の場合、次の式では NULL が戻されます。

```
NEW_BALANCE = @COMPUTE (BALANCE + AMOUNT)
```

これらの例外条件によって、計算全体が無効になります。変換を確実に成功させるには、@COLSTAT、@COLTEST および @IF の各関数を使用して、例外条件をテスト (および上書き) してください。

@COLSTAT の使用

@COLSTAT 関数を使用して、列が欠落しているか、NULL であるか、または無効であることを示すインジケータを Extract または Replicat に戻します。このインジケータは、追加の変換関数が含まれるより複雑な操作式の一部として使用できます。

例 1　次の例では、ターゲット列の ITEM に NULL を戻します。

```
ITEM = @COLSTAT (NULL)
```

例 2 次の @IF の計算では、PRICE および QUANTITY が 0 (ゼロ) 未満の場合に、@COLSTAT を使用してターゲット列に NULL を戻します。

```
ORDER_TOTAL = PRICE * QUANTITY, @IF (PRICE < 0 AND QUANTITY < 0, @COLSTAT(NULL))
```

@COLTEST の使用

@COLTEST 関数を使用して、次の条件をチェックします。

- PRESENT は、列が存在して NULL ではないかどうかをテストします。
- NULL は、列が存在して NULL であるかどうかをテストします。
- MISSING は、列が存在していないかどうかをテストします。
- INVALID は、列が存在して無効なデータを含んでいるかどうかをテストします。

例

```
@COLTEST (AMOUNT, NULL, INVALID)
```

@IF の使用

@IF 関数を使用して、条件に基づいて 2 つのうちのいずれかの値を戻します。この関数を @COLSTAT 関数および @COLTEST 関数とともに使用して、1 つ以上の例外条件をテストする条件付き引数を開始し、そのテストの結果に基づいて処理を実行します。

例

```
NEW_BALANCE = @IF (@COLTEST (BALANCE, NULL, INVALID) OR
@COLTEST (AMOUNT, NULL, INVALID), @COLSTAT (NULL), BALANCE + AMOUNT)
```

この変換では次のいずれかの値が戻されます。

- NULL(BALANCE または AMOUNT が NULL または INVALID の場合)
- MISSING(いずれかの列が欠落している場合)
- 列の合計

テストの実行

@CASE、@VALONEOF および @EVAL の各関数によって、データを操作またはマップする前にテストを実行するための追加の方法が提供されます。

@CASE の使用

@CASE を使用して、一連の値テストに応じて値を選択します。

例

```
@CASE (PRODUCT_CODE, "CAR", "A car", "TRUCK", "A truck")
```

この例では、次の要素が戻されます。

- "A car"(PRODUCT_CODE が "CAR" の場合)
- "A truck"(PRODUCT_CODE が "TRUCK" の場合)
- FIELD_MISSING インジケータ (PRODUCT_CODE が他のどちらの条件も満たさない場合)

@VALONEOF の使用

@VALONEOF を使用して、列または文字列と値リストを比較します。

例

```
@IF (@VALONEOF (STATE, "CA", "NY"), "COAST", "MIDDLE")
```

この例では、STATE が CA または NY の場合、式により "COAST" が戻されます。これは、値が 0 (ゼロ) ではない (True に相当) 場合に @IF によって戻されるレスポンスです。

@EVAL の使用

@EVAL を使用して、独立した一連の条件テストに基づいて値を選択します。

例

```
@EVAL (AMOUNT > 10000, "high amount", AMOUNT > 5000, "somewhat high")
```

この例では、次の要素が戻されます。

- "high amount"(AMOUNT が 10000 を超える場合)
- "somewhat high"(前述の条件が満たされなかった場合で、AMOUNT が 5000 を超え、10000 以下の場合)
- FIELD_MISSING インジケータ (どちらの条件も満たされない場合)

トークンの使用

データを抽出して、証跡レコード・ヘッダーのユーザー・トークン領域内に格納できます。トークン・データを取得して、Oracle GoldenGate による情報の配信方法をカスタマイズするために様々な用途で使用できます。たとえば、トークン・データを次の用途で使用できます。

- 列マップ
- SQLEXEC 文によってコールされるストアド・プロシージャ
- ユーザー・イグジット
- マクロ

トークンの定義

トークンを使用するには、トークン名を定義してデータに関連付けます。データとして使用できるのは、システム、データベース、トランザクションまたはレコードから取得された値か、問合せ、プロシージャまたは他のコール・ファンクションから取得された値の任意の英数字データです。

レコード・ヘッダーのトークン領域には、最大 2,000 バイトのデータを格納できます。トークン名、データ長およびデータ自体がこの領域に収まる必要があります。

トークンを定義するには、Extract パラメータ・ファイルの TABLE パラメータの TOKENS オプションを使用します。

構文

```
TABLE <table spec>, TOKENS (<token name> = <token data> [, ...]);
```

条件 :

- <table spec> は、ソース表の名前です。所有者名は、表名の前に付ける必要があります。
- <token name> は、ユーザー指定のトークン名です。この名前には、任意の数の英数字を使用できます。大 / 小文字は区別されません。
- <token data> は、最大 2000 バイトの文字列です。データとして指定できるのは、二重引用符で囲んだ定数か、Oracle GoldenGate 列変換関数の結果です (次の例を参照)。

例

```
TABLE ora.oratest, TOKENS (
TK-OSUSER = @GETENV ("GGENVIRONMENT" , "OSUSERNAME"),
TK-GROUP = @GETENV ("GGENVIRONMENT" , "GROUPNAME")
TK-HOST =  @GETENV("GGENVIRONMENT" , "HOSTNAME"));
```

この例に示されているとおり、Oracle GoldenGate の @GETENV 関数は、トークン・データを移入するための効果的な方法です。この関数には、トークンにマップしてターゲット・システムで列マッピングに使用できる環境情報を取得するための複数のオプションがあります。@GETENV 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。

ターゲット表でのトークン・データの使用

ターゲット表にトークン・データをマップするには、Replicat の MAP 文に含まれる COLMAP 句のソース式で @TOKEN 列変換関数を使用します。@TOKEN 関数によって、マップするトークンの名前を提供します。@TOKEN を使用した COLMAP 構文は、次のとおりです。

構文

```
COLMAP (<target column> = @TOKEN ("<token name>"))
```

例

次の MAP 文では、host や gg_group などのターゲット列が、tk-host や tk-group などのトークンにマップされます。

```
MAP ora.oratest, TARGET ora.rpt,
COLMAP (USEDEFAULTS,
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"));
```

この例のトークンは、証跡のレコード・ヘッダー内では次のように示されます。

```
User tokens:
tk-host          :sysA
tk-group         :extora
tk-osuser        :jad
tk-domain        :admin
tk-ba_ind        :B
tk-commit_ts     :2011-01-24 17:08:59.000000
tk-pos           :3604496
tk-rba           :4058
tk-table         :oratest
tk-optype        :insert
```

Unicode とネイティブ文字のマッピングおよび変換

Oracle GoldenGate では、文字の列内の文字を Unicode で、または Windows、UNIX および Linux オペレーティング・システムのネイティブ文字エンコーディングで表すためのエスケープ・シーケンスの使用がサポートされています。エスケープ・シーケンスは、TABLE 文または MAP 文内の次の要素で使用できます。

- WHERE 句
- Unicode 列に Unicode 文字を割り当てるか、列にネイティブ・エンコーディングされた文字を割り当てる COLMAP 句
- COLMAP 句内の Oracle GoldenGate 列変換関数

Oracle GoldenGate では、次のタイプのエスケープ・シーケンスがサポートされます。

- \uFFFF: Unicode エスケープ・シーケンス。
- \377: 8 進エスケープ・シーケンス
- \xFF: 16 進エスケープ・シーケンス

次の制限が適用されます。

- このサポートは、7 ビット ASCII に相当する U+0000 から U+007F の間の UTF-16 コード・ポイントに限定されます。
- ソース列とターゲット列の両方が Unicode である必要があります。
- ソースとターゲットのデータ型は、同一である必要があります (NCHAR と NCHAR など)。

エスケープ・シーケンスを使用する手順

各エスケープ・シーケンスは、逆斜線 (コード・ポイント U+005C) で始め、次に文字コード・ポイントを続けます。(逆斜線は、一般的にはバックスラッシュ記号と呼ばれます。) パラメータ文または列変換関数の入力文字列内で、実際の文字のかわりにエスケープ・シーケンスを使用します。

\uFFFF Unicode エスケープ・シーケンスを使用する手順

- 小文字の u で始め、次に正確に 4 桁の 16 進数を続ける必要があります。
- サポートされる範囲 :
  - 0 から 9(U+0030 から U+0039)
  - A から F(U+0041 から U+0046)
  - a から f(U+0061 から U+0066)
- NCHAR 列および NVARCHAR 列に使用可能なエスケープ・シーケンスは、これのみです。
- サロゲート・ペアはサポートされていません。

例 \u20ac は、ユーロ通貨記号の Unicode エスケープ・シーケンスです。

注意 **信頼できるクロス・プラットフォーム・サポートのために、Unicode エスケープ・シーケンスを使用してください。8 進および 16 進のエスケープ・シーケンスは、異なるオペレーティング・システムでは標準化されていません。**

\377 8 進エスケープ・シーケンスを使用する手順

- 正確に 3 桁の 8 進数が含まれる必要があります。

- サポートされる範囲 :
  - 1 桁目の範囲は 0 から 3(U+0030 から U+0033)
  - 2 桁目と 3 桁目の範囲は 0 から 7(U+0030 から U+0037)

例 \200 は、Microsoft Windows でのユーロ通貨記号の 8 進エスケープ・シーケンスです。

\xFF 16 進エスケープ・シーケンスを使用する手順

- 小文字の x で始め、次に正確に 2 桁の 16 進数を続ける必要があります。
- サポートされる範囲 :
  - 0 から 9(U+0030 から U+0039)
  - A から F(U+0041 から U+0046)
  - a から f(U+0061 から U+0066)

例 \x80 は、Microsoft Windows でのユーロ通貨記号の 16 進エスケープ・シーケンスです。

第 18 章

Oracle GoldenGate 処理のカスタマイズ

カスタム処理の概要

処理をカスタマイズおよび効率化する場合、次の機能が役に立ちます。

- SQL プロシージャおよび問合せ
- マクロ
- ユーザー・イグジット
- イベント・マーカー

SQLEXEC を使用したコマンド、ストアド・プロシージャおよび問合せの実行

Oracle GoldenGate の SQLEXEC パラメータによって、Extract および Replicat でデータベースと通信し、次の処理を実行できます。

- データベース・コマンド、ストアド・プロシージャまたは SQL 問合せの実行によるデータベース機能の実行、結果の返却 (SELECT 文) または DML 操作の実行 (INSERT、UPDATE、DELETE) が可能です。
- プロシージャから出力パラメータを取得して FILTER 句または COLMAP 句に入力できます。

SQLEXEC を使用して実行できる処理

SQLEXEC によって、Oracle GoldenGate ではデータベースのネイティブ SQL を使用してカスタム処理命令を実行できるため、Oracle GoldenGate とデータベース両方の機能が拡張されます。

- ストアド・プロシージャおよび問合せを使用して、データベースを対象とするデータの選択または挿入、データの集計、データの非正規化または正規化、あるいは入力にデータベース操作を必要とする他の任意の機能を実行できます。Oracle GoldenGate では、入力を取得するストアド・プロシージャと出力を生成するストアド・プロシージャがサポートされます。
- データベース・コマンドを発行して、Oracle GoldenGate 処理を効率化するために必要なデータベース機能を実行できます (ターゲット表に対するトリガーを無効化して、その後再度有効化するなど)。

SQLEXEC でサポートされるデータベースおよびデータ型

次に、SQLEXEC でサポートされるデータベースおよび入力と出力の各パラメータでサポートされるデータ型を示します。

DB2 LUW and z/OS

- CHAR
- VARCHAR
- DATE
- すべての数値データ型
- BLOB データ型

Ingres

LOB 以外のすべてのデータ型

MySQL

TEXT および BLOB 以外のすべてのデータ型

Oracle

*次を除く*すべての Oracle タイプがサポートされます。

- BFILE
- BLOB
- CFILE
- CLOB
- NCLOB
- NTY

SQL Server

- CHAR
- VARCHAR
- DATETIME
- すべての数値データ型
- 長さが 200 バイト未満のイメージおよびテキスト・データ型
- TIMESTAMP パラメータ・タイプはネイティブにサポートされませんが、他のデータ型をパラメータとして使用し、ストアド・プロシージャ内で TIMESTAMP 形式に値を変換できます。

Sybase

TEXT、IMAGE および UDT 以外のすべてのデータ型

Teradata

Oracle GoldenGate でサポートされるすべての Teradata データ型

SQLEXEC の使用方法

SQLEXEC パラメータは、次の方法で使用できます。

- TABLE 文または MAP 文の句として
- Extract または Replicat のパラメータ・ファイルのルート・レベルに存在するスタンドアロン・パラメータとして

TABLE 文または MAP 文内での SQLEXEC の実行

TABLE 文または MAP 文内で使用する場合、SQLEXEC では、パラメータを受け渡すことができます。このコマンドは、プロシージャおよび問合せに使用できますが、データベース・コマンドには使用できません。

TABLE 文または MAP 文内でプロシージャを実行する手順

構文

```
SQLEXEC (SPNAME <sp name>,
[ID <logical name,]
{PARAMS <param spec> | NOPARAMS})
```

引数	説明
SPNAME	ストアド・プロシージャを実行するための句を開始する必須キーワード。
<sp name>	実行するストアド・プロシージャの名前を指定します。
ID <logical name>	プロシージャの論理名を定義します。このオプションを使用して、TABLE 文または MAP 文内でプロシージャを複数回実行します。プロシージャを 1 回のみ実行する場合は不要です。
PARAMS <param spec> | NOPARAMS	プロシージャでパラメータを受け入れるかどうかを指定します。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 (268 ページの「入力パラメータと出力パラメータの使用」を参照)。

TABLE 文または MAP 文内で問合せを実行する手順

構文

```
SQLEXEC (ID <logical name>, QUERY " <sql query> ",
{PARAMS <param spec> | NOPARAMS})
```

引数	説明
ID <logical name>	問合せの論理名を定義します。論理名は、問合せ結果から値を抽出するために必要です。ID <logical name> は、問合せによって戻される列値を参照します。
QUERY " <sql query> "	データベースに対して実行する SQL 問合せ構文を指定します。これにより、SELECT 文を使用して結果を戻すことや、INSERT 文、UPDATE 文または DELETE 文を使用してデータベースを変更できます。問合せは、引用符で囲み、全体を 1 行で記述する必要があります。
PARAMS <param spec> | NOPARAMS	問合せでパラメータを受け入れ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 (268 ページの「入力パラメータと出力パラメータの使用」を参照)。

スタンドアロン文としての SQLEXEC の実行

Extract または Replicat のパラメータ・ファイルでスタンドアロン・パラメータ文として使用する場合、SQLEXEC は、ストアド・プロシージャ、問合せまたはデータベース・コマンドを実行できます。この場合、このパラメータを特定の表に関連付ける必要はなく、一般的な SQL 操作を実行するために使用できます。たとえば、Oracle GoldenGate のデータベース・ユーザー・アカウントがアイドル時にタイムアウトされるように構成されている場合、SQLEXEC を使用して、Oracle GoldenGate が見かけ上アイドル状態とならないように、定義した間隔で問合せを実行できます。別の例としては、SQLEXEC を使

用して、ターゲット・トリガーの無効化などの重要なデータベース・コマンドを発行できます。スタンドアロンの SQLEXEC 文では、入力パラメータを受け入れることや、出力パラメータを戻すことはできません。

構文

パラメータ構文	用途
`SQLEXEC "exec <procedure name>()"`	ストアド・プロシージャの実行
`SQLEXEC "<sql query>"`	問合せの実行
`SQLEXEC "<database command>"`	データベース・コマンドの実行

引数	説明
`"exec <procedure name> ()"`	実行するストアド・プロシージャの名前を指定します。文は二重引用符で囲む必要があります。 例 : `SQLEXEC "exec prc_job_count ()"`
`"<sql query>"`	実行する問合せの名前を指定します。問合せは、全体を 1 行で記述し、二重引用符で囲む必要があります。最適な結果を得るために、開始引用符の後と終了引用符の前に空白を入力してください。
`"<database command>"`	実行するデータベース・コマンドを指定します。データベースにとって有効なコマンドである必要があります。

SQLEXEC には、処理動作、メモリー使用およびエラー処理を制御するためのオプションがあります。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。

入力パラメータと出力パラメータの使用

Oracle GoldenGate では、TABLE 文または MAP 文内の SQLEXEC で実行されるプロシージャまたは問合せを対象として入力値と出力値を受け渡すためのオプションを提供しています。

入力値を渡す手順

ストアド・プロシージャまたは問合せ内の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。

構文

```
PARAMS ([OPTIONAL | REQUIRED] <param name> = {<source column> | <GG function>}
[, ...] )
```

条件 :

- OPTIONAL は、パラメータ値が SQL の実行にとって必須では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から欠落している場合や、ソース列が欠落しているために列変換関数を正常に完了できない場合でも、SQL は実行されます。
- 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> は、次のいずれかです。

  ストアド・プロシージャの場合、入力を受け入れるプロシージャの任意のパラメータの名前です (参照表の列など)。

  Oracle 問合せの場合、問合せの任意の入力パラメータの名前 (先頭のコロンを除く) です。たとえば、:param1 は、PARAMS 句では param1 として指定します。

  Oracle 以外の問合せの場合は Pn です。n は、1 から始まる文内のパラメータの番号です。たとえば、2 つのパラメータを含む問合せの <param name> エントリは、P1 および P2 です。

- {<source column> | <GG function} は、プロシージャに入力値として渡す列または Oracle GoldenGate 変換関数です。

出力値を渡す手順

ストアド・プロシージャまたは問合せからの値を入力値として FILTER 句または COLMAP 句に渡すには、次の構文を使用します。

構文

```
{<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す。この引数は、現在の Oracle GoldenGate プロセスの有効期間中にプロシージャを 1 回実行する場合にのみ使用します。
- <logical name> は、SQLEXEC の ID オプションで指定された論理名です。この引数は、問合せを実行する場合、またはストアド・プロシージャを複数回実行する場合に使用します。
- <parameter> は、パラメータの名前または RETURN_VALUE(戻り値を抽出する場合) です。

例

次の例では、SQLEXEC を使用して、コードに従って説明を戻す問合せを実行する LOOKUP というストアド・プロシージャを実行します。その後、結果を NEWACCT_VAL というターゲット列にマップします。

LOOKUP プロシージャの内容 :

```
CREATE OR REPLACE PROCEDURE LOOKUP
(CODE_PARAM IN VARCHAR2, DESC_PARAM OUT VARCHAR2)

BEGIN
    SELECT DESC_COL
    INTO DESC_PARAM
    FROM LOOKUP_TABLE
    WHERE CODE_COL = CODE_PARAM
END;
```

MAP 文の内容 :

```
MAP sales.account, TARGET sales.newacct, &
    SQLEXEC (SPNAME lookup, PARAMS (code_param = account_code)), &
        COLMAP (newacct_id = account_id, newacct_val = lookup.desc_param);
```

SQLEXEC は、LOOKUP ストアド・プロシージャを実行します。SQLEXEC 句内の PARAMS (code_param = account_code) 文では、code_param をプロシージャ・パラメータとして識別し、account 表の account_code 列から入力値を受け入れます。

Replicat では、列マップを実行する前に LOOKUP ストアド・プロシージャが実行されるため、COLMAP 句によりその結果を抽出して newacct_val 列にマップできます。

例　次の例では、前述の例で使用されているものと同じロジックを実装していますが、ストアド・プロシージャのかわりに SQL 問合せを実行し、列マップで @GETVAL 関数を使用します。

> **注意**　このドキュメントのスペース上の制約のため、問合せは複数の行にわたって記載されています。ただし、実際の問合せは 1 行で記載する必要があります。また、Oracle GoldenGate のパラメータ文を複数の行に分割する場合、行末にアンパサンド (&) が必要です。

Oracle Database:

```
MAP sales.account, TARGET sales.newacct, &
SQLEXEC (ID lookup, &
QUERY "select desc_col 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));
```

Oracle 以外のデータベース :

```
MAP sales.account, TARGET sales.newacct, &
SQLEXEC (ID lookup, &
QUERY "select desc_col desc_param from lookup_table where code_col = ?", &
PARAMS (p1 = account_code)), &
COLMAP (newacct_id = account_id, newacct_val = &
@getval (lookup.desc_param));
```

> **注意**　プロシージャまたは問合せにパラメータが含まれる場合、追加の SQLEXEC オプションを使用できます。『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』の SQLEXEC に関する詳細な説明を参照してください。

エラーの処理

SQLEXEC を実装する場合に考慮する必要のあるエラー状態には、次の 2 つのタイプがあります。

1. 列マップで、ソース・データベースの操作から欠落している列が必要とされる場合。この状態は、データベースでトランザクション・ログの圧縮更新を使用している場合に、更新操作で発生する可能性があります。デフォルトでは、必要な列が欠落している場合、または Oracle GoldenGate 列変換関数で列の欠落状態が発生した場合には、ストアド・プロシージャは実行されません。その後、ストアド・プロシージャから出力パラメータを抽出するしようとすると、COLMAP 句または FILTER 句で列の欠落状態が発生します。

   または

2. データベースでエラーが生成された場合。

欠落している列値を処理する手順

@COLTEST 関数を使用して、渡されたパラメータの結果をテストし、必要に応じて欠落値を埋めるために列の代替値をマップします。別の方法として、列値を使用できるようにするため、TABLE パラメータの FETCHCOLS オプションまたは FETCHCOLSEXCEPT オプションを使用して、ログに存在しない値をデータベースからフェッチできます。列をフェッチするかわりに、該当する列のロギングを有効化する方が効果的な場合があります。

データベース・エラーを処理する手順

SQLEXEC 句の ERROR オプションを使用して、Oracle GoldenGate に次のいずれかの方法でレスポンスするように指示します。

表 45　ERROR オプション

アクション	説明
IGNORE	Oracle GoldenGate は、ストアド・プロシージャまたは問合せに関連付けられたすべてのエラーを無視し、処理を継続します。どのパラメータ抽出の結果も、列の欠落状態となります。この設定がデフォルトです。
REPORT	ストアド・プロシージャまたは問合せに関連付けられたすべてのエラーを、廃棄ファイルにレポートします。このレポートは、エラーの原因をトレースする場合に役立ちます。レポートには、エラーの説明と、プロシージャまたは問合せを対象に受け渡されたパラメータの値が含まれます。Oracle GoldenGate は、エラーのレポート後に処理を継続します。
RAISE	Replicat のパラメータ・ファイルに指定された REPERROR パラメータの設定による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 文または MAP 文に関連付けられた他のストアド・プロシージャまたは問合せの処理を継続します。
FINAL	実行方法は RAISE とほぼ同じですが、プロシージャまたは問合せに関連付けられたエラーが発生すると、残りのすべてのストアド・プロシージャおよび問合せは省略されます。エラー処理は、エラーが発生した直後に起動されます。
FATAL	Oracle GoldenGate は、プロシージャまたは問合せに関連付けられたエラーが発生すると、即座に異常終了します。

SQLEXEC のその他のガイドライン

- TABLE または MAP の 1 つのエントリで、最大 20 のプロシージャまたは問合せを実行できます。これらは、パラメータ文にリストされている順序で実行されます。
- Oracle GoldenGate ユーザーによるデータベース・ログインは、SQLEXEC 句に先行する必要があります。データベース・タイプおよび構成済の認証方式に応じて、Extract パラメータ・ファイルで SOURCEDB または USERID パラメータ (あるいはその両方) を使用するか、Replicat パラメータ・ファイルで TARGETDB または USERID パラメータ (あるいはその両方) を使用します。
- SQL は、Oracle GoldenGate 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・プロシージャを実行し、RDBM 提供のプロシージャをコールする権限を持っている必要があります。
- デフォルトでは、出力値はパラメータごとに 255 バイトで切り捨てられます。これより長いパラメータを必要とする場合、SQLEXEC の MAXVARCHARLEN オプションを使用します。

- ターゲット・データベースを変更するストアド・プロシージャまたは問合せを使用する場合、SQLEXEC 句の DBOP オプションを使用します。DBOP によって、変更がデータベースで適切にコミットされます。それ以外の場合、変更がロールバックされる可能性があります。
- ストアド・プロシージャまたは問合せ内のデータベース操作は、元のトランザクションと同じコンテキストでコミットされます。
- SQLEXEC を使用して主キー列の値を更新しないでください。SQLEXEC を使用してキー列の値を更新すると、元のキー値が使用できなくなるため、Replicat プロセスで後続の更新操作または削除操作を実行できなくなります。キー値を変更する必要がある場合、元のキー値を別の列にマップして、TABLE パラメータまたは MAP パラメータの KEYCOLS オプションを使用してその列を指定できます。
- DB2 では、Oracle GoldenGate は ODBC の SQLExecDirect 関数を使用して SQL 文を動的に実行します。この場合、接続しているデータベース・サーバーでは、文を動的に準備できる必要があります。ODBC では、(リクエストされた間隔で)SQL 文が実行されるたびに、その文が準備されます。通常、この処理が Oracle GoldenGate ユーザーにとって問題になることはありません。詳細は、DB2 のドキュメントを参照してください。
- SQLEXEC は、パススルー・モードのデータ・ポンプ Extract によって処理される表に対して使用しないでください。
- SQLEXEC のストアド・プロシージャまたは問合せの影響を受けるすべてのオブジェクトは、SQL の実行前に適切な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 の実行前に発生している必要があります。
- スタンドアロンの SQLEXEC 文の影響を受けるすべてのオブジェクトは、Oracle GoldenGate プロセスが起動する前に存在している必要があります。このため、DDL サポートは、これらのオブジェクトに対して無効にする必要があります。そうしないと、SQLEXEC のプロシージャまたは問合せが実行される前に、DDL 操作によって構造が変更されたり、オブジェクトが削除される可能性があります。

Oracle GoldenGate マクロを使用した作業の簡略化および自動化

パラメータ・ファイルで Oracle GoldenGate マクロを使用することで、パラメータ、コマンドおよび変換関数を構成および再利用できます。マクロは次の用途で使用できます。

- パラメータ文の複数回使用の実装。
- 複数のコマンドの統合。
- 他のマクロの起動。
- よく使用されるマクロのライブラリへの格納。

Oracle GoldenGate マクロは、次のパラメータ・ファイルと連携して動作します。

- Manager
- Extract
- Replicat

マクロは、パススルー・モードのデータ・ポンプ Extract によって処理される表のデータを操作するために使用しないでください。

マクロの定義

Oracle GoldenGate マクロを定義するには、パラメータ・ファイルで MACRO パラメータを使用します。

構文

```
MACRO #<macro name>
PARAMS (#<p1>, #<p2> [, ...])
BEGIN
<macro body>
END;
```

表 46　マクロ定義の引数

引数	説明
MACRO	必須です。Oracle GoldenGate マクロを示します。
#<macro name>	マクロの名前。273 ページの「マクロのネーミング規則」を参照してください。
PARAMS (#<p1>, #<p2>)	パラメータの名前。273 ページの「マクロのネーミング規則」を参照してください。パラメータ文はオプションです。パラメータを使用する場合、リスト内の各パラメータをカンマで区切るか、各パラメータを個別の行にリストして可読性を向上します (必ず開きカッコと閉じカッコを使用してパラメータ・リストを囲みます)。275 ページの「入力パラメータの使用」を参照してください。
BEGIN	マクロ本体を開始します。マクロ本体の前に指定する必要があります。

<macro body>	マクロ本体。マクロ本体には、次のタイプの文を含めることができます。
	◆ 単純なパラメータ文 (次の例を参照):
	`COL1 = COL2`
	◆ 複雑な文 (次の例を参照):
	`COL1 = #val2`
	◆ 他のマクロの起動 (次の例を参照):
	`#colmap (COL1, #sourcecol)`

END	マクロ定義を終了します。

マクロのネーミング規則

マクロおよびパラメータを作成する場合、次のネーミング規則に従ってください。

- マクロ名とパラメータ名は、マクロ文字で始める必要があります。デフォルトのマクロ文字は、番号記号 (#) です (#macro1 や #param1 など)。# マクロ文字で始まるパラメータ・ファイル内のすべての要素は、マクロまたはマクロ・パラメータとみなされます。引用符内のマクロ名またはパラメータ名は、無視されます。

  マクロ文字は、# 以外の文字に変更できます。たとえば、表名に # 文字が含まれる場合にマクロ文字を変更できます。異なるマクロ文字を定義するには、パラメータ・ファイルで MACRO 文の前に MACROCHAR <character> パラメータを配置します。次の例では、$ をマクロ文字として定義しています。

```
MACROCHAR $
MACRO $mymac
PARAMS ($p1)
BEGIN
col = $p1
END;
```

MACROCHAR パラメータは、1 回のみ使用できます。

- マクロ名とパラメータ名では、大 / 小文字は区別されません。
- マクロおよびパラメータで有効な文字は、先頭のマクロ文字 (# またはユーザー定義) に加え、英数字とアンダースコア (_) です。

マクロの起動

マクロを起動するには、パラメータ・ファイルでマクロを実行するすべての場所に起動文を配置します。

構文

```
[<target> =] <macro name> (<val1>, <val2> [, ...])
```

表 47　マクロ起動の引数

引数	説明

<target> =	オプションです。マクロ処理の結果を割り当てるターゲット (通常はターゲット列) を指定します。たとえば、次の場合、列 DATECOL1 がターゲットです。
	`DATECOL1 = #make_date (YR1, MO1, DAY1)`
	ターゲットなしの構文は次のとおりです。
	`#make_date (YR1, MO1, DAY1)`

<macro name>	マクロの名前。次に例を示します。 `#assign_date`

(<val1>, <val2>)	マクロ定義の PARAMS 文で定義されているパラメータの値。次に例を示します。
	`#custdate (#year, #month, #day)`
	マクロでパラメータを必要としない場合、パラメータ値のリストは空になりますが、開きカッコと閉じカッコは必要です。次に例を示します。
	`#no_params_macro ()`
	
	有効なパラメータ値は、プレーン・テキスト、引用符付きテキスト、および他のマクロの起動です。次に例を示します。
	`my_col_1`
	`"your text here"`
	`#mycalc (col2, 100)`
	`#custdate (#year, #month, #day)`
	`#custdate (#getyyyy (#yy), #month, #day)`

入力パラメータの使用

マクロでの入力パラメータの使用は、オプションです。

パラメータを使用してマクロを実行する手順

MACRO パラメータの PARAMS 引数を使用して、マクロに入力パラメータの内容を示します。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ます。また、マクロを起動する場合、その起動に各パラメータの値が含まれる必要があります。

例

次の例は、パラメータを使用するマクロによって列マッピングを改良する方法を示しています。次の例では、マクロで #year、#month および #day の各パラメータを定義し、独自仕様の日付書式を変換します。

```
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;
```

カッコ内に各パラメータの値リストを指定して、マクロを次のように起動します。

```
MAP sales.acct_tab, TARGET sales.account,
COLMAP
(
targcol1 = sourcecol1,
datecol1 = #make_date(YR1, MO1, DAY1),
datecol2 = #make_date(YR2, MO2, DAY2)
);
```

マクロは次のように展開されます。

```
MAP sales.acct_tab, TARGET sales.account,
COLMAP
(
targcol1 = sourcecol1,
datecol1 = @DATE("YYYY-MM-DD", "CC", @IF(YR1 < 50, 20, 19),"YY", YR1, "MM", MO1,
"DD", DAY1),
datecol2 = @DATE("YYYY-MM-DD", "CC", @IF(YR2 < 50, 20, 19),"YY", YR2, "MM", MO2,
"DD", DAY2)
);
```

パラメータなしでマクロを実行する手順

パラメータを指定せずにマクロを作成できます。たとえば、よく使用するコマンドのセットに対してマクロを作成できます。次に例を示します。

例

```
MACRO #option_defaults
BEGIN
GETINSERTS
GETUPDATES
GETDELETES
INSERTDELETES
END;
```

カッコ内にパラメータ値を指定せずに、マクロを次のように起動します。

```
#option_defaults ()
IGNOREUPDATES
MAP owner.srctab, TARGET owner.targtab;

#option_defaults ()
MAP owner.srctab2, TARGET owner.targtab2;
```

マクロは次のように展開されます。

```
GETINSERTS
GETUPDATES
GETDELETES
INSERTDELETES
IGNOREUPDATES
MAP owner.srctab, TARGET owner.targtab;

GETINSERTS
GETUPDATES
GETDELETES
INSERTDELETES
MAP owner.srctab2, TARGET owner.targtab2;
```

パラメータ置換

Oracle GoldenGate では、次のルールに従ってマクロ本体のパラメータ値が置換されます。

1. マクロ・プロセッサは、マクロ本体を読み取って、PARAMS 文に指定されたパラメータ名のインスタンスを検索します。
2. パラメータ名が出現するたびに、起動時に指定された対応するパラメータ値に置換されます。
3. パラメータ名がリスト内に出現しない場合、マクロ・プロセッサは、かわりにその項目が別のマクロの起動ではないかどうかを評価します。(276 ページの「マクロからの他のマクロの起動」を参照してください。)

マクロからの他のマクロの起動

マクロから他のマクロを起動するには、次のようなマクロ定義を作成します。

```
MACRO #assign_date
PARAMS (#target_col, #year, #month, #day)
BEGIN
#target_col = #make_date (#year, #month, #day)
END;
```

マクロ・ライブラリの作成

1 つ以上のマクロを含むマクロ・ライブラリを作成できます。マクロ・ライブラリを使用すると、マクロを 1 回定義するだけで、そのマクロを多くのパラメータ・ファイルで使用できます。

マクロ・ライブラリを作成する手順

1. テキスト・エディタで新規ファイルを開きます。
2. 必要に応じて、コメント行を使用してライブラリを説明します。
3. 273 ページの「マクロの定義」に記載されている構文を使用して、各マクロの構文を入力します。
4. ファイルを次の形式で Oracle GoldenGate ディレクトリの dirprm サブディレクトリに保存します。

```
<filename>.mac
```

条件： <filename> は、ファイルの名前です。.mac 拡張子によって、ファイルをマクロ・ライブラリとして定義します。

例 次の datelib というサンプル・ライブラリには、#make_date および #assign_date という 2 つのマクロが含まれます。

```
-- datelib macro library
--
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;
MACRO #assign_date
PARAMS (#target_col, #year, #month, #day)
BEGIN
#target_col = #make_date (#year, #month, #day)
END;
```

パラメータ・ファイルでマクロ・ライブラリを使用する手順

マクロ・ライブラリを使用するには、パラメータ・ファイルの先頭で INCLUDE パラメータを使用します。次のサンプルの Replicat パラメータ・ファイルを参照してください。

例

```
INCLUDE /ggs/dirprm/mdatelib.mac
REPLICAT rep
ASSUMETARGETDEFS
USERID ggs, PASSWORD ggs123,
MAP fin.acct_tab, TARGET fin.account;
```

レポート・ファイルのリスト表示の抑止

パラメータ・ファイルに長いマクロ・ライブラリが含まれる場合、NOLIST パラメータを使用して、Extract または Replicat のレポート・ファイルで各マクロがリスト表示されることを抑止できます。リスト表示の有効化と無効化を切り替えるには、パラメータ・ファイル内またはマクロ・ライブラリ・ファイル内の任意の場所に LIST パラメータおよび NOLIST パラメータを配置します。次の例では、NOLIST によって、hugelib マクロ・ライブラリに含まれる各マクロのリスト表示を抑止します。INCLUDE 文の後に LIST を指定することで、レポート・ファイルを通常のリスト表示に戻します。

例

```
NOLIST
INCLUDE /ggs/dirprm/hugelib.mac
LIST
INCLUDE /ggs/dirprm/mdatelib.mac
REPLICAT REP
```

マクロ展開のトレース

CMDTRACE パラメータを使用して、マクロ展開をトレースできます。CMDTRACE を有効化すると、マクロ展開ステップが Extract または Replicat のレポート・ファイルに表示されます。

構文

```
CMDTRACE [ON | OFF | DETAIL]
```

条件 :

- ON は、トレースを有効化します。
- OFF は、トレースを無効化します。
- DETAIL は、マクロ展開の詳細表示を生成します。

次の例では、トレースが `#testmac` の起動前に有効化され、マクロの実行後に無効化されます。

```
REPLICAT REP
MACRO #testmac
BEGIN
COL1 = COL2,
COL3 = COL4,
END;
...
CMDTRACE ON
MAP test.table1, TARGET test.table2,
COLMAP (#testmac);
CMDTRACE OFF
```

ユーザー・イグジットを使用した Oracle GoldenGate 機能の拡張

ユーザー・イグジットは、C プログラミング・コードで記述して処理中にコールするカスタム・ルーチンです。ユーザー・イグジットによって、複雑さとリスクを最小限に抑えながら Extract プロセスと Replicat プロセスの機能を拡張およびカスタマイズできます。ユーザー・イグジットの使用により、本番プログラムを変更することなく、データベース・イベントの発生時にレスポンスすることができます。

ユーザー・イグジットを実装する場合

ユーザー・イグジットは、Oracle GoldenGate 内で使用できる列変換関数のかわりとして、またはそれらと組み合せて使用できます。ユーザー・イグジットは、データを 2 回 (データの抽出時に 1 回と変換の実行時に 1 回) 処理するかわりに、データの抽出時に 1 回のみ処理するため、組込み関数の代用として適しています。

ユーザー・イグジットは次の用途で実装できます。

- ある表から別の表へのマップ時における算術演算、データ変換または表検索の実行。
- レコード・アーカイブ関数のオフライン実装。
- 通常とは異なるデータベース・イベントに対するカスタム形式でのレスポンス (出力値に基づいて電子メール・メッセージまたは通知を送信するなど)。
- 合計値の蓄積および統計値の収集。
- レコードの操作。
- 無効なデータの修復。
- 更新前後のレコードにおける正味の差異の計算。

- 複雑な基準に基づいた抽出またはレプリケーションのためのレコードの受入れまたは拒否。
- 変換時のデータベースの正規化。

ユーザー・イグジットの作成

次の内容は、Windows および UNIX システムでユーザー・イグジットを作成する場合に役立ちます。ここに記載されているパラメータおよび関数の詳細は、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』を参照してください。

ユーザー・イグジットを作成する手順

1. C コードで、共有オブジェクト (UNIX システム) または DLL(Windows) を作成し、Extract または Replicat からコールするルーチンを作成またはエクスポートします。このルーチンは、Oracle GoldenGate とユーザー独自のルーチン間の通信ポイントになります。このルーチンに任意の名前を付けます。ルーチンでは、次の Oracle GoldenGate ユーザー・イグジット・パラメータを受け入れる必要があります。
   - EXIT_CALL_TYPE: 処理中にルーチンをコールする時点を示します。
   - EXIT_CALL_RESULT: ルーチンにレスポンスを提供します。
   - EXIT_PARAMS: ルーチンに情報を提供します。
2. ソース・コードで、Oracle GoldenGate のインストール・ディレクトリにある usrdecs.h ファイルをインクルードします。このファイルには、型定義、戻りステータス値、コールバック関数コードなど、多数の定義が含まれます。このファイルは変更しないでください。
3. 必要に応じて Oracle GoldenGate コールバック・ルーチンをユーザー・イグジットに含めます。コールバック・ルーチンは、レコードおよびアプリケーション・コンテキスト情報を取得して、データ・レコードの内容を変更します。コールバック・ルーチンを実装するには、共有オブジェクトで ERCALLBACK 関数を使用します。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関数コードに基づいて異なる動作をします。

構文

```
ERCALLBACK (<function_code>, <buffer>, <result_code>);
```

条件 :

- <function_code> は、コールバック・ルーチンによって実行される関数です。
- <buffer> は、指定した関数コードに関連付けられた事前定義構造を含むバッファへの void ポインタです。
- <result_code> は、コールバック・ルーチンによって実行される関数のステータスです。コールバック・ルーチンによって戻される結果コードは、コールバック関数が成功したかどうかを示します。

Windows システムでは、Extract および Replicat によって、ユーザー・イグジット・ルーチンからコールされる ERCALLBACK 関数がエクスポートされます。ユーザー・イグジットでは、適切な Windows API コールを使用して実行時にコールバック関数を明示的にロードする必要があります。

4. Extract または Replicat のパラメータ・ファイルに CUSEREXIT パラメータを含めます。このパラメータでは、共有オブジェクトまたは DLL の名前と、Extract または Replicat からコールされるエクスポート・ルーチンの名前を使用します。共有オブジェクトまたは DLL のフルパスを指定するか、オペレーティング・システムの標準検索機能を使用して共有オブジェクトの場所を特定できます。このパラメータでは、次の処理を実行するためのオプションも使用できます。

- パススルー・モードで稼働するデータ・ポンプでのユーザー・イグジットの使用
- 更新操作のための変更前イメージの取得
- 起動文字列 (起動パラメータなど) の指定

構文

```
CUSEREXIT <DLL or shared object name> <routine name>
[, PASSTHRU]
[, INCLUDEUPDATEBEFORES]
[, PARAMS "<startup string>"]
```

例 UNIX システムのパラメータ・ファイル構文の例 :

```
CUSEREXIT eruserexit.so MyUserExit
```

例 Windows システムのパラメータ・ファイル構文の例 :

```
CUSEREXIT eruserexit.dll MyUserExit
```

ユーザー・イグジット関数を使用する方法のサンプルの表示

Oracle GoldenGate では、Oracle GoldenGate のインストール・ディレクトリの UserExitExamples ディレクトリに次のサンプル・ユーザー・イグジット・ファイルがインストールされます。

- exitdemo.c は、ユーザー・イグジットを初期化し、特定のイグジット・ポイントでコールバックを発行して、データを変更する方法を示しています。このデモは、いずれかのデータベース・タイプに固有ではありません。
- exitdemo_passthru.c は、Extract データ・ポンプで CUSEREXIT パラメータの PASSTHRU オプションを使用する方法を示しています。
- exitdemo_more_recs.c は、同じ入力レコードを複数回使用して複数のターゲット・レコードを生成する方法の例を示しています。
- exitdemo_lob.c は、LOB データに対する読取りアクセスを取得する方法の例を示しています。
- exitdemo_pk_befores.c は、主キーの更新レコードの変更前および変更後イメージの一部と、通常の更新 (キー以外の更新) の変更前イメージにアクセスする方法を示しています。また、競合検出の手段として、Replicat パラメータ・ファイルの SQLEXEC を使用してターゲット行の値を取得する方法も示しています。ターゲットからフェッチされた値は、ユーザー・イグジットに取得された時点でターゲット・レコードとしてマップされます。

各ディレクトリには、.c ファイルに加え、makefile および readme.txt ファイルが含まれます。

ユーザー・イグジットのアップグレード

usrdecs.h ファイルは、新しい機能や構造変更などの拡張またはアップグレードが Oracle GoldenGate の新規リリースに追加された場合に、既存のユーザー・イグジットとの下位互換性に対応するためにバージョンが付けられています。usrdecs.h ファイルのバージョンは、Replicat または Extract の起動時にレポート・ファイルに出力されます。

ユーザー・イグジットの新しい機能を使用するには、独自のルーチンを再コンパイルして新しい usrdecs ファイルをインクルードする必要があります。新しい機能を使用しないルーチンは、再コンパイルする必要はありません。

Oracle GoldenGate イベント・マーカー・システムを使用したデータベース・イベントの起動

Oracle GoldenGate では、Oracle GoldenGate プロセスによって、(プロセスのデータソースに応じて) トランザクション・ログまたは証跡の*イベント・レコード*に基づいて定義済のアクションを実行できる*イベント・マーカー・システム*が提供されます。イベント・レコードは、アクションを実行するための特定のフィルタ基準に適合するレコードです。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。

イベント・マーカー・システムの使用方法

このシステムの使用方法の例には、プロセスの開始または停止、変換の実行、統計のレポートなどがあります。イベント・マーカー・システムは、次の用途で使用できます。

- SQLEXEC またはユーザー・イグジット関数を実行できる同期ポイントの確立
- データ検証スクリプトを実行するシェル・コマンドの実行
- 特定のアカウント番号検出時のトレースのアクティブ化
- ラグ履歴の取得
- バッチ・プロセスまたは日次レポート・プロシージャを開始するポイントの確立

イベント・マーカー機能は、データ変更のレプリケートではサポートされますが、初期ロードではサポートされません。

イベント・マーカー・システムを使用する手順

このシステムでは、次の入力要素が必要です。

1. アクションを起動する*イベント・レコード*を指定します。これを行うには、次のいずれかのパラメータ文に FILTER 句、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・ファイル内の TABLE 文
   - Replicat パラメータ・ファイル内の MAP 文
   - ソース表をターゲット表にマップせずに EVENTACTIONS アクションを実行できる、Replicat パラメータ・ファイル内の特別な TABLE 文
2. イベント・レコードを指定した同じ TABLE 文または MAP 文に、プロセスで実行されるアクションを指定する適切なオプションとともに EVENTACTIONS パラメータを指定します。

   これらのパラメータの詳細は、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』を参照してください。

複数のアクションの組合せ方

- 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て使用できます。目的を達成するために、2 つ以上のアクションを組み合せる必要が生じることがあります。

- 最初に EVENTACTIONS 文全体が解析され、その後で、指定されたオプションが優先順位に応じて実行されます。次のリストで Process the record より前に表示されているアクションは、レコードが証跡に書き込まれる前、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)。Process the record より後に表示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
  - (Process the record)
  - REPORT
  - ABORT
  - CHECKPOINT AFTER
  - FORCESTOP
  - STOP

次の例は、EVENTACTIONS オプションの組合せ方法を示しています。他の例は、288 ページの「イベント・マーカー・システムの使用方法のケース・スタディ」を参照してください。

例 次の例では、プロセスでチェックポイントを発行し、情報メッセージを記録し、トランザクション全体を無視して (何も処理されません)、レポートを生成します。

```
EVENTACTIONS (CP BEFORE, REPORT, LOG, IGNORE TRANSACTION)
```

例 次の例では、廃棄ファイルにイベント・レコードを書き込み、トランザクション全体を無視します。

```
EVENTACTIONS (DISCARD, IGNORE TRANS)
```

例 次の例では、情報メッセージを記録し、プロセスを正常に停止します。

```
EVENTACTIONS (LOG INFO, STOP)
```

例 次の例では、証跡ファイルをロールオーバーし、新規ファイルにイベント・レコードを書き込みません。

```
EVENTACTIONS (ROLLOVER, IGNORE)
```

イベント・レコード自体の処理の制御方法

イベント・レコード自体が通常の方法で処理されないようにするには、IGNORE オプションまたは DISCARD オプションを使用します。IGNORE および DISCARD は、レコード自体の前に評価されるので、レコードが処理されるのを防ぎます。これらのオプションがない場合、Extract によってレコードが証跡に書き込まれ、そのレコード内の操作は Replicat によってターゲット・データベースに適用されます。

イベント・アクションを起動するレコードがトランザクションに 2 つ以上含まれている可能性を考慮する必要があります。そのような場合、特定の EVENTACTIONS 指定が複数実行されることがあります。たとえば、有効なレコードが 2 つあり、ROLLOVER アクションが 2 件続けて起動されると、Extract では証跡が 2 回ロールオーバーされ、2 つのうちの 1 つが実質的に空になります。

構文

```
EVENTACTIONS (
[STOP | ABORT | FORCESTOP]
[IGNORE [TRANSACTION [INCLUDEVENT]]
[DISCARD]
[LOG [INFO | WARNING]]
[REPORT]
[ROLLOVER]
[SHELL <command>]
[TRACE <trace file> [TRANSACTION] [PURGE | APPEND]]
[CHECKPOINT [BEFORE | AFTER | BOTH]]
[, ...]
)
```

アクション	説明
STOP	指定されたイベント・レコードが検出された場合に、プロセスを正常に停止します。プロセスは、オープン・トランザクションが完了するのを待ってから停止します。Replicat によってグループ化またはバッチ化されたトランザクションの場合、現在のトランザクション・グループが適用されてから、プロセスは正常に停止します。このレコードがトランザクションの最後であることも示唆されている場合は、イベント・レコードの直後のレコードからプロセスを再起動します。 トランザクションがまだオープン状態のためプロセスを即時停止できない場合、メッセージがログに記録されます。ただし、長期間実行されているオープン・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、トランザクションの未コミット状態を通知する警告メッセージは表示されません。したがって、STOP イベントにかかわらず、プロセスが長期間実行されたままになることがあります。 STOP は、ABORT および FORCESTOP 以外の EVENTACTIONS オプションと組み合せることができます。
ABORT	指定されたイベント・レコードが検出された場合、オープン・トランザクションの有無にかかわらず、プロセスを即座に終了します。イベント・レコードは処理されません。致命的なエラーはログに書き込まれ、DISCARD も指定されている場合、イベント・レコードは破棄ファイルに書き込まれます。プロセスは起動時にリカバリされます。 ABORT と組み合せることができるのは、CHECKPOINT BEFORE、DISCARD、SHELL および REPORT のみです。

アクション	説明
FORCESTOP	指定されたイベント・レコードが検出されると、そのイベント・レコードがトランザクションの最後の操作であるか、またはトランザクションの唯一のレコードである場合にのみ、プロセスを正常に停止します。レコードは通常どおり書き込まれます。 長期間実行されているオープン・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、ABORT と同様に、警告メッセージがログに書き込まれ、プロセスが即座に終了します。この場合、起動時にリカバリが必要になる場合があります。長時間実行されているトランザクションの途中で FORCESTOP アクションが起動された場合、プロセスは警告メッセージなしに終了します。 FORCESTOP は、ABORT、STOP、CHECKPOINT AFTER および CHECKPOINT BOTH 以外の EVENTACTIONS オプションと組み合せることができます。ROLLOVER とともに使用すると、プロセスが正常に停止した場合にのみロールオーバーが発生します。
IGNORE [TRANSACTION [INCLUDEVENT]]	プロセスでは指定されたイベント・レコードがデフォルトで無視されます。警告や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示されます。 ◆ TRANSACTION を使用すると、イベントを起動したレコードを含むトランザクション全体が無視されます。TRANSACTION を使用する場合、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。トランザクションが無視されると、イベント・レコードもデフォルトで無視されます。TRANSACTION は、TRANS に短縮できます。 ◆ INCLUDEEVENT と TRANSACTION を組み合せて使用すると、イベント・レコードは証跡またはターゲットに伝播されますが、残りの関連するトランザクションは無視されます。 IGNORE は、ABORT および DISCARD 以外のすべての EVENTACTIONS オプションと組み合せることができます。
DISCARD	プロセスで次の処理が行われます。 ◆ 指定されたイベント・レコードを破棄ファイルに書き込みます。 ◆ Oracle GoldenGate 統計を更新して、レコードが破棄されたことを示します。 証跡内の次のレコードから処理が再開されます。このオプションを使用すると、DISCARDFILE パラメータに破棄ファイルの名前を指定できます。デフォルトでは、破棄ファイルは作成されません。 DISCARD は、IGNORE 以外のすべての 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
LOG [INFO | WARNING]	指定されたイベント・レコードが検出された場合に、プロセスでイベントがログに記録されます。メッセージはレポート・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 は、重大度の低い情報メッセージを示します。この設定がデフォルトです。 ◆ WARNING は、重大度の高い警告メッセージを示します。 LOG は、ABORT 以外のすべての 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、LOG は必要ありません (プロセスが終了する前に ABORT によって致命的なエラーがログに記録されるため)。
REPORT	指定されたイベント・レコードが検出された場合に、プロセスでレポート・ファイルが生成されます。これは、GGSCI で SEND コマンドに REPORT オプションを使用する場合と同じです。 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 (DISCARD、IGNORE または ABORT が使用されている場合を除く)、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。指定されたイベント・レコードが検出された場合に、Extract によって証跡が新しいファイルにロール・オーバーされます。ROLLOVER アクションは、Extract がイベント・レコードを証跡ファイルに書き込む前に発生するので、このレコードが新しいファイル内の最初にレコードになります (DISCARD、IGNORE または ABORT も使用されている場合を除く)。 ROLLOVER は、ABORT 以外のすべての EVENTACTIONS オプションと組み合せることができます。 注意 : 次の理由により、ROLLOVER と ABORT を組み合せることはできません。 ◆ ROLLOVER では、プロセスによるチェックポイントの書込みが行われません。 ◆ ROLLOVER は ABORT の前に発生します。 ROLLOVER チェックポイントがない場合、ABORT によって、Extract は再起動時に以前の証跡ファイル内にある以前のチェックポイントに移動します。その結果、ロールオーバーが取り消されます。

アクション	説明
`SHELL <command>`	指定されたイベント・レコードが検出された場合に、指定されたシェル・コマンドがプロセスで実行されます。 ◆ <command> には、発行されるシステム・コマンドまたはシェル・コマンドを指定します。 ◆ シェル・コマンドが成功すると、情報メッセージがレポート・ファイルとイベント・ログに書き込まれます。成功したかどうかは、UNIX シェル言語に従い、コマンドの終了ステータスに基づいて判断されます。この言語では、ゼロは成功を示します。 ◆ システム・コールに失敗すると、プロセスは致命的エラーで異常終了します。UNIX シェル言語では、ゼロ以外は失敗です。 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。

`TRACE <trace file>`
`[TRANSACTION]`
`[PURGE | APPEND]`
	指定されたイベント・レコードが検出された場合に、プロセスによってトレース情報がトレース・ファイルに書き込まれます。
	デフォルトで、トレースはプロセスが終了するまで有効です。トレース・レベルを設定するには、Oracle GoldenGate の TRACE または TRACE2 パラメータを使用します。
	◆ トレース・ファイルの名前を指定する <trace file> は、TRACE キーワードの直後に配置する必要があります。一意のトレース・ファイルを指定するか、またはスタンドアロンの TRACE または TRACE2 パラメータで指定されるデフォルトのトレース・ファイルを使用できます。
	EVENTACTIONS TRACE が使用されている他の TABLE 文または MAP 文で、同じトレース・ファイルを使用できます。同じトレース・ファイル名が指定されている複数の TABLE 文または MAP 文で、TRACE オプションの使用が徹底されていない場合は、このトレース・ファイルが含まれ、最後に解決された TABLE 文または MAP のオプションが優先されます。
	◆ TRANSACTION を使用する場合、トレースが有効なのは現在のトランザクションが終了するまでで、プロセスの終了時まででではありません。Replicat の場合、トランザクション境界は、Replicat によってグループ化またはバッチ化された通常のターゲット・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。
	◆ PURGE を使用して、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るか、または APPEND を使用して、既存のレコードの最後に新しいトレース・レコードを書き込みます。APPEND がデフォルトです。
	TRACE は、ABORT 以外のすべての EVENTACTIONS オプションと組み合せることができます。
	指定されたトレース・ファイルへのトレースを無効化するには、TRACE OFF <filename> オプションを指定して、GGSCI の SEND <process> コマンドを発行します。

アクション	説明
CHECKPOINT [BEFORE | AFTER | BOTH]	指定されたイベント・レコードが検出された場合に、プロセスでチェックポイントが書き込まれます。チェックポイント操作により、TABLE 文または MAP 文で定義されている処理に関するコンテキストが提供されます。このコンテキストには開始ポイントと終了ポイントがあり、これにより SQLEXEC およびユーザー・イグジットで実行される関数のマッピング用の同期化ポイントが与えられます。 ◆ BEFORE Extract プロセスの BEFORE では、Extract がイベント・レコードを証跡に書き込む前に、チェックポイントが書き込まれます。 Replicat プロセスの 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、チェックポイントが書き込まれます。 BEFORE では、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。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 を使用すると、イベント・レコードで開始される前のトランザクションがすべてコミットされていることが保証されます。 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 の AFTER では、Extract がイベント・レコードを証跡に書き込んだ後に、チェックポイントが書き込まれます。 Replicat の AFTER 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した後に、チェックポイントが書き込まれます。 AFTER では、チェックポイント・リクエストが勧告としてフラグされます (つまり、実質的な次の機会にのみチェックポイントが発行されます)。たとえば、イベント・レコードが複数のレコードからなる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 のデータ整合性モデルに従って、次のトランザクション境界で発生します。 CHECKPOINT AFTER は、ABORT 以外のすべての EVENTACTIONS オプションと組み合せることができます。 ◆ BOTH BOTH は BEFORE と AFTER を組み合せたものです。Extract または Replicat のプロセスでは、イベント・レコードが処理される前と後にチェックポイントが書き込まれます。 CHECKPOINT BOTH は、ABORT 以外のすべての 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。

イベント・マーカー・システムの使用方法のケース・スタディ

日次処理の起動

この例では、ソース・データベースの event_table という特別な表に実行される操作の取得を指定します。この表は、決められた時刻 (毎日午後 5:00 など) に挿入操作を受信するためにのみ存在します。Replicat は、この操作のトランザクション・レコードを受信すると、オペレータが日次処理ジョブを開始できるように正常に停止します。毎日 event_table 表に対して行われる挿入を使用することで、オペレータは、Replicat によって 5:00 までのすべてのコミット済トランザクションが適用されたことを確認します。IGNORE によって、Replicat は、ターゲット・データベースでの用途がないイベント・レコード自体を無視します。LOG INFO によって、Replicat は、操作に関する情報メッセージを記録します。

例

```
TABLE source.event_table, EVENTACTIONS (IGNORE, LOG INFO, STOP);
```

初期ロードから変更同期への移行の簡略化

イベント・アクションおよびイベント表を使用して、初期ロードから進行中の変更レプリケーションへの移行を支援できます。たとえば、データが移入された既存のソース表を Oracle GoldenGate 構成に追加する必要があるとします。この表をターゲットに作成してから、エクスポートおよびインポートを使用して 2 つの表を同期する必要があります。この例では、source.event_table というイベント表がソース・データベースに存在しており、Replicat の TABLE 文で指定されると仮定します。

ユーザーが新しいソース表で作業を継続できるように、そのソース表を Replicat のパラメータ・ファイルではなく Extract のパラメータ・ファイルに追加します。Extract は、この表から証跡へのデータの取得を開始し、データは証跡に格納されます。

エクスポート後にソースとターゲットの読取り一貫性が確保された時点で、ソースのイベント表にイベント・レコードが挿入され、その内容がターゲットに伝播されます。Replicat が (読取り一貫性ポイントを示す) イベント・レコードを受信すると、プロセスは、EVENTACTIONS STOP の指示に従って停止します。これにより、新規表が Replicat の MAP 文に追加されます。Replicat は、イベント・レコードのタイムスタンプからレプリケーションを開始するように設定できるため、HANDLECOLLISIONS パラメータを使用する必要がなくなります。証跡内の操作のうちイベント・レコードより前のものは、エクスポートで適用されていることが確認されているため、無視できます。

イベント・レコード自体は Replicat によって無視されますが、情報メッセージが記録されます。

例

```
TABLE source.event_table, EVENTACTIONS (IGNORE, LOG INFO, STOP);
```

データ異常値が検出された場合の処理の停止

この例では、ABORT を使用して、銀行レコードで異常値が検出された場合 (顧客が口座残高より多くの現金を引き出そうとした場合)、致命的エラーとともに Replicat を即座に終了します。この場合、ソース表は、ターゲットに対する実際のレプリケーションを目的とする Replicat の MAP 文でターゲット表にマップされます。ソース表には TABLE 文も使用され、ABORT アクションによって、Replicat は異常値をターゲット・データベースに適用する前に停止されます。ABORT は、レコードの処理に優先します。

例

```
MAP source.account, TARGET target.account;
TABLE source.account, FILTER (withdrawal > balance), EVENTACTIONS (ABORT);
```

ハートビート表およびロギングを使用したラグの分析

この例では、ログ・アクションの構成方法を示します。heartbeat 表は、ソース・データベースの現在の時刻で定期的に更新されます。heartbeat 表に対する更新は、Extract によって取得され、証跡に書き込まれます。ハートビート・レコードをイベント・レコードとして使用することで、Replicat は、FILTER 句を持つ 2 つの異なる MAP 文のラグの計算に基づいてメッセージを記録します。

最初の FILTER 句では、ラグが 60 秒を超え、120 秒未満の場合に情報メッセージが書き込まれます。2 番目の FILTER 句では、ラグが 120 秒を超えた場合に警告メッセージが記録されます。

この例では、ハートビート・レコードもターゲット・データベースの heartbeat 監査表に書き込まれます。この表は、履歴ラグ分析に使用できます。代替オプション (IGNORE または DISCARD) を使用すれば、ハートビート・レコードを無視または破棄できます。

注意 **同じソース・オブジェクトとターゲット・オブジェクトに対して重複する MAP 文が存在するため、ALLOWDUPTARGETMAPS を使用します。**

例 ALLOWDUPTARGETMAPS

```
MAP source.heartbeat, TARGET target.heartbeat, &
FILTER ( &
@DATEDIFF ("SS", hb_timestamp, @DATENOW()) > 60 AND &
@DATEDIFF ("SS", hb_timestamp, @DATENOW()) < 120), &
EVENTACTIONS (LOG INFO);

MAP source.heartbeat, TARGET target.heartbeat, &
FILTER (@DATEDIFF ("SS", hb_timestamp, @DATENOW()) > 120), &
EVENTACTIONS (LOG WARNING);
```

特定の注文番号のトレース

次の例では、特定の注文番号 (order_no = 1) に対する挿入操作を含む注文トランザクションのみを対象に Replicat のトレースを有効化します。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 パラメータによって、ターゲット表に対するソース表のマッピングを指定します。

例 MAP sales.order, TARGET rpt.order;

```
TABLE source.order,
FILTER (@GETENV ("GGHEADER", "OPTYPE") = "INSERT" AND order_no = 1), &
EVENTACTIONS (TRACE order_1.trc TRANSACTION);
```

バッチ・プロセスの実行

この例では、バッチ・プロセスを 1 か月に 1 回実行して、ソース・データベースから蓄積されたデータを消去します。トランザクション (通常はバッチ・トランザクション) の開始時に、レコードが特別な job 表に書き込まれ、バッチ・ジョブが開始したことが示されます。ターゲット・システムには削除されたレコードを反映する必要がないため、TRANSACTION を IGNORE とともに使用して、トランザクション全体を Extract で無視するように指定します。Extract 側の作業を無視することで、不要な証跡およびネットワーク・オーバーヘッドが排除されます。

注意 **論理バッチ削除が複数のより小さいバッチで構成される場合、それらのより小さいバッチごとに、トランザクションの最初のレコードとしてジョブ表に挿入する操作が必要です。**

例

```
TABLE source.job, FILTER (@streq(job_type = "HOUSEKEEPING")=1), &
EVENTACTIONS (IGNORE TRANSACTION);
```

結果となる操作を除く SQL 文のみの伝播

この例では、ソースとターゲットで異なる EVENTACTIONS 句を組み合せて使用し、SQL 文の結果となる操作ではなく、その SQL 文のみをレプリケートする方法を示します。この例の SQL 文は、INSERT INTO...SELECT トランザクションです。このようなトランザクションでは、伝播する必要のある大量の行が生成される可能性がありますが、この方法であれば、初期 SQL 文のみが伝播されるため、証跡およびネットワーク・オーバーヘッドを削減できます。各 SELECT 文は、すべてターゲットに対して実行されます。この構成では、データ整合性を保持するためにソース表とターゲット表を完全に同期する必要があります。

この構成を使用するため、statement 表にトランザクションの最初の操作 (イベント・レコードとなる INSERT INTO...SELECT) を移入します。

> **注意** **サイズの大きい SQL 文の場合、文を表の複数の列に書き込むことができます。たとえば、8 つの VARCHAR (4000) 列を使用して、最大 32KB の長さの SQL 文を格納できます。**

IGNORE TRANS INCLUDEEVENT があるため、Extract では、文の SELECT 部分に関連する後続の挿入はすべて無視されますが、SQL テキストを含むイベント・レコードは証跡に書き込まれます。TABLE 文の使用により、Replicat は、必要に応じて SQL テキスト列を連結する SQLEXEC 文にイベント・レコードを渡し、SELECT 副問合せの入力としてターゲット表を使用して INSERT INTO...SELECT 文を実行します。

例

Extract:

```
TABLE source.statement, EVENTACTIONS (IGNORE TRANS INCLUDEEVENT);
```

Replicat:

```
TABLE source.statement, SQLEXEC (<execute SQL statement>), &
EVENTACTIONS (INFO, IGNORE);
```

長時間実行トランザクション開始前の他のトランザクションのコミット

この EVENTACTIONS の使用によって、Replicat で処理されているすべてのオープン・トランザクションが、長時間実行トランザクションの開始前にターゲットにコミットされることが保証されます。この場合、Replicat によって、大規模トランザクションの作業開始前に強制的にチェックポイントを書き込みます。チェックポイントを強制することで、リカバリの可能性を長時間実行トランザクションのみに制限します。Replicat のチェックポイントによって、データベースに対する暗黙的コミットが実行されるため、未処理のロックが解放され、保留中の変更を他のセッションで認識できるようになります。

例

```
TABLE source.batch_table, EVENTACTIONS (CHECKPOINT BEFORE);
```

データ検証のためのシェル・スクリプトの実行

この例では、シェル・スクリプトを実行して、Replicat がテスト実行の最後のトランザクションを適用した後にデータ検証を行う別のスクリプトを実行します。ソースでは、イベント・レコードが source.event というイベント表に書き込まれます。レコードによって、値 COMPARE がイベント表の event_type 列に挿入されます。このレコードは、他のテスト・データの最後にレプリケートされます。Replicat パラメータ・ファイルの TABLE 文で、FILTER 句によってレコードを限定し、EVENTACTIONS 句の SHELL の指定によってシェル・スクリプト compare_db.sh を実行します。その後、FORCESTOP の指定によって Replicat は即座に停止します。

例

Extract:

```
TABLE src.*;
TABLE test.event;
```

Replicat:

```
MAP src.*, TARGET targ.*;
TABLE test.event, FILTER (@streq(event_type, "COMPARE")=1), &
EVENTACTIONS (SHELL "compare_db.sh", FORCESTOP);
```

第 19 章

Oracle GoldenGate 処理の監視

Oracle GoldenGate 監視ツールの概要

Oracle GoldenGate 処理を監視してプロセス・ステータス、統計およびイベントを表示するには、次のツールを使用します。

- GGSCI 情報コマンド
- ggserr.log ファイル (*エラー・ログ*)
- プロセス・レポート
- 廃棄ファイル
- Windows システムのイベント ビューアまたは UNIX システムの syslog によるオペレーティング・システム・レベルでのエラーの表示

GGSCI での情報コマンドの使用

処理情報を表示する主な方法は、GGSCI を使用することです。これらのコマンドの詳細は、『*Oracle GoldenGate Windows and UNIX リファレンス・ガイド*』を参照してください。

表 48　プロセス情報を表示するコマンド

コマンド	表示内容
INFO {EXTRACT | REPLICAT} <group> [DETAIL]	実行ステータス、チェックポイント、ラグ概算および環境情報
INFO MANAGER	実行ステータスおよびポート番号
INFO ALL	システム上のすべての Oracle GoldenGate プロセスに関する INFO 出力
STATS {EXTRACT | REPLICAT} <group>	処理された操作の統計
STATUS {EXTRACT | REPLICAT} <group>	実行ステータス (起動中、実行中、停止済、異常終了済)
STATUS MANAGER	実行ステータス
LAG {EXTRACT | REPLICAT} <group>	処理された最新レコードとデータソースのタイムスタンプとの間の待機誤差

表 48　プロセス情報を表示するコマンド (続き)

コマンド	表示内容
INFO {EXTTRAIL | RMTTRAIL} <path name>	関連プロセスの名前、最後に処理されたデータの位置、最大ファイル・サイズ
SEND MANAGER	実行ステータス、子プロセスに関する情報、ポート情報、証跡消去設定
SEND {EXTRACT | REPLICAT}	プロセスに応じて、メモリー・プール、ラグ、TCP 統計、長時間実行トランザクション、プロセス・ステータス、リカバリ進行状況などに関する情報が戻されます。
VIEW REPORT <group>	プロセス・レポートの内容
VIEW GGSEVT	Oracle GoldenGate エラー・ログの内容

<command> ER <wildcard>	<command> タイプに応じた次の情報 :
	INFO
	LAG
	SEND
	STATS
	STATUS
	
	<wildcard> は、影響を受けるプロセス・グループに応じたワイルドカードの指定です。次に例を示します。
	`INFO ER ext*`
	`STATS ER *`

Extract リカバリの監視

注意　**このトピックは、Oracle 以外のすべてのデータベース・タイプに適用されます。Oracle では、*制限リカバリ*と呼ばれる異なるリカバリ・メカニズムが使用されます。詳細は、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』の BR パラメータの説明を参照してください。**

長時間実行トランザクションがオープンしているときに Extract が異常終了すると、Extract の再起動時のリカバリに時間がかかることがあります。処理状態をリカバリするため、Extract は、必要に応じて以前のオンライン・ログとアーカイブ・ログを検索し、その長時間実行トランザクションの最初のログ・レコードを検出する必要があります。トランザクションの開始時点が古いほど、一般的にリカバリにかかる時間も長くなり、Extract は停止したように見えることがあります。

Extract のリカバリ状況が適切であることを確認するには、SEND EXTRACT コマンドを STATUS オプション付きで使用します。次のステータス記録のいずれかが表示されるため、リカバリの実行中に Extract がそのログ読取り位置を変更するのに応じて、作業の進行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まで復帰してリカバリを実行しています。
- In recovery[2]: Extract は、チェックポイントから証跡の最後に向かってリカバリを実行しています。
- Recovery complete: リカバリは終了し、通常の処理が再開されます。

ラグの監視

ラグ統計は、Oracle GoldenGate プロセスが、ビジネス・アプリケーションによって生成されたデータの量に後れを取らず適切に処理を進めているかどうかを示します。この情報によって、潜在的な問題を診断し、Oracle GoldenGate プロセスのパフォーマンスをチューニングしてソース・データベースとターゲット・データベース間の待機時間を最小化できます。Oracle GoldenGate をチューニングしてラグを最小化する方法の詳細は、『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。

ラグについて

Extract でのラグとは、(システム・クロックに基づいて)Extract によってレコードが処理された時刻と、データソースにおけるそのレコードのタイムスタンプとの間の差異 (秒単位) です。

Replicat でのラグとは、(システム・クロックに基づいて)Replicat によって最後のレコードが処理された時刻と、証跡におけるそのレコードのタイムスタンプとの間の差異 (秒単位) です。

ラグ統計を表示する手順

GGSCI で LAG コマンドまたは SEND コマンドを使用します。

構文

```
LAG {EXTRACT | REPLICAT | ER} {<group | wildcard>}
```

または

構文

```
SEND {EXTRACT | REPLICAT} {<group | wildcard>}, GETLAG
```

注意 INFO コマンドでもラグ統計は戻されますが、この統計は、処理中の現在のレコードではなく、チェックポイントが指定された最後のレコードから取得された統計です。この統計は、LAG や INFO と比較して正確性に劣ります。

図 20 Extract および Replicat のすべてのプロセスを対象とするサンプルのラグ統計

```
GGSCI (sysb) 13> lag er *

Sending GETLAG request to EXTRACT ORAEXT...
Last record lag: 1 seconds.
At EOF, no more records to process.

Sending GETLAG request to REPLICAT ORAREP...
No records yet processed.
At EOF, no more records to process.

Sending GETLAG request to REPLICAT REPORA...
Last record lag: 7 seconds.
At EOF, no more records to process.
```

ラグのレポート方法を制御する手順

LAGREPORTMINUTES パラメータまたは LAGREPORTHOURS パラメータを使用して、Manager で Extract および Replicat のラグをチェックする間隔を指定します。

LAGCRITICALSECONDS、LAGCRITICALMINUTES または LAGCRITICALHOURS の各パラメータを使用して、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強制的にエラー・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 プロセスと Replicat プロセスに影響します。

LAGINFOSECONDS、LAGINFOMINUTES、LAGINFOHOURS のいずれかのパラメータを使用して、ラグ情報をエラー・ログに記録する頻度を指定します。ラグが LAGCRITICAL パラメータで指定した値を超えると、そのラグは Manager によってクリティカルとしてレポートされます。それ以外の場合、ラグは情報メッセージとしてレポートされます。0 (ゼロ) の値は、LAGREPORTMINUTES パラメータまたは LAGREPORTHOURS パラメータで指定された頻度で強制的にメッセージを書き込みます。

処理量の監視

量統計は、Oracle GoldenGate プロセスによって処理されているデータ量と、そのデータ量が Oracle GoldenGate システムを通じて移動されている速度を示します。この情報によって、潜在的な問題を診断し、Oracle GoldenGate プロセスのパフォーマンスをチューニングできます。

量統計を表示する手順

構文

```
STATS {EXTRACT | REPLICAT | ER} {<group | wildcard>}
[TABLE {<name | wildcard>}]
```

図 21 1 つの表を対象とするサンプルの基本的な STATS EXTRACT

```
GGSCI (sysa) 32> stats extract oraext
Sending STATS request to EXTRACT ORAEXT...
Start of Statistics at 2011-01-08 16:46:38.
Output to c:\goldengate802\dirdat\xx:
Extracting from HR.EMPLOYEES to HR.EMPLOYEES:

*** Total statistics since 2011-01-08 16:35:05 ***
        Total inserts                                704.00
        Total updates                                  0.00
        Total deletes                                160.00
        Total discards                                 0.00
        Total operations                             864.00

*** Daily statistics since 2011-01-08 16:35:05 ***
        Total inserts                                704.00
        Total updates                                  0.00
        Total deletes                                160.00
        Total discards                                 0.00
        Total operations                             864.00

*** Hourly statistics since 2011-01-08 16:35:05 ***
        Total inserts                                704.00
        Total updates                                  0.00
        Total deletes                                160.00
        Total discards                                 0.00
        Total operations                             864.00
```

```
*** Latest statistics since 2011-01-08 16:35:05 ***
        Total inserts                              704.00
        Total updates                                0.00
        Total deletes                              160.00
        Total discards                               0.00
        Total operations                           864.00
```

処理速度を表示する手順

構文

```
STATS {EXTRACT | REPLICAT | ER} {<group | wildcard>},
REPORTRATE {HR | MIN | SEC}
```

図 22 REPORTRATE のサンプル出力

```
*** Latest statistics since 2011-01-08 16:35:05 ***
        Total inserts/hour:                        718.34
        Total updates/hour:                          0.00
        Total deletes/hour:                          0.00
        Total discards/hour:                         0.00
        Total operations/hour:                     718.34
```

起動後に各表で処理された操作のサマリーを表示する手順

構文

```
STATS {EXTRACT | REPLICAT | ER} {<group | wildcard>},
TOTALSONLY <table>
```

図 23 TOTALSONLY のサンプル出力

```
GGSCI (sysa) 37> stats extract oraext, totalsonly hr.departments

Sending STATS request to EXTRACT ORAEXT...
Start of Statistics at 2011-01-08 17:06:43.
Output to c:\goldengate802\dirdat\xx:
Cumulative totals for specified table(s):

*** Total statistics since 2011-01-08 16:35:05 ***
        Total inserts                              352.00
        Total updates                                0.00
        Total deletes                                0.00
        Total discards                               0.00
        Total operations                           352.00

*** Daily statistics since 2011-01-08 16:35:05 ***
        Total inserts                              352.00
        Total updates                                0.00
        Total deletes                                0.00
        Total discards                               0.00
        Total operations                           352.00

*** Hourly statistics since 2011-01-08 17:00:00 ***

        No database operations have been performed.
```

```
*** Latest statistics since 2011-01-08 16:35:05 ***
        Total inserts                                352.00
        Total updates                                  0.00
        Total deletes                                  0.00
        Total discards                                 0.00
        Total operations                             352.00

End of Statistics.
```

表示される統計のタイプを制限する手順

構文

```
STATS {EXTRACT | REPLICAT | ER} {<group | wildcard>},
{TOTAL | DAILY | HOURLY | LATEST}
```

図 24 LATEST のサンプル統計

```
GGSCI (sysa) 39> stats extract oraext, latest

Sending STATS request to EXTRACT ORAEXT...
Start of Statistics at 2011-01-08 17:18:23.
Output to c:\goldengate802\dirdat\xx:
Extracting from HR.EMPLOYEES to HR.EMPLOYEES:

*** Latest statistics since 2011-01-08 16:35:05 ***
        Total inserts                                704.00
        Total updates                                  0.00
        Total deletes                                160.00
        Total discards                                 0.00
        Total operations                             864.00

End of Statistics.
```

前のオプションで設定されたすべてのフィルタを消去する手順

構文

```
STATS {EXTRACT | REPLICAT | ER} {<group | wildcard>}, RESET
```

レポート・ファイルに仮統計を送信する手順

構文

```
SEND {EXTRACT | REPLICAT | ER} {<group | wildcard>}, REPORT
```

エラー・ログの使用

Oracle GoldenGate のエラー・ログを使用して、次の情報を表示できます。

- GGSCI コマンドの履歴
- 起動および停止した Oracle GoldenGate プロセス
- 実行された処理
- 発生したエラー
- 情報メッセージおよび警告メッセージ

エラー・ログにはイベントが発生順に記録されているため、エラーの (1 つ以上の) 原因を検出する場合に役立ちます。たとえば、次の情報を検索できます。

- ユーザーがプロセスを停止したこと
- プロセスが TCP/IP またはデータベース接続の確立に失敗したこと
- プロセスがファイルを開くことに失敗したこと

図 25 Oracle GoldenGate エラー・ログ (ggserr.log ファイル)

```
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 ).
2011-01-08 11:21:31  GGS INFO     302  GoldenGate Manager for Oracle, mgr.prm:
Manager started GGSCI process on port 7841.
2011-01-08 11:24:15  GGS INFO     301  GoldenGate Manager for Oracle, mgr.prm:
Command received from GUI (START GGSCI ).
2011-01-08 11:24:15  GGS INFO     302  GoldenGate Manager for Oracle, mgr.prm:
Manager started GGSCI process on port 7842.
2011-01-08 11:24:16  GGS INFO    399  GoldenGate Command Interpreter for Oracle:
GGSCI command (ggs): add extract extcust  tranlog, begin now.
2011-01-08 11:30:19  GGS INFO    399  GoldenGate Command Interpreter for Oracle:
GGSCI command (ggs): add rmttrail /home/ggs, extract ggs
```

エラー・ログを表示する手順

次のいずれかの方法を使用します。

- 標準のシェル・コマンドによる Oracle GoldenGate のルート・ディレクトリに含まれる ggserr.log ファイルの表示
- Oracle GoldenGate Director
- GGSCI の VIEW GGSEVT コマンド

構文

```
VIEW GGSEVT
```

エラー・ログをフィルタする手順

エラー・ログのサイズは、非常に大きくなる可能性がありますが、キーワードに基づいてその内容をフィルタできます。たとえば、次のフィルタではエラーのみが表示されます。

```
$ more ggserr.log | grep ERROR
```

エラー・ログのサイズは、Oracle GoldenGate の使用に従って継続的に増加するため、ファイル内の最も古いエントリは、アーカイブして削除することを検討してください。

注意 Collector プロセスは、ログのクリーンアップ後に UNIX システムのログに対するレポートを停止することがあります。レポートを再開するには、クリーンアップ後に Collector プロセスを再起動します。

プロセス・レポートの使用

プロセス・レポートを使用して、(プロセスに応じて) 次の情報を表示できます。

- 使用中のパラメータ
- 表および列マッピング

- データベース情報
- 実行時メッセージおよびエラー
- 処理された操作の数に関する実行時統計

Extract、Replicat および Manager のすべてのプロセスによって、各実行の終了時にレポート・ファイルが生成されます。このレポートは、実行中に発生した問題 (無効なマッピング構文、SQL エラー、接続エラーなど) を診断する場合に役立ちます。

図 26 Extract のサンプル・プロセス・レポート

```
***********************************************************************
** Running with the following parameters **
***********************************************************************
sourceisfile
userid ggs, password ********
rmthost sys1, mgrport 8040
rmtfile /home/ggsora/dirdat/tcustord.dat, purge
table tcustord;

Processing table TCUSTORD

***********************************************************************
** Run Time Statistics **
***********************************************************************
Report at 2011-01-13 11:07:36 (activity since 2011-01-13 11:07:31)

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

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

プロセス・レポートを表示する手順

次のいずれかの方法を使用します。

- 標準のシェル・コマンドによるテキスト・ファイルの表示
- Oracle GoldenGate Director
- GGSCI の VIEW REPORT コマンド

構文

```
VIEW REPORT {<group> | <file name> | MGR}
```

条件 :

- <group> は、デフォルト名 (関連グループの名前) を持つ Extract レポートまたは Replicat レポートを示します。
- <file name> は、指定されたパス名と一致する Extract または Replicat のレポート・ファイルを示します。これを使用する必要があるのは、デフォルト以外のレポート名が、グループの作成時に ADD EXTRACT コマンドまたは ADD REPLICAT コマンドの REPORT オプションを使用して割り当てられている場合です。
- MGR は、Manager プロセス・レポートを示します。

オペレーティング・システムで大 / 小文字が区別される場合、レポート名は大文字にします。デフォルトでは、レポートのファイル拡張子は.rptです(EXTORA.rptなど)。デフォルトの場所は、Oracle GoldenGateディレクトリの dirrpt サブディレクトリです。

プロセス・レポートの名前と場所を確認する手順

GGSCI で INFO コマンドを使用します。

構文

```
INFO <group>, DETAIL
```

プロセスがレポートを生成せずに異常終了した場合に情報を表示する手順

(GGSCI ではなく) オペレーティング・システムのコマンド・シェルからプロセスを実行し、端末に情報を送信します。

構文

```
<process> paramfile <path name>.prm
```

条件 :

- <process> は、Extract または Replicat です。
- paramfile <path name>.prm は、パラメータ・ファイルの完全修飾名です。

例

```
replicat paramfile /ggs/dirdat/repora.prm
```

プロセス・レポートの実行時統計のスケジュール

デフォルトでは、実行時統計は、各実行の終了時に 1 回のみレポートに書き込まれます。長時間の実行や継続的な実行では、オプション・パラメータを使用することで、各実行の終了を待機せずにこれらの統計を定期的に表示できます。

実行時統計をレポートするためのスケジュールを設定する手順

Extract または Replicat のパラメータ・ファイルで REPORT パラメータを使用して、レポートで実行時統計を生成する日時を指定します。

必要に応じてレポートに実行時統計を送信する手順

SEND EXTRACT コマンドまたは SEND REPLICAT コマンドを REPORT オプション付きで使用して、必要時に現在の実行時統計を表示します。

プロセス・レポートのレコード数の表示

REPORTCOUNT パラメータを使用して、Extract または Replicat が起動後に処理したトランザクション・レコードの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 が取得し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。

プロセス・レポートの管理

レポート・ファイルは、一度作成したら、処理の開始後に Oracle GoldenGate を適切に動作させるため、元の場所から移動しないでください。

プロセスが起動するたびに、Oracle GoldenGate では新しいレポート・ファイルが作成され、古いファイルは名前に順序番号が追加されてエージングされます。番号は、0(直前のファイル) から 9(最も古いファイル) まで増分されます。

プロセスは、最大 10 個のエージングされたレポートと 1 個のアクティブなレポートを保持できます。10 個目のレポートがエージングされると、新規レポートの作成時に最も古いレポートが削除されます。エージングされたレポート・ファイルについては、サービス・リクエストの解決で必要とされる場合に備えて、アーカイブ・スケジュールを設定してください。

図 27 Extract および Manager の現行レポートとエージングされたレポート

```
-rw-rw-rw-      1 ggs ggs       1193 Oct 11 14:59        MGR.rpt
-rw-rw-rw-      1 ggs ggs       3996 Oct 5  14:02        MGR0.rpt
-rw-rw-rw-      1 ggs ggs       4384 Oct 5  14:02        TCUST.rpt
-rw-rw-rw-      1 ggs ggs       1011 Sep 27 14:10        TCUST0.rpt
-rw-rw-rw-      1 ggs ggs       3184 Sep 27 14:10        TCUST1.rpt
-rw-rw-rw-      1 ggs ggs       2655 Sep 27 14:06        TCUST2.rpt
-rw-rw-rw-      1 ggs ggs       2655 Sep 27 14:04        TCUST3.rpt
-rw-rw-rw-      1 ggs ggs       2744 Sep 27 13:56        TCUST4.rpt
-rw-rw-rw-      1 ggs ggs       3571 Aug 29 14:27        TCUST5.rpt
```

Extract または Replicat レポート・ファイルを適度な大きさに抑える手順

REPORTROLLOVER パラメータを使用して、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・ファイルを強制的にエージングします。長時間の実行や継続的な実行では、エージング・スケジュールを設定して、アクティブなレポート・ファイルのサイズを制御します。この設定によって、アーカイブ・ルーチンに含めることができるアーカイブのセットの予測可能性が高まります。

Replicat レポートが SQL エラーで一杯になることを防止する手順

WARNRATE パラメータを使用して、プロセス・レポートおよびエラー・ログにレポートする前に任意のターゲット表で許容する SQL エラーの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。現在の環境でこれらのエラーを多く許容できる場合、WARNRATE を増加させることで、各ファイルのサイズを最小限に抑えることができます。

廃棄ファイルの使用

廃棄ファイルを使用して、失敗した Oracle GoldenGate 操作に関する情報を取得します。この情報は、データ・エラー (無効な列マッピングに関連するエラーなど) を解決する場合に役立ちます。

廃棄ファイルでは、次のような情報がレポートされます。

- データベースのエラー・メッセージ
- データソースまたは証跡ファイルの順序番号
- データソースまたは証跡ファイルのレコードの相対バイト・アドレス
- 破棄された操作の詳細 (DML 文の列値や DDL 文のテキストなど)

廃棄ファイルは、Extract または Replicat で使用できますが、再構成または適用できなかった操作を Replicat で記録する場合に最も役立ちます。

図 28 サンプルの廃棄ファイル

```
ORA-20017:  asta0009 6144935
ORA-06512: at "LON.STARTASTA0009_INSERT", line 31
ORA-04088: error during execution of trigger 'LON.STARTASTA0009_INSERT'

Operation failed at seqno 45 rba 12483311
Problem replicating PRODTAB.ASTA0009 to ASTA0009

Error occurred with insert record (target format)...
*
A_TIMESTAMP = 2011-01-15 13:18:32
RELA_PERSON_NR = 3618047
RELA_BEZART = 1
RELA_BEZCODE = 01
RELA_AZ_BAFL = 2819220
RELA_STEMPEL = 0
AKTION = I
OK = 1.0000
NOTOK = -1.0000
*
```

廃棄ファイルを使用する手順

Extract または Replicat のパラメータ・ファイルに DISCARDFILE パラメータを含めます。ファイルの名前を指定する必要があります。このパラメータには、最大ファイル・サイズ (これを超過した場合にプロセスを異常終了させる) を制御するオプションや、新しい内容で既存の内容を上書きするか、または新しい内容を既存の内容に追加するかを制御するオプションがあります。

構文

```
DISCARDFILE <file name> [, APPEND | PURGE] [, MAXBYTES <n> | MEGABYTES <n>]
```

注意 廃棄ファイルの手動メンテナンスの実行を避ける場合、PURGE オプションまたは APPEND オプションを使用します。それ以外の場合、各プロセス実行を開始する前に、異なる廃棄ファイル名を指定する必要があります (Oracle GoldenGate は既存の廃棄ファイルに書込みを行わないため)。

廃棄ファイルを表示する手順

次のいずれかの方法を使用します。

- 標準のシェル・コマンドによる名前別でのファイルの表示
- GGSCI の VIEW REPORT コマンド (入力として廃棄ファイル名を指定)

構文

```
VIEW REPORT <file name>
```

廃棄ファイルを管理する手順

DISCARDROLLOVER パラメータを使用して、廃棄ファイルのエージングのスケジュールを設定します。長時間の実行や継続的な実行では、エージング・スケジュールを設定して、廃棄ファイルが一杯になりプロセスが異常終了することを防止します。この設定によって、アーカイブ・ルーチンに含めることができるアーカイブのセットが予測可能になります。

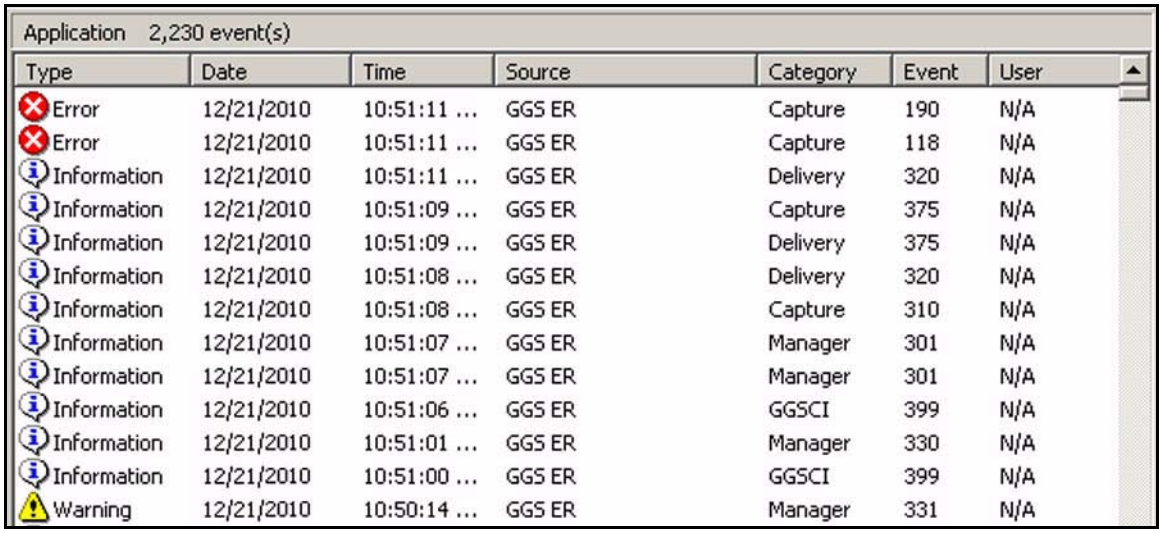
構文

```
DISCARDROLLOVER {AT <hh:mi> | ON <day of week> | AT <hh:mi> ON <day of week>}
```

システム・ログの使用

Oracle GoldenGate は、オペレーティング・システムのレベルで生成されたエラーを、Windows のイベント ビューアまたは UNIX および Linux の syslog に書き込みます。Oracle GoldenGate イベントの形式は、基本的に UNIX、Linux および Windows のシステム・ログで同じです。システム・ログに出力される Oracle GoldenGate エラーは、Oracle GoldenGate エラー・ログにも出力されます。

図 29 Windows イベント ビューアに表示される Oracle GoldenGate メッセージ



Application 2,230 event(s)

Type	Date	Time	Source	Category	Event	User
Error	12/21/2010	10:51:11 ...	GGS ER	Capture	190	N/A
Error	12/21/2010	10:51:11 ...	GGS ER	Capture	118	N/A
Information	12/21/2010	10:51:11 ...	GGS ER	Delivery	320	N/A
Information	12/21/2010	10:51:09 ...	GGS ER	Capture	375	N/A
Information	12/21/2010	10:51:09 ...	GGS ER	Delivery	375	N/A
Information	12/21/2010	10:51:08 ...	GGS ER	Delivery	320	N/A
Information	12/21/2010	10:51:08 ...	GGS ER	Capture	310	N/A
Information	12/21/2010	10:51:07 ...	GGS ER	Manager	301	N/A
Information	12/21/2010	10:51:07 ...	GGS ER	Manager	301	N/A
Information	12/21/2010	10:51:06 ...	GGS ER	GGSCI	399	N/A
Information	12/21/2010	10:51:01 ...	GGS ER	Manager	330	N/A
Information	12/21/2010	10:51:00 ...	GGS ER	GGSCI	399	N/A
Warning	12/21/2010	10:50:14 ...	GGS ER	Manager	331	N/A

UNIX および Linux の場合、syslog に対する Oracle GoldenGate メッセージ機能は、デフォルトで有効です。Windows の場合、イベント ビューアに対する Oracle GoldenGate メッセージ機能は、Oracle GoldenGate メッセージ DLL を登録してインストールする必要があります。

Windows で Oracle GoldenGate メッセージ機能を登録する手順

1. addevents オプションを指定して install プログラムを実行します。これによって、一般的なメッセージの記録が有効化されます。
2. (オプション) より詳細な Windows メッセージを取得するには、install 実行の前または後に、Oracle GoldenGate ディレクトリから SYSTEM32 ディレクトリに category.dll ライブラリと ggsmsg.dll ライブラリをコピーします。詳細メッセージには、Oracle GoldenGate のユーザー名とプロセス、パラメータ・ファイルの名前およびエラー・テキストが含まれます。

注意 Windows のイベント・メッセージ機能は、Oracle GoldenGate のインストール時にインストールされている可能性があります。install の実行の詳細は、使用中のデータベースに対応する Oracle GoldenGate のインストレーション・ガイドを参照してください。

Windows および UNIX で Oracle GoldenGate メッセージ機能をフィルタする手順

SYSLOG パラメータを使用して、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。次の処理を実行できます。

- すべての Oracle GoldenGate メッセージの記録
- すべての Oracle GoldenGate メッセージの抑止
- 情報、警告、エラーのいずれかのメッセージを記録するか、これらのタイプの任意の組合せを記録するためのフィルタ処理

SYSLOG は、GLOBALS または Manager パラメータ (あるいはその両方) として使用できます。GLOBALS パラメータ・ファイルに指定すると、このパラメータでシステムのすべての Oracle GoldenGate プロセスを対象にメッセージ・フィルタリングを制御できます。Manager パラメータ・ファイルに指定すると、このパラメータで Manager プロセスのみを対象にメッセージ・フィルタリングを制御できます。GLOBALS と Manager の両方のパラメータ・ファイルで使用すると、Manager プロセスに関しては、Manager 設定が GLOBALS 設定に優先します。これにより、Manager と他のすべての Oracle GoldenGate プロセスで別個の設定を使用できます。

時間の差異の調整

ソース・システムとターゲット・システム間の時間の差異に対応するには、Extract パラメータ・ファイルで TCPSOURCETIMER パラメータを使用します。このパラメータにより、レプリケートされたレコードのタイムスタンプがレポート目的で調整されるため、同期ラグの解析が容易になります。

NonStop システムへのイベント・メッセージの送信

Windows または UNIX システムの Collector プロセスおよび Replicat プロセスで作成されたイベント・メッセージは、取得して NonStop システムの EMS に送信できます。この機能によって、複数のプラットフォームにわたる Oracle GoldenGate メッセージの集約表示が可能になります。この機能を使用するには、次の 2 つの手順を実行します。

- Windows または UNIX システムで EMS クライアントを実行します。
- NonStop システムで Collector プロセスを起動します。

Windows または UNIX システムでの EMSCLNT の実行

EMSCLNT ユーティリティによって、Windows または UNIX システムで生成された Oracle GoldenGate イベント・メッセージを取得し、それらのメッセージを NonStop システムの TCP/IP Collector プロセスに送信します。EMSCLNT は、指定されたエラー・ログを読み取り、別のメッセージの送信を待機しながら無制限に実行されます。

Windows または UNIX システムの Oracle GoldenGate ディレクトリから emsclnt を実行するには、次の構文を使用します。

```
emsclnt -h <hostname> | <IP address>
-p <port number>
-f <filename>
-c <Collector>
```

条件 :

- -h <hostname> | <IP address> は、EMS メッセージの送信先となる NonStop Server の名前または IP アドレスです。
- -p <port number> は、NonStop Collector プロセスのポート番号です。
- -f <filename> は、エラー・メッセージの配信元となるローカル・ファイルの名前です。ファイルが Oracle GoldenGate ディレクトリ以外の場所に存在する場合、フルパス名を使用してください。
- -c <Collector> は、このクライアントの EMS Collector です。

例 次の Windows の例では、DOS プロンプトからコマンドを実行し、エラー・メッセージのファイル d:\ggs\ggserr.log を読み取ります。エラー・メッセージは、ポート 9876 でリスニングしている NonStop ホスト ggs2 の Collector に送信されます。NonStop の Collector プロセスは、フォーマットされたメッセージを EMS Collector $0 に書き込みます。

```
> emsclnt h ggs2 p 9876 f d:\ggs\ggserr.log c $0
```

例 次の UNIX の例では、エラー・メッセージのファイル ggserr.log を読み取ります。エラー・メッセージは、ポート 7850 でリスニングしている IP アドレスが 13.232.123.89 の NonStop Server の Collector に送信されます。NonStop の Collector は、フォーマットされたメッセージを EMS Collector $0 に書き込みます。

```
emsclnt h 13.232.123.89 p 7850 f ggserr.log c '$0'
```

注意 UNIX ではドル記号は変数を示すため、$0 は一重引用符で囲む必要があります。

NonStop での Collector の実行

NonStop システムの Collector(このプラットフォームでは Server-Collector と呼ばれます) は、EMS メッセージを収集して配信します。Collector を起動するには、server プログラムを実行します。Windows または UNIX システムで実行する EMSCLNT プロセスごとに、1 つの server プロセスを NonStop システムで起動します。

たとえば、次の例では、server を実行してメッセージを $DATA1.GGSERRS.SERVLOG に出力します。

```
> ASSIGN STDERR, $DATA1.GGSERRS.SERVLOG
> RUN SERVER /NOWAIT/ p 7880
```

NonStop プラットフォームで Server-Collector プロセスを実行する方法の詳細は、『Oracle GoldenGate *HP NonStop リファレンス・ガイド*』を参照してください。

監視およびチューニングに関するヘルプの取得

Oracle GoldenGate を監視、チューニングおよびトラブルシューティングする方法の詳細は、『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。

第 20 章

管理操作の実行

管理操作の概要

この章では、レプリケーション環境をアクティブにし、データ変更を処理しながら、アプリケーション、システムおよび Oracle GoldenGate を変更する手順について説明します。手順の内容は次のとおりです。

- アプリケーション・パッチの実行
- プロセス・グループの追加
- トランザクション・ログの初期化
- システムの停止
- データベース属性の変更
- 証跡ファイルのサイズの変更

アプリケーション・パッチの実行

通常、アプリケーション・パッチおよびアプリケーション・アップグレードでは、新規オブジェクトの追加や既存オブジェクトの変更などを伴う DDL が実行されます。パッチまたはアップグレードによって実行される DDL をレプリケートするには、Oracle GoldenGate を使用するか、またはその手順をソースとターゲットの両方で手動で実行できます。

Oracle GoldenGate を使用してパッチの DDL をレプリケートする手順

Oracle GoldenGate を使用してソースで実行されるパッチからターゲットに DDL をレプリケートする場合、データ・レプリケーションを停止する必要はありません。この目的で Oracle GoldenGate を使用するには、141 ページの第 14 章を参照してください。その場合、最初に Oracle GoldenGate DDL 環境について学習して構成する必要があるため、事前に計画を立ててください。DDL 環境を整えておけば、将来パッチをレプリケートするのが容易になります。

この方法を使用する場合の考慮事項は、次のとおりです。

1. アプリケーションのパッチまたはアップグレードによって追加される新規オブジェクトをデータ・レプリケーションに含める場合、必ずそれらを TABLE 文および MAP 文に追加します。この手順は、315 ページを参照してください。
2. アプリケーションのパッチまたはアップグレードによってトリガーまたはカスケード削除制約がインストールされる場合、ターゲットで実行される DML と、ソースのトリガーまたはカスケード削除からレプリケートされる同じ DDL との間で衝突が発生しないように、ターゲットでそれらのオブジェクトを無効化します。

ソースおよびターゲットでパッチを手動で適用する手順

1. ソース・データベースへのアクセスを停止します。
2. Extract がトランザクション・ログに残っているトランザクション・データの取得を終了するまで待機します。Extract の終了を確認するには、EOF に到達したことを示すメッセージが戻されるまで GGSCI で次のコマンドを発行します。

   ```
   SEND EXTRACT <group> GETLAG
   ```

3. Extract を停止します。

   ```
   STOP EXTRACT <group>
   ```

4. ソースに対するパッチの適用を開始します。
5. データ・ポンプ (使用している場合) および Replicat がそれぞれの証跡に含まれるデータの処理を終了するまで待機します。それぞれの終了を確認するには、EOF に到達したことを示すメッセージが戻されるまで次のコマンドを使用します。

   ```
   SEND EXTRACT <group> GETLAG
   SEND REPLICAT <group> GETLAG
   ```

6. データ・ポンプおよび Replicat を停止します。

   ```
   STOP EXTRACT <group>
   STOP REPLICAT <group>
   ```

   この時点で、ソースからレプリケートされたトランザクション変更はすべてターゲットに適用されたため、ソースとターゲットのデータは同一となります。

7. ターゲットにパッチを適用します。
8. パッチによってソース表およびターゲット表の定義が変更された場合、影響を受けたソース表に対して DEFGEN ユーティリティを実行し、更新後のソース定義を生成します。ターゲット・システムにある*既存*のソース定義ファイルで、各表の古い定義を新しい定義で置き換えます。
9. ユーザー・アクティビティの取得を再開する準備が整ったら、Oracle GoldenGate プロセスを起動します。

プロセス・グループの追加

作業を開始する前に

ここでの手順では、すでにアクティブである構成に対してプロセス・グループを追加します。各手順は、Oracle GoldenGate の操作経験のあるユーザーによって実行される必要があります。これらの手順では、プロセスを少しの間停止してパラメータ・ファイルを再構成します。これらの手順を実行するユーザーには、次のことが要求されます。

- Oracle GoldenGate 構成の基本コンポーネントに関する知識があること
- Oracle GoldenGate のパラメータおよびコマンドを理解していること
- GGSCI にアクセスしてグループおよびパラメータ・ファイルを作成できること
- 特定の状況で使用するパラメータを把握していること

各手順の詳細は、次の項を参照してください。

- アクティブな構成へのパラレル Extract グループの追加
- アクティブな構成へのデータ・ポンプの追加
- アクティブな構成へのパラレル Replicat グループの追加

アクティブな構成へのパラレル Extract グループの追加

この手順では、既存の Extract グループにパラレルな新規 Extract グループを追加します。また、データ・ポンプ・グループ (適用する場合) および Replicat グループを含めて、新規 Extract グループで取得されたデータを伝播する方法についても説明します。

手順はソース・システムとターゲット・システムで実行します。

1. この手順を完了する前にオンライン・ログを再利用する場合、アーカイブ・トランザクション・ログが使用できることを確認します。
2. 新規 Extract グループの名前を選択します。
3. データ・ポンプを使用するかどうかを決定します。
4. ソース・システムで、GGSCI を実行します。
5. 新規 Extract グループのパラメータ・ファイルを作成します。

   ```
   EDIT PARAMS <group>
   ```

   注意 **元のパラメータ・ファイルをコピーしてこのグループで使用できますが、必ず EXTRACT グループ名を変更し、この新規グループに適用される他のすべての関連パラメータを変更してください。**

6. パラメータ・ファイルに次のパラメータを含めます。
   - 新規グループを指定する EXTRACT パラメータ。
   - 適切なデータベース・ログイン・パラメータ。
   - 現在の構成に応じた他の適切な Extract パラメータ。
   - ローカル証跡を参照する EXTTRAIL パラメータ (データ・ポンプを追加する場合) ***または*** RMTTRAIL パラメータ (データ・ポンプを追加しない場合)。
   - RMTHOST パラメータ (この Extract で直接リモート証跡に書込みを行う場合)。
   - 新規グループで処理する表に対応する 1 つ以上の TABLE 文 (および適切な場合は TABLEEXCLUDE)。
7. ファイルを保存して閉じます。
8. 元の Extract のパラメータ・ファイルを編集して、新規グループに移動される表の TABLE 文を削除します。または、ワイルドカードを使用している場合、TABLEEXCLUDE パラメータを追加してワイルドカードの指定からそれらの表を除外します。

9. 新規グループに移動された表をロックし、ロックが適用された時点のタイムスタンプを記録します。Oracle 表では、次のスクリプトを実行できます (処理の終了後にロックは解放されます)。

```
-- temp_lock.sql
-- use this script to temporary lock a table in order to
-- get a timestamp

lock table &schema .&table_name in EXCLUSIVE mode;
SELECT TO_CHAR(sysdate,'MM/DD/YYYY HH24:MI:SS') "Date" FROM dual;
commit;
```

10. 前の手順でスクリプトを使用しなかった場合、1 つ以上の表のロックを解放します。

11. 古い Extract グループと既存のすべてのデータ・ポンプを停止します。

```
STOP EXTRACT <group>
```

12. 新規 Extract グループを追加して、前に記録したタイムスタンプの時点から起動するように構成します。

```
ADD EXTRACT <group>, TRANLOG, BEGIN <YYYY/MM/DD HH:MI:SS:CCCCCC>
```

13. 新規 Extract グループの証跡を追加します。

```
ADD {EXTTRAIL | RMTTRAIL} <trail name>, EXTRACT <group>
```

条件 :

- EXTTRAIL ではローカル証跡が作成されます。このオプションは、新規 Extract グループと組み合せて使用するデータ・ポンプを作成する場合に指定します。パラメータ・ファイルの EXTTRAIL で指定されている証跡を指定します。証跡を作成したら、「ローカル・データ・ポンプを追加する手順」に進んでください。
- RMTTRAIL ではリモート証跡が作成されます。このオプションは、データ・ポンプを使用しない場合に指定します。パラメータ・ファイルの RMTTRAIL で指定されている証跡を指定します。証跡を作成したら、「リモート Replicat を追加する手順」に進んでください。

相対名またはフルパス名を指定できます。例 :

```
ADD RMTTRAIL dirdat/rt, EXTRACT primary
ADD EXTTRAIL c:\ogg\dirdat\lt, EXTRACT primary
```

ローカル・データ・ポンプを追加する手順

1. ソース・システムで、データソースとして EXTTRAIL 証跡を使用し、データ・ポンプ Extract グループを追加します。

```
ADD EXTRACT <group>, EXTTRAILSOURCE <trail name>
```

次に例を示します。

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

2. データ・ポンプのパラメータ・ファイルを作成します。

```
EDIT PARAMS <group>
```

3. パラメータ・ファイルに現在の構成に応じた適切な Extract パラメータを含め、さらに次のパラメータを含めます。
   - ターゲット・システムを参照する RMTHOST パラメータ。
   - 新規リモート証跡 (この後で指定) を参照する RMTTRAIL パラメータ。
   - このデータ・ポンプで処理する表に対応する 1 つ以上の TABLE パラメータ。

   注意　データ・ポンプでデータを処理するが、フィルタリング、マッピングまたは変換を実行しない場合、PASSTHRU パラメータを含めてデータベース参照のオーバーヘッドを回避できます。データベース認証パラメータも省略できます。

4. ソース・システムの GGSCI で、データ・ポンプのリモート証跡を追加します。パラメータ・ファイルの RMTTRAIL で指定されている証跡名を使用します。

   ```
   ADD RMTTRAIL <trail name>, EXTRACT <group>
   ```

   次に例を示します。

   ```
   ADD RMTTRAIL dirdat/rt, EXTRACT pump
   ```

5. 「リモート Replicat を追加する手順」の指示に従います。

リモート Replicat を追加する手順

1. ターゲット・システムの GGSCI で、リモート証跡を読み取る Replicat グループを追加します。EXTTRAIL には、RMTTRAIL Extract パラメータおよび ADD RMTTRAIL コマンドと同じ証跡を指定します。

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

   次に例を示します。

   ```
   ADD REPLICAT rep, EXTTRAIL /home/ggs/dirdat/rt
   ```

2. この Replicat グループのパラメータ・ファイルを作成します。1 つ以上の MAP 文を使用して、新規プライマリ Extract およびデータ・ポンプ (使用する場合) で指定した表と同じ表を指定します。
3. ソース・システムで、Extract グループおよびデータ・ポンプを起動します。

   ```
   STOP EXTRACT <group>
   START EXTRACT <group>
   ```

4. ターゲット・システムで、新規 Replicat グループを起動します。

   ```
   START REPLICAT <group>
   ```

アクティブな構成へのデータ・ポンプの追加

この手順では、ソース・システムのアクティブなプライマリ Extract グループにデータ・ポンプ Extract グループを追加します。これにより、次の変更が発生します。

- プライマリ Extract は、ローカル証跡に書込みを行います。
- データ・ポンプは、古い証跡のデータがターゲットに適用された後に、新しいリモート証跡に書込みを行います。
- 古い Replicat グループは、新しい Replicat グループで置き換えられます。

手順はソース・システムとターゲット・システムで実行します。

1. ソース・システムで、GGSCI を実行します。
2. <group> にプライマリ Extract グループの名前を使用してローカル証跡を追加します。

   ```
   ADD EXTTRAIL <trail name>, EXTRACT <group>
   ```

   次に例を示します。

   ```
   ADD EXTTRAIL dirdat\lt, EXTRACT primary
   ```

3. プライマリ Extract グループのパラメータ・ファイルを開き、作成したローカル証跡を参照する EXTTRAIL パラメータで RMTTRAIL パラメータを置き換えます。

   ```
   EDIT PARAMS <group>
   ```

   EXTTRAIL パラメータの例 :

   ```
   EXTTRAIL dirdat\lt
   ```

4. RMTHOST パラメータを削除します。
5. ファイルを保存して閉じます。
6. データソースとしてステップ 2 で指定した証跡を使用し、新規データ・ポンプ Extract グループを追加します。

   ```
   ADD EXTRACT <group>, EXTTRAILSOURCE <trail name>
   ```

   次に例を示します。

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

7. 新規データ・ポンプのパラメータ・ファイルを作成します。

   ```
   EDIT PARAMS <group>
   ```

8. パラメータ・ファイルに現在の構成に応じた適切な Extract パラメータを含め、さらに次のパラメータを含めます。
   - このデータ・ポンプで処理する表に対応する 1 つ以上の TABLE パラメータ。
   - ターゲット・システムを参照する RMTHOST パラメータ。
   - 新規リモート証跡 (この後で作成) を参照する RMTTRAIL パラメータ。

     注意　データ・ポンプでデータを処理するが、フィルタリング、マッピングまたは変換を実行しない場合、PASSTHRU パラメータを含めてデータベース参照のオーバーヘッドを回避できます。データベース認証パラメータも省略できます。

9. ソース・システムの GGSCI で、データ・ポンプのリモート証跡を追加します。データ・ポンプのパラメータ・ファイルの RMTTRAIL で指定されている証跡名を使用し、EXTRACT のデータ・ポンプのグループ名を指定します。

   ```
   ADD RMTTRAIL <trail name>, EXTRACT <group>
   ```

   次に例を示します。

   ```
   ADD RMTTRAIL dirdat/rt, EXTRACT pump
   ```

   注意　このコマンドによって、証跡名が Extract グループにバインドされますが、実際の証跡は作成されません。証跡ファイルは、処理が開始した時点で作成されます。

10. ターゲット・システムで、GGSCI を実行します。
11. 新規 Replicat グループを追加してリモート証跡にリンクします。

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

次に例を示します。

```
ADD REPLICAT rep, EXTTRAIL dirdat/rt
```

12. この Replicat グループのパラメータ・ファイルを作成します。元の Replicat グループからパラメータ・ファイルをコピーできますが、必ず REPLICAT パラメータを新規グループ名に変更してください。
13. ソース・システムで、追加したパラメータの変更を反映するため、プライマリ Extract グループを一度停止してから再起動します。

```
STOP EXTRACT <group>
START EXTRACT <group>
```

14. ソース・システムで、データ・ポンプを起動します。

```
START EXTRACT <group>
```

15. ターゲット・システムで、EOF に到達して処理するレコードがなくなったことを示すメッセージがレポートされるまで、古い Replicat に対して LAG REPLICAT コマンドを発行し続けます。

```
LAG REPLICAT <group>
```

16. 古い Replicat グループを停止します。

```
STOP REPLICAT <group>
```

17. 古い Replicat グループのチェックポイント表を使用する場合、GGSCI からデータベースにログインします。

```
DBLOGIN [SOURCEDB <dsn>,] [USERID <user>[, PASSWORD <password>]]
```

18. 古い Replicat グループを削除します。

```
DELETE REPLICAT <group>
```

19. 新規 Replicat グループを起動します。

```
START REPLICAT <group>
```

注意 **古いリモート証跡は削除しないでください (後からサポート・ケースで必要となる場合などに備えるため)。古い証跡は、必要に応じて別の場所に移動できます。**

アクティブな構成へのパラレル Replicat グループの追加

この手順では、既存の Replicat グループにパラレルな新規 Replicat グループを追加します。新規 Replicat は、元の Replicat と同じ証跡から読取りを行います。

手順はソース・システムとターゲット・システムで実行します。

1. 新規グループの名前を選択します。
2. ターゲット・システムで、GGSCI を実行します。

3. 新規 Replicat グループのパラメータ・ファイルを作成します。

```
EDIT PARAMS <group>
```

注意 元のパラメータ・ファイルをコピーしてこのグループで使用できますが、必ず REPLICAT グループ名を変更し、この新規グループに適用される他のすべての関連パラメータを変更してください。

4. MAP 文を追加して (またはコピーした文を編集して)、このグループに移動する表を指定します。
5. パラメータ・ファイルを保存して閉じます。
6. ソース・システムで、GGSCI を実行します。
7. Extract グループを停止します。

```
STOP EXTRACT <group>
```

8. ターゲット・システムで、Replicat の古いパラメータ・ファイルを編集して、新規 Replicat グループに移動した表を指定している MAP 文を削除します。この Replicat で処理を続ける MAP 文のみを残します。
9. ファイルを保存して閉じます。
10. EOF に到達して処理するレコードがなくなったことを示すメッセージがレポートされるまで、古い Replicat グループに対して LAG REPLICAT コマンドを発行し続けます。

```
LAG REPLICAT <group>
```

11. 古い Replicat グループを停止します。

```
STOP REPLICAT <group>
```

12. 新規 Replicat グループを追加します。EXTTRAIL では、この Replicat が読み取る証跡を指定します。

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

次に例を示します。

```
ADD REPLICAT rep, EXTTRAIL dirdat/rt
```

13. ソース・システムで、Extract グループを起動します。

```
START EXTRACT <group>
```

14. ターゲット・システムで、古い Replicat グループを起動します。

```
START REPLICAT <group>
```

15. 新規 Replicat グループを起動します。

```
START REPLICAT <group>
```

トランザクション・ログの初期化

トランザクション・ログを初期化する場合、最初にすべてのデータを Oracle GoldenGate によって処理し、次に Extract グループとその関連証跡を削除してから再度追加する必要があります。

1. アプリケーションによるデータベースへのアクセスを停止します。これにより、トランザクション・データの記録が停止されます。
2. GGSCI を実行し、プライマリ Extract グループに対して SEND EXTRACT コマンドを LOGEND オプション付きで発行します。このコマンドで、Extract がトランザクション・ログに残っているレコードの処理を終了したかどうかを問い合せます。

   ```
   SEND EXTRACT <group name> LOGEND
   ```

3. 処理するレコードがなくなったことを示す YES ステータスが戻されるまで、コマンドを発行し続けます。
4. ターゲット・システムで GGSCI を実行し、SEND REPLICAT コマンドを STATUS オプション付きで発行します。このコマンドで、Replicat が証跡に残っているデータの処理を終了したかどうかを問い合せます。

   ```
   SEND REPLICAT <group name> STATUS
   ```

5. 現在のトランザクションで 0 (ゼロ) レコードと示されるまで、コマンドを発行し続けます。次に例を示します。

   ```
   Sending STATUS request to REPLICAT REPSTAB...
   Current status:
     Seqno 0, Rba 9035
     0 records in current transaction.
   ```

6. プライマリ Extract グループ、データ・ポンプ (使用している場合) および Replicat グループを停止します。

   ```
   STOP EXTRACT <group name>
   STOP EXTRACT <pump name>
   STOP REPLICAT <group name>
   ```

7. Extract、データ・ポンプおよび Replicat グループを削除します。

   ```
   DELETE EXTRACT <group name>
   DELETE EXTRACT <pump name>
   DELETE REPLICAT <group name>
   ```

8. 標準のオペレーティング・システム・コマンドを使用して、証跡ファイルを削除します。
9. データベースを停止します。
10. データベースを初期化して再起動します。
11. プライマリ Extract グループを再作成します。

    ```
    ADD EXTRACT <group name> TRANLOG, BEGIN NOW
    ```

12. ローカル証跡を再作成します (使用する場合)。

    ```
    ADD EXTTRAIL <trail name>, EXTRACT <group name>
    ```

13. データ・ポンプを再作成します (使用する場合)。

```
ADD EXTRACT <pump name>, EXTTRAILSOURCE <local trail name>
```

14. リモート証跡を再作成します。

```
ADD RMTTRAIL <trail name>, EXTRACT <pump name>
```

15. Replicat グループを再作成します。

```
ADD REPLICAT <group name>, EXTTRAIL <trail name>
```

16. Extract、データ・ポンプ (使用する場合) および Replicat を起動します。

```
START EXTRACT <group name>
START EXTRACT <pump name>
START REPLICAT <group name>
```

システムの停止

メンテナンスや他の Oracle GoldenGate に影響する作業のためにシステムを停止する場合、次の手順に従って、Extract で確実にすべてのトランザクション・ログ・レコードを処理します。そうしないと、同期データが失われる可能性があります。

1. Oracle GoldenGate によって処理されるトランザクションを生成するすべてのアプリケーションおよびデータベース・アクティビティを停止します。
2. GGSCI を実行します。
3. GGSCI で、SEND EXTRACT コマンドを LOGEND オプション付きで発行します。このコマンドで、Extract プロセスがデータソースのレコードの処理を終了したかどうかを問い合せます。

```
SEND EXTRACT <group name> LOGEND
```

4. YES ステータスが戻されるまで、コマンドを発行し続けます。その時点で、すべてのトランザクション・ログ・データが処理されているため、Oracle GoldenGate とシステムを安全に停止できます。

データベース属性の変更

この項では、データベースの表および構造に対して実行される管理操作について説明します。

ソース・データベースへの表の追加

Oracle GoldenGate では、プロセスを停止および起動したり、特別な手順を実行することなく、同期対象の表を追加できます。手順は、TABLE パラメータでの表の指定にワイルドカードを使用しているかどうかに応じて異なります。

表を追加する手順 (ワイルドカードを使用する場合)

TABLE パラメータでワイルドカードを使用して同期対象の表を指定している場合、Oracle GoldenGate は、そのワイルドカード・パターンに一致する名前を持つすべての新しい表の同期を開始します。新しい表の同期を準備するには、次の手順を実行します。

1. Extract を停止します。

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

2. Extract および Replicat のパラメータ・ファイルで、WILDCARDRESOLVE パラメータが、デフォルトの DYNAMIC 以外の設定で使用されていないことを確認します。
3. 必要に応じて TABLE 文および MAP 文に表名を追加します。
4. 新しい表をソース・データベースに追加し、ターゲット表をターゲット・データベースに追加します。(Oracle GoldenGate では DDL がレプリケートされないため、両方に表を作成する必要があります。) 新しい表へのユーザー・アクセスはまだ許可しないでください。
5. 現在のデータベースでサポートされる場合、表に対して GGSCI の ADD TRANDATA コマンドを実行します。
6. ソース表とターゲット表の定義が異なる場合、ソース表に対し DEFGEN ユーティリティを実行してソース定義を生成し、その新しい定義を、ターゲット・システムにある*既存*のソース定義ファイルにコピーします。追加する表に、現在の定義テンプレートと一致する定義が含まれる場合、DEFGEN を実行する必要はありません。MAP パラメータの DEF オプションでそのテンプレートを指定します。
7. 表へのユーザー・アクセスを許可します。

表を追加する手順 (ワイルドカードを使用しない場合)

表名が (ワイルドカードを使用せずに) 完全名で明示的に定義されている場合、新しい表をソース・データベースに追加するには、次の手順を実行します。

1. Extract を停止します。

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

2. 新しい表をソース・データベースに追加し、ターゲット表をターゲット・データベースに追加します。(Oracle GoldenGate では DDL がレプリケートされないため、両方に表を作成する必要があります。) 新しい表へのユーザー・アクセスはまだ許可しないでください。
3. 現在のデータベースでサポートされる場合、表に対して GGSCI の ADD TRANDATA コマンドを実行します。
4. ソース表とターゲット表の定義が異なる場合、ソース表に対し DEFGEN ユーティリティを実行してソース定義を生成し、その新しい定義を、ターゲット・システムにある*既存*のソース定義ファイルにコピーします。追加する表に、現在の定義テンプレートと一致する定義が含まれる場合、DEFGEN を実行する必要はありません。MAP パラメータの DEF オプションでそのテンプレートを指定します。
5. 表へのユーザー・アクセスを許可します。

同期しているソース表の属性の変更

注意　「DB2 z/OS の表の列を追加する ALTER TABLE の実行」も参照してください。

列やパーティションの追加または変更、サプリメンタル・ロギング詳細の変更 (Oracle)、またはすでに同期されている表に対するその他の変更を行う場合は、Oracle GoldenGate プロセスを停止して再起動し、新規属性がオブジェクト・レコードに追加されるようにする必要があります。

1. 変更する表へのユーザーおよびアプリケーションによるアクセスを停止します。
2. ソース表を変更します。

3. 現在の時刻をメモします。
4. GGSCI で、Extract の現在のチェックポイントの位置が、メモした時刻を過ぎるのを確認するまで、SHOWCH を指定して INFO EXTRACT コマンドを発行し続けます。

```
INFO EXTRACT <group name>, SHOWCH
```

5. Extract を停止します。

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

6. Replicat が証跡の最後 (EOF) に到達したのを確認するまで、INFO REPLICAT コマンドを発行し続けます。
7. ターゲット表を変更します。
8. ソース表とターゲット表の定義が異なる場合、ソース表に対し DEFGEN ユーティリティを実行して更新後のソース定義を生成してから、ターゲット・システムにある*既存*のソース定義ファイルで、その表の古い定義を新しい定義で置き換えます。
9. Extract、Replicat の順に起動します。

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

DB2 z/OS の表の列を追加する ALTER TABLE の実行

再配列された行形式で 1 つ以上の可変長の列を含む表に、固定長の列を追加する場合、その表を静止できるかどうかに応じて次のいずれかの手順を実行する必要があります。

表を静止できる場合

1. Extract が静止前に発生したトランザクションの取得を終了するまで待機します。
2. 表を変更して列を追加します。
3. 表領域を再編成します。
4. Extract を再起動します。
5. 表のアクティビティを再開します。

表を静止できない場合

1. Extract を停止します。
2. パラメータ・ファイルの TABLE 文から表を削除します。
3. Extract を再起動します。
4. 表を変更して列を追加します。
5. 表領域を再編成します。
6. Extract を停止します。
7. 表を TABLE 文に再度追加します。
8. ソース表とターゲット表を再同期します。
9. Extract を起動します。
10. 表のアクティビティを再開します。

ソース表の削除および再作成

Oracle GoldenGate の実行中にソース表の削除および再作成を行う場合、慎重に作業する必要があります。

1. 表へのアクセスを停止します。
2. Extract がトランザクション・ログからその表について残っている変更をすべて処理するまで待機します。Extract が終了したことを確認するには、GGSCI で INFO EXTRACT コマンドを使用します。

   ```
   INFO EXTRACT <group name>
   ```

3. Extract を停止します。

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

4. 表を削除して再作成します。
5. 現在のデータベースでサポートされる場合、表に対して GGSCI の ADD TRANDATA コマンドを実行します。
6. 再作成操作によって、ソース表の定義がターゲットの定義と異なるように変更された場合、ソース表に対して DEFGEN ユーティリティを実行し、ソース定義を生成して、ターゲット・システムの*既存の*ソース定義ファイルの新しい定義で古い定義を置き換えます。
7. 表へのユーザー・アクセスを許可します。

REDO スレッドの数の変更

Oracle RAC データベース・クラスタの REDO スレッドの数が変更されたら、必ず Extract グループを削除して再度追加する必要があります。Extract グループを削除して追加するには、次の手順を実行します。

1. ソース・システムとターゲット・システムで、GGSCI を実行します。
2. Extract と Replicat を停止します。

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

3. ソース・システムで、次のコマンドを発行して Extract グループを削除します。

   ```
   DELETE EXTRACT <group name>
   ```

4. 標準のオペレーティング・システム・コマンドを使用して、証跡ファイルを削除します。
5. 新しいスレッド数を指定して、Extract グループを再度追加します。

   ```
   ADD EXTRACT <group name> TRANLOG, THREADS <n>, BEGIN NOW
   ```

6. 証跡を再度追加します。

   ```
   ADD RMTTRAIL <trail name>, EXTRACT <group name>
   ```

7. Extract のパラメータ・ファイルを開き、新しい証跡を参照するように RMTTRAIL パラメータを変更します。

   ```
   EDIT PARAMS <group name>
   ```

8. Extract を起動します。

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

ORACLE_SID の変更

ORACLE_SID および ORACLE_HOME は、オペレーティング・システム・レベルの環境変数を修正せずに変更できます。変更対象がソース・データベースとターゲット・データベースのどちらであるかに応じて、Extract または Replicat のパラメータ・ファイルで次のパラメータを設定します。その後、パラメータの変更を反映するため、Extract または Replicat を停止して再起動します。

```
SETENV (ORACLE_HOME=<location>)
SETENV (ORACLE_SID="<SID>")
```

アーカイブ・ログの消去

Oracle のアーカイブ・ログは、Extract の読取りチェックポイントおよび書込みチェックポイントがそのログの最後を過ぎたら、安全に消去できます。Extract は、トランザクションのコミットが終了するまでそのトランザクションを証跡に書き込まないため、すべてのオープン・トランザクションを追跡する必要があります。そのため、Extract では、各オープン・トランザクションの開始記録が含まれるアーカイブ・ログと、それ以降のすべてのアーカイブ・ログにアクセスする必要があります。

Extract は、新規トランザクションに関する現在のアーカイブ・ログ (読取りチェックポイント) を読み取ると同時に、未コミットのトランザクションが存在する最も古いアーカイブ・ログ内にチェックポイント (リカバリ・チェックポイント) を保持します。

Extract のチェックポイント位置を特定するには、GGSCI で次のコマンドを使用します。

```
INFO EXTRACT <group name>, SHOWCH
```

- Input Checkpoint フィールドに、Extract の起動時に処理を開始した位置が示されます。
- Recovery Checkpoint フィールドに、最も古い未コミット・トランザクションの位置が示されます。
- Next Checkpoint フィールドに、Extract が読取りを行っている REDO ログの位置が示されます。
- Output Checkpoint フィールドに、Extract が書込みを行っている位置が示されます。

Recovery Checkpoint フィールドにリストされている順序番号を取得して、Extract で不要になったすべてのアーカイブ・ログを消去するシェル・スクリプトを記述できます。その番号よりも前のすべてのアーカイブ・ログは、安全に削除できます。

DB2 表の再編成 (z/OS プラットフォーム)

IBM 社の REORG ユーティリティを使用して圧縮表領域が含まれる DB2 表を再編成する場合、その表が Oracle GoldenGate で処理中であれば、KEEPDICTIONARY オプションを指定してください。この操作より、REORG ユーティリティによって圧縮ディクショナリが再作成されることを防止します。圧縮ディクショナリが再作成されると、変更前に書き込まれたログ・データを解凍できなくなり、Extract は異常終了します。別の方法としては、再編成を実行する前に必ず表のすべての変更を Oracle GoldenGate で抽出しておきます。そうしないと、表が切り捨てられます。

証跡ファイルのサイズの変更

証跡ファイルのサイズを変更するには、証跡がローカル証跡であるかリモート証跡であるかに応じて、ALTER EXTTRAIL または ALTER RMTTRAIL コマンドの MEGABYTES オプションを使用します。ファイル・サイズを変更するには、次の手順を実行します。

1. 証跡の場所に応じて次のコマンドのいずれかを発行し、変更する証跡のパス名および関連する Extract グループの名前を表示します。ワイルドカードを使用してすべての証跡を表示します。

   (リモート証跡)

   ```
   INFO RMTTRAIL *
   ```

   (ローカル証跡)

   ```
   INFO EXTTRAIL *
   ```

2. 証跡の場所に応じて次のコマンドのいずれかを発行し、ファイル・サイズを変更します。

   (リモート証跡)

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

   (ローカル証跡)

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

3. 次のコマンドを発行して、Extract の証跡を次のファイルに切り替えます。

   ```
   SEND EXTRACT <group name>, ROLLOVER
   ```

第 21 章

リバース・ユーティリティによるデータ変更の取消し

リバース・ユーティリティの概要

リバース・ユーティリティは、変更前イメージを使用して、指定された表、レコードおよび期間のデータベース変更を元に戻します。これにより、データベース全体のリストアが必要な他の方法とは異なり、選択的な取消しを実行できます。

リバース・ユーティリティは、次の用途で使用できます。

- テスト・データベースをテスト実行前の元の状態にリストアします。リバース・ユーティリティは変更の取消しのみを行うため、テスト・データベースを数分程度でリストアできます。これは、数時間かかることもあるデータベースの完全なリストアと比較して、ずっと効率的です。
- データの破損または偶発的な削除によって発生したエラーを取り消します。たとえば、WHERE 句を指定せずに UPDATE コマンドまたは DELETE コマンドを発行した場合、リバース・ユーティリティでその操作を取り消すことができます。

リバース・ユーティリティを使用するには、次の手順を実行します。

- Extract を実行して変更前データを抽出します。
- リバース・ユーティリティを実行して、トランザクションを反転します。
- Replicat を実行して、リストアされたデータをターゲット・データベースに適用します。

リバース・ユーティリティでは、次の操作を実行して先行操作を反転します。

- 逆の順序で処理できるように、1 つの抽出ファイル、一連の抽出ファイルまたは 1 つの証跡におけるデータベース操作の順序を反転し、同じキーを持つレコードが適切に適用されることを保証します。
- 削除操作を挿入操作に変更します。
- 挿入を削除に変更します。
- 変更前イメージの更新を変更後イメージの更新に変更します。
- 開始トランザクション・インジケータと終了トランザクション・インジケータを反転します。

図 30 リバース・ユーティリティのアーキテクチャ

リバース・ユーティリティの制限

- コミット・タイムスタンプは反転手順で変更されないため、証跡の時間順序は過去に戻ります。そのため、タイムスタンプに基づいて Replicat の位置を指定することはできません。
- Oracle GoldenGate では、次のデータ型の変更前イメージが保存されないため、これらの型はリバース・ユーティリティではサポートされません。更新操作および削除操作を反転するには、変更前イメージが必要です。

表 49　リバース・ユーティリティでサポートされないデータ型

DB2 (サポート対象の全 OS)	Oracle	SQL Server	Sybase	Teradata
BLOB CLOB DBCLOB	CLOB BLOB NCLOB LONG LONG RAW XMLType UDT NESTED TABLE VARRAY	TEXT IMAGE NTEXT VARCHAR (MAX)	VARBINARY BINARY TEXT IMAGE	サポート対象なし。この理由は、Teradata ベンダー・アクセス・モジュールによって取得されるのが行の変更後イメージのみであるためです。

リバース・ユーティリティの構成

トランザクション・データは、次のいずれかの方法で抽出できます。

- バッチ・プロセスとして (コマンド・シェルから作成および起動され、1 つの抽出ファイルまたは一連の抽出ファイルに書込みを行います)。複数のファイルを使用している場合、Oracle GoldenGate では、トランザクションの順序を維持するため、反転処理中にファイル順序が自動的に反転されます。次に例を示します。

  ファイル IN000004 は反転され、OUT000001 に書き込まれます。

  ファイル IN000003 は反転され、OUT000002 に書き込まれます。

  ファイル IN000002 は反転され、OUT000003 に書き込まれます。

  ... (以降同様)

- オンライン・プロセスとして (GGSCI を通じて作成および起動され、標準のローカル証跡またはリモート証跡に書込みを行います)。Oracle GoldenGate では、トランザクションの順序を維持するため、反転処理中にファイル順序が自動的に反転されます。

リバース・ユーティリティを構成する手順

リバース・ユーティリティを構成するには、表 50 および表 51 に記載されているパラメータを使用して Extract および Replicat のパラメータ・ファイルを作成します。これらのパラメータに加え、他の任意のパラメータや、現在の同期構成に必要とされる特別な MAP 文も指定します。

表 50　リバース・ユーティリティ用の Extract パラメータ・ファイル

パラメータ	説明
{EXTRACT <group> | SPECIALRUN, TRANLOG}	◆ EXTRACT <group> では、これをオンライン Extract プロセスとして指定します。GGSCI でこのプロセスを作成します。 ◆ SPECIALRUN, TRANLOG では、これを特別バッチ実行として指定し、変更前データを抽出するためのソースとしてトランザクション・ログを定義します。
BEGIN <time>	(SPECIALRUN のみ) 反転処理を開始する時点となるデータソースのタイムスタンプ。処理は、BEGIN で指定された時刻以降のタイムスタンプを持つ最初のレコードから開始されます。 注意 : オンライン・プロセスの開始時刻を指定するには、ADD EXTRACT コマンドを使用します。
END {<time> | RUNTIME}	<time> によって、Extract は、このパラメータで指定された時刻より後のタイムスタンプを持つデータソースのレコードに到達したときに終了します。 有効な値 : ◆ <date> は、yyyy-mm-dd という書式の日付です。 ◆ <time> は、24 時間制に基づく hh:mi[:ss[.cccccc]] という書式の時刻です。 RUNTIME によって、Extract は、現在の日時より後のタイムスタンプを持つデータソースのレコードに到達したときに終了します。この時点以前のタイムスタンプを持つ処理されていないすべてのレコードが処理されます。RUNTIME を使用するメリットの 1 つとして、実行ごとにパラメータ・ファイルを修正して日時を変更する必要がなくなります。かわりに、独自のバッチ・プログラム内でプロセスの開始時刻を制御できます。
[SOURCEDB <dsn>,] [USERID <user id> [, PASSWORD <pw>]] ◆ SOURCEDB では、データソース名を指定します (接続情報で必要な場合)。Oracle では必要ありません。 ◆ USERID では、必要に応じてデータベース資格証明を指定します。Oracle では、次のようなホスト文字列を含めることができます。 USERID ggs@ora1.ora, PASSWORD ggs123	データベース接続情報を指定します。これらのパラメータでは、オペレーティング・システム・レベルでの認証も可能です。『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』を参照してください。
NOCOMPRESSDELETES	このパラメータによって、Extract は、主キーに限定せずにすべての列データを出力に送信します。削除操作を挿入操作に逆変換できます。

表 50　リバース・ユーティリティ用の Extract パラメータ・ファイル (続き)

パラメータ	説明
GETUPDATEBEFORES	Oracle GoldenGate で更新をロールバックできるように変更前イメージを抽出します。
RMTHOST <hostname>	ターゲット・システムの名前または IP アドレス。
{EXTFILE <input file>| RMTFILE <input file>} [, MAXFILES <n>] または {EXTTRAIL <input trail> | RMTTRAIL <input trail>}	次のいずれかを使用してリバース・ユーティリティに対する 1 つ以上の*入力ファイル*を指定します。 **SPECIALRUN を使用してバッチ実行を実施する場合 :** ◆ EXTFILE を使用してローカル・システムの抽出ファイルを指定します。 ◆ RMTFILE を使用してリモート・システムの抽出ファイルを指定します。 次のように、ファイルの相対名またはフルパス名を指定します。 EXTFILE /home/ggs/dirdat/input.dat Oracle GoldenGate 証跡と同様にロールオーバーする一連のファイルを作成するには、MAXFILES オプションを使用します。このオプションを使用すると、オペレーティング・システムのファイル・サイズの制限に対応できます。 **GGSCI を使用してオンライン Extract グループを作成する場合 :** ◆ EXTTRAIL を使用してローカル・システムの抽出証跡を指定します。 ◆ RMTTRAIL を使用してリモート・システムのリモート証跡を指定します。 次のように証跡の相対名またはフルパス名を指定します (2 文字の証跡名を含めます)。 EXTTRAIL /home/ggs/dirdat/rt
TABLE <owner.name>;	複数の TABLE 文またはワイルドカードで指定した、処理する 1 つ以上の表。特別な選択基準およびマッピング基準を含めます。

例　この Extract パラメータ・ファイルの例では、リモート抽出ファイルを使用します。

```
SPECIALRUN, TRANLOG
BEGIN 2011-01-09 14:04:55
END 2011-01-09 14:12:20
USERID ggs, PASSWORD ggs
GETUPDATEBEFORES
NOCOMPRESSDELETES
RMTHOST sysb, MGRPORT 8040
RMTFILE /home/ggs/dirdat/input.dat, purge
TABLE tcustmer;
TABLE tcustord;
```

例 この Extract パラメータ・ファイルの例では、最大 10 個のファイルがディスク上に配置されるまで必要に応じてロールオーバーされる一連のリモート抽出ファイルを使用します。

```
SPECIALRUN, TRANLOG
BEGIN 2011-01-09 14:04:55
END 2011-01-09 14:12:20
USERID ggs, PASSWORD ggs
GETUPDATEBEFORES
NOCOMPRESSDELETES
RMTHOST sysb, MGRPORT 8040
RMTFILE /home/ggs/dirdat/in, MAXFILES 10
TABLE tcustmer;
TABLE tcustord;
```

例 この Extract パラメータ・ファイルの例では、リモート証跡を使用します。

```
EXTRACT ext_1
END 2011-01-09 14:12:20
USERID ggs, PASSWORD ggs
GETUPDATEBEFORES
NOCOMPRESSDELETES
RMTHOST sysb, MGRPORT 8040
RMTTRAIL /home/ggs/dirdat/in
TABLE tcustmer;
TABLE tcustord;
```

表 51 リバース・ユーティリティ用の Replicat パラメータ・ファイル

パラメータ	説明
{REPLICAT <group> | SPECIALRUN}	◆ REPLICAT <group> は、これが GGSCI で作成するオンライン Replicat プロセスであることを示します。 ◆ SPECIALRUN は、これが特別バッチ実行 Replicat プロセスであることを示します。
END {<time> | RUNTIME}	<time> によって、Replicat は、このパラメータで指定された時刻より後のタイムスタンプを持つデータソースのレコードに到達したときに終了します。 有効な値 : ◆ <date> は、yyyy-mm-dd という書式の日付です。 ◆ <time> は、24 時間制に基づく hh:mi[:ss[.cccccc]] という書式の時刻です。 RUNTIME によって、Replicat は、現在の日時より後のタイムスタンプを持つデータソースのレコードに到達したときに終了します。この時点以前のタイムスタンプを持つ処理されていないすべてのレコードが処理されます。RUNTIME を使用するメリットの 1 つとして、実行ごとにパラメータ・ファイルを修正して日時を変更する必要がなくなります。かわりに、独自のバッチ・プログラム内でプロセスの開始時刻を制御できます。

表 51　リバース・ユーティリティ用の Replicat パラメータ・ファイル (続き)

パラメータ	説明
EXTFILE <output file> [, MAXFILES <n>]	SPECIALRUN を使用する場合にのみ使用します。リバース・ユーティリティ用の 1 つ以上の*出力*ファイルを指定します。Replicat は、出力ファイルからの読取りを行い、変更前データをデータベースに適用して、それを以前の状態にリストアします。 *このファイルは*、EXTFILE または RMTFILE で指定した入力ファイルとは異なる必要があります。次のように、一意の相対名または完全修飾名を使用します。 EXTFILE /home/ggs/dirdat/output.dat Oracle GoldenGate 証跡と同様にロールオーバーする一連のファイルを作成するには、MAXFILES オプションを使用します。このオプションを使用すると、オペレーティング・システムのファイル・サイズの制限に対応できます。
[TARGETDB <dsn>,] [USERID <user id> [, PASSWORD <password>]] ◆ TARGETDB では、データソース名を指定します (接続情報で必要な場合)。Oracle では必要ありません。 ◆ USERID では、必要に応じてデータベース資格証明を指定します。Oracle では、次のようなホスト文字列を含めることができます。 USERID ggs@ora1.ora, PASSWORD ggs123	データベース接続情報を指定します。これらのパラメータでは、オペレーティング・システム・レベルでの認証も可能です。『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』を参照してください。
{SOURCEDEFS <full_pathname>} | ASSUMETARGETDEFS ◆ SOURCEDEFS は、ソース表とターゲット表に異なる定義が含まれる場合に使用します。DEFGEN によって生成されたソース定義ファイルを指定します。DEFGEN の詳細は、115 ページを参照してください。 ◆ ASSUMETARGETDEFS は、ソース表とターゲット表に同じ定義が含まれる場合に使用します。	データ定義の解釈方法を指定します。
MAP <owner.name>, TARGET <owner.name>;	(複数の MAP 文またはワイルドカードで指定した) 反転データを適用する 1 つ以上の表。ソース・データベースからのデータを反転する場合、ソースとターゲットの TABLE エントリは同じです。ターゲット・データベースからのレプリケート・データを反転する場合、各 MAP 文のソースとターゲットは異なります。

例 次に示すのは、SPECIALRUN を使用した Replicat パラメータ・ファイルの例です。

```
SPECIALRUN
END RUNTIME
EXTFILE /home/ggs/dirdat/output.dat
USERID ggs, PASSWORD ggs
ASSUMETARGETDEFS
MAP tcustmer, TARGET tcustmer;
```

例 次に示すのは、オンライン Replicat グループを使用した Replicat パラメータ・ファイルの例です。

```
REPLICAT rep_1
END RUNTIME
USERID ggs, PASSWORD ggs
ASSUMETARGETDEFS
MAP tcustmer, TARGET tcustmer;
```

反転処理のためのオンライン・プロセス・グループおよび証跡の作成

オンライン・プロセス・グループを使用して取消し手順を実行するには、次のものを作成する必要があります。

- オンライン Extract グループ。
- Extract グループにリンクしたローカル証跡またはリモート証跡。Extract は、データベースからデータを取得し、そのデータをこの証跡に書き込みます。これは、リバース・ユーティリティに対する入力証跡となります。
- オンライン Replicat グループ。
- Replicat グループにリンクしたもう 1 つのローカル証跡またはリモート証跡。これは、リバース・ユーティリティによって書き込まれる出力証跡となります。Replicat は、この証跡を読み取って反転データを適用します。

反転処理用のオンライン Extract グループを作成する手順

```
ADD EXTRACT <group name>, TRANLOG, BEGIN {NOW | <start point>}
```

条件 :

- <group name> は、Extract グループの名前です。グループ名には、最大 8 文字を指定できます。大 / 小文字は区別されません。
- TRANLOG では、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。
- BEGIN では、処理を開始する時点となる開始タイムスタンプを指定します。次のいずれかを使用します。
  - NOW では、グループを作成するために ADD EXTRACT コマンドが実行された時点のタイムスタンプが指定された変更から抽出を開始します。
  - <YYYY-MM-DD HH:MM[:SS[.CCCCCC]]> は、開始ポイントとして正確なタイムスタンプを指定するための書式です。

Extract グループにリンクした入力証跡を作成する手順

```
ADD {EXTTRAIL | RMTTRAIL} <pathname>, EXTRACT <group name>
[, MEGABYTES <n>]
```

条件 :

- EXTTRAIL では、ローカル・システムの証跡を指定します。RMTTRAIL では、リモート・システムの証跡を指定します。
- <pathname> は、2 文字の名前 (任意の 2 つの英数字) を含む入力証跡の相対名または完全修飾名です (c:\ggs\dirdat\rt など)。これは、Extract のパラメータ・ファイルで指定したものと同じ名前である必要があります。
- EXTRACT <group name> では、Extract グループの名前を指定します。
- MEGABYTES <n> は、各証跡ファイルのサイズを MB 単位で設定できるオプション引数です (デフォルトは 10 です)。

反転処理用の Replicat グループを作成する手順

```
ADD REPLICAT <group name>, EXTTRAIL <pathname>
[, BEGIN <start point> | , EXTSEQNO <seqno>, EXTRBA <rba>]
[, CHECKPOINTTABLE <owner.table>]
[, NODBCHECKPOINT]
```

条件 :

- <group name> は、Replicat グループの名前です。グループ名には、最大 8 文字を指定できます。大 / 小文字は区別されません。
- EXTTRAIL <pathname> は、ADD RMTTRAIL コマンドを使用してこの Replicat に作成する出力証跡の相対名または完全修飾名です。
- BEGIN <start point> では、処理のための初期チェックポイントおよび開始ポイントを確定してオンライン Replicat グループを定義します。次のいずれかを使用します。
  - NOW では、グループを作成するために ADD REPLICAT コマンドが実行された時点のタイムスタンプが指定されたレコードからレプリケートを開始します。
  - <YYYY-MM-DD HH:MM[:SS[.CCCCCC]]> は、開始ポイントとして正確なタイムスタンプを指定するための書式です。
- EXTSEQNO <seqno> では、処理を開始する証跡内のファイルの順序番号を指定します。EXTRBA <relative byte address> では、そのファイル内の開始ポイントとして相対バイト・アドレスを指定します。このオプションを使用しない場合、処理はデフォルトで証跡の最初から開始されます。順序番号には数値を指定しますが、埋込み用の 0 (ゼロ) は使用しません。たとえば、証跡ファイルが c:\ggs\dirdat\aa000026 である場合、EXTSEQNO 26 と指定します。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 にアクセスしてください。
- CHECKPOINTTABLE <owner.table> では、GLOBALS ファイルで指定されたデフォルト以外のチェックポイント表の所有者および名前を指定します。このオプションを使用するには、ADD CHECKPOINTTABLE コマンドを使用してデータベースにチェックポイント表を追加する必要があります (121 ページの「チェックポイント表の作成」を参照)。
- NODBCHECKPOINT では、この Replicat グループでチェックポイント表を使用しないことを指定します。

Replicat グループにリンクした出力証跡を作成する手順

```
ADD {EXTTRAIL | RMTTRAIL} <pathname>, REPLICAT <group name>
[, MEGABYTES <n>]
```

条件 :

- EXTTRAIL では、ローカル・システムの証跡を指定します。RMTTRAIL では、リモート・システムの証跡を指定します。

- <pathname> は、2 文字の名前 (任意の 2 つの英数字) を含む出力証跡の相対名または完全修飾名です (c:\ggs\dirdat\rt など)。これは、ADD REPLICAT コマンドの EXTTRAIL で指定した証跡である必要があります。
- REPLICAT <group name> では、Replicat グループの名前を指定します。
- MEGABYTES <n> は、各証跡ファイルのサイズを MB 単位で設定できるオプション引数です (デフォルトは 10 です)。

リバース・ユーティリティの実行

反転処理をバッチ・ジョブとして実行する手順

オペレーティング・システムのコマンド・シェル内で次の手順を実行します。プログラムを実行する場合、Oracle GoldenGate ディレクトリのフルパス名を使用します。

1. コマンド・シェルから、作成した Extract のパラメータ・ファイルを使用して Extract を実行します。Extract は、Extract のパラメータ・ファイルで指定されているデータを取得します。

   ```
   /<GoldenGate_directory>/extract paramfile irprm/<Extract_paramfile>.prm
   ```

2. データ抽出が終了したら、完全修飾パス名を使用するか、Oracle GoldenGate ディレクトリに移動してそこから reverse を実行することで、リバース・ユーティリティを実行します。

   注意 UNIX システムでは、UNIX の reverse コマンドとの混同を避けるため、フルパス名または Oracle GoldenGate ディレクトリを基準とする相対パスを使用することが特に重要です。

   ```
   /<GoldenGate_directory>/reverse <input file>, <output file>
   ```

   条件 :

   - <input file> は、Extract のパラメータ・ファイルの EXTFILE または RMTFILE で指定されている入力ファイルです。ワイルドカードを使用して、一連の複数のファイルを指定します (in* など)。
   - <output file> は、Replicat のパラメータ・ファイルの EXTFILE で指定されている出力ファイルです。ワイルドカードを使用して、一連の複数のファイルを指定します (out* など)。

     警告 UNIX システムでは、ファイル名の後に空白を*入れない*でください。 そうしないと、UNIX ファイル・システムでは、Oracle GoldenGate に適切にファイル名を戻すことができません。

例

```
/home/ggs/reverse input.dat, output.dat
```

例

```
/home/ggs/reverse in*, out*
```

3. reverse の実行が終了したら、作成した Replicat のパラメータ・ファイルを使用して Replicat を実行します。これにより、反転出力されたデータがデータベースに適用されます。

   ```
   /<GoldenGate_directory>/replicat paramfile dirprm/<Replicat param file>.prm
   ```

反転処理をオンライン・プロセスとして実行する手順

1. GGSCI から Extract を実行します。

   ```
   START EXTRACT <group>
   ```

2. Extract が指定されたレコードの取得を終了したことを示す EOF というメッセージが戻されるまで次のコマンドを発行します。

```
SEND EXTRACT <group>, STATUS
```

3. 完全修飾パス名を使用するか、Oracle GoldenGate ディレクトリに移動してそこから reverse を実行することで、リバース・ユーティリティを実行します。

注意 UNIX システムでは、UNIX の reverse コマンドとの混同を避けるため、フルパス名または Oracle GoldenGate ディレクトリを基準とする相対パスを使用することが特に重要です。

```
/<GoldenGate_directory>/reverse <input file>, <output file>
```

条件 :

- <input file> は、Extract のパラメータ・ファイルの EXTTRAIL または RMTTRAIL で指定されている入力ファイルです。
- <output file> は、ADD REPLICAT コマンドの EXTFILE で指定されている出力ファイルです。

例

```
\home\ggs\reverse input.c:\ggs\dirdat\et, output.c:\ggs\dirdat\rt
```

4. reverse の実行が終了したら、反転出力されたデータをデータベースに適用するために Replicat を実行します。

```
START REPLICAT <group>
```

リバース・ユーティリティにより実行された変更の取消し

反転処理によって予期しない結果や意図しない結果が発生した場合、データベースに元の変更を再適用できます。これを行うには、Replicat のパラメータ・ファイルを編集し、出力ファイルのかわりに*入力*ファイルを指定して、Replicat を再度実行します。

付録 1

Oracle GoldenGate レコードの形式

この付録では、証跡または抽出ファイルに書き込まれる Oracle GoldenGate レコードの形式について説明します。Oracle GoldenGate によって証跡または抽出ファイルに書き込まれる各変更レコードには、ヘッダー領域 (NOHEADERS パラメータが指定されていない場合)、データ領域および (状況により) ユーザー・トークン領域が含まれます。

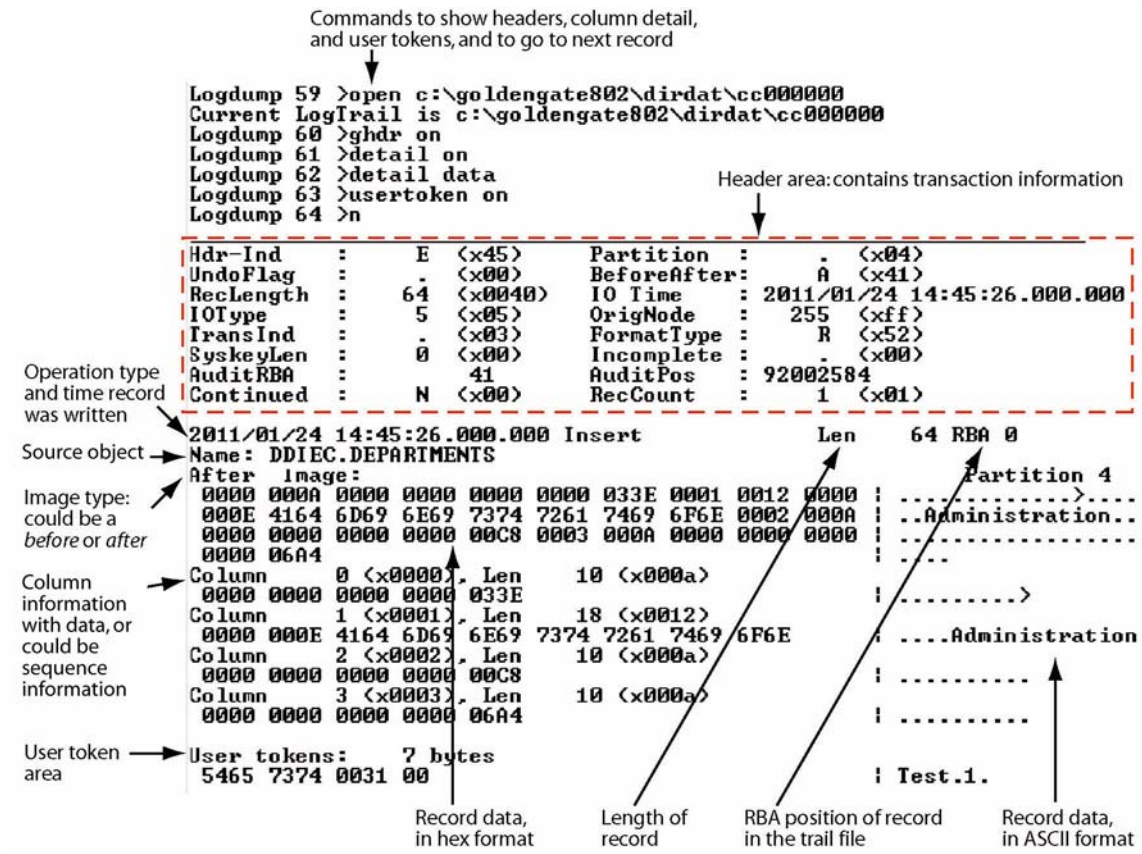
Oracle GoldenGate の証跡ファイルは構造化されていません。Oracle GoldenGate レコードは、Oracle GoldenGate ソフトウェアに付属するログダンプ・ユーティリティを使用して表示できます。詳細は、『*Windows and UNIX トラブルシューティングおよびチューニング・ガイド*』のログダンプの説明を参照してください。

注意 Oracle GoldenGate ソフトウェアの機能拡張のために、証跡レコードの形式は、このドキュメントに記載されていない変更に応じて修正される可能性があります。現在の構造を表示するには、ログダンプ・ユーティリティを使用してください。

Oracle GoldenGate レコードの例

次に、ログダンプでの表示どおりに Oracle GoldenGate レコードを記載します。最初の部分 (フィールドのリスト) はヘッダーで、2 番目の部分はデータ領域です。レコードは、Oracle GoldenGate でサポートされるすべてのプラットフォームで次のように表示されます。

図 31 ログダンプ・ユーティリティで表示されるサンプルの証跡レコード



レコード・ヘッダー領域

Oracle GoldenGate レコードのヘッダーには、レコードに格納されているデータのメタデータがあり、次の情報が含まれます。

- 挿入、更新、削除などの操作タイプ
- 更新の前または後を示すインジケータ
- トランザクション・グループやコミット・タイムスタンプなどのトランザクション情報

ヘッダー・フィールドの説明

次に、Oracle GoldenGate レコードのヘッダー・フィールドについて説明します。一部のフィールドは特定のプラットフォームにのみ適用されます。

表 52　Oracle GoldenGate レコードのヘッダー・フィールド

フィールド	説明
Hdr-Ind	この値は常に E である必要があり、レコードが Extract プロセスによって作成されたことを示します。それ以外のすべての値は、無効なデータであることを示します。
UndoFlag	(NonStop)Oracle GoldenGate が TMF 監査証跡から中断されたトランザクションを抽出する場合に条件付きで設定されます。通常、UndoFlag は 0 (ゼロ) に設定されますが、レコードが以前成功した操作を取り消したものである場合、UndoFlag は 1 に設定されます。制約違反のためにディスク・プロセスによって実行された取消しは、取消しとしてマークされません。
RecLength	レコード・バッファの長さ (バイト単位)。
IOType	レコードにより表される操作のタイプ。操作タイプのリストは、337 ページの表 53 を参照してください。
TransInD	現在のトランザクション内におけるレコードの位置。値は次のとおりです。 0: トランザクションの最初のレコード 1: トランザクションの最初と最後以外のレコード 2: トランザクションの最後のレコード 3: トランザクションの唯一のレコード
SyskeyLen	(NonStop) ソースが NonStop ファイルであり、システム・キーを保持している場合、そのシステム・キーの長さ (4 または 8 バイト)。システム・キーが存在する場合、レコードの最初の Syskeylen バイトがシステム・キーです。それ以外の場合、SyskeyLen は 0 (ゼロ) です。
AuditRBA	コミット・レコードの相対バイト・アドレス あるトランザクションのすべてのレコードは、同じコミット相対バイト・アドレスを持ちます。IO Time と AuditRBA の組合せによって、特定のトランザクションのデータが一意に識別されます。

表 52　Oracle GoldenGate レコードのヘッダー・フィールド (続き)

フィールド	説明
Continued	(Windows および UNIX) レコードが、大きすぎて 1 つのレコードに収まらないより大きなデータの 1 セグメントであるかどうかを示します。各セグメントには、LOB、CLOB および複数の VARCHAR が格納されます。 Y ：レコードはセグメントです。Oracle GoldenGate にこのデータが別のレコードに続いていることを示します。 N: データは別のセグメントに続いていません。一連のデータの最後の部分であるか、より大きなデータのセグメントではないレコードです。
Partition	このフィールドは、Oracle GoldenGate で内部的に使用されるもので、通常は特定のデータベースにとって意味のある情報ではありません。 Windows および UNIX のレコードの場合、このフィールドの値は常に 4(内部形式における FieldComp 圧縮レコード) です。これらのプラットフォームでは、Partition という語は、データがデータベース構造内の特定の論理パーティションまたは物理パーティションであることを示しているわけでは*ありません*。 NonStop のレコードの場合、このフィールドの値は、レコード・タイプによって異なります。 ◆ BulkIO 操作の場合、Partition は、バルク操作が実行されたソース・パーティションの数を示します。これによって、Oracle GoldenGate は、データが最初に書き込まれたソース・パーティションを判別します。Replicat は、Partition フィールドを使用してターゲット・パーティションの名前を特定します。レコード・ヘッダーのファイル名は、常にプライマリ・パーティションの名前になります。BulkIO レコードの有効な値は、0 から 15 です。 ◆ 他のバルク操作以外の NonStop 操作の場合、値は 0 または 4 のいずれかです。4 の値は、データが FieldComp レコード形式であることを示します。
BeforeAfter	レコードが更新操作の変更前 (B) イメージまたは変更後 (A) イメージのどちらであるかを示します。挿入は常に変更後イメージであり、削除は常に変更前イメージです。
IO Time	ソース・システムのローカル時刻を GMT 形式で示したコミット・レコードのタイムスタンプ。あるトランザクションのすべてのレコードは、同じコミット・タイムスタンプを持ちます。IO Time と AuditRBA の組合せによって、特定のトランザクションのデータが一意に識別されます。
OrigNode	(NonStop) データが抽出されたシステムのノード番号。NonStop クラスタの各システムは、一意のノード番号を持ちます。ノード番号の範囲は、0 から 255 です。 NonStop 以外から導出されたレコードの場合、OrigNode は 0 (ゼロ) です。

表 52　Oracle GoldenGate レコードのヘッダー・フィールド (続き)

フィールド	説明
FormatType	データがトランザクション・ログから読み取られたか、データベースからフェッチされたかを示します。 F: データベースからのフェッチ R: トランザクション・ログからの読取り
Incomplete	このフィールドは現在使用されていません。
AuditPos	Extract プロセスのトランザクション・ログでの位置を識別します。
RecCount	(Windows および UNIX)Oracle GoldenGate ファイルに書き込むために LOB データをチャンクに分割する必要がある場合に使用されます。RecCount は、チャンクを再構築するために使用されます。

ヘッダー・データの使用

Oracle GoldenGate レコードのヘッダーから取得可能な一部のデータは、マッピングに使用できます。これを行うには、@GETENV 関数の GGHEADER オプションを使用するか、TABLE または MAP パラメータの COLMAP 文のソース式として次のいずれかのトランザクション要素を使用します。

- GGS_TRANS_TIMESTAMP
- GGS_TRANS_RBA
- GGS_OP_TYPE
- GGS_BEFORE_AFTER_IND

@GETENV 関数の詳細は、『*Windows and UNIX リファレンス・ガイド*』を参照してください。

レコード・データ領域

Oracle GoldenGate 証跡レコードのデータ領域には、次のデータが含まれます。

- 変更が Oracle GoldenGate ファイルに書き込まれた時刻
- データベース操作のタイプ
- レコードの長さ
- 証跡ファイル内の相対バイト・アドレス
- 表名
- 16 進形式のデータ変更

次に、Windows、UNIX、Linux および NonStop システムの Oracle GoldenGate によって使用されるレコード・イメージ形式の相違点について説明します。この説明では、イメージ形式について「完全」および「圧縮」という語を使用しています。ここでは、これらの語は、このドキュメントの他の部分とは異なるコンテキストで使用されています。つまり、Extract が列データを証跡に書き込む方法を示しており、キー列および変更列のみが書き込まれるのか (圧縮)、すべての列が証跡に書き込まれるのか (未圧縮または完全イメージ) を意味しています。

完全レコード・イメージ形式

完全レコード・イメージ形式は、ソース・システムが HP NonStop で、レコード・ヘッダーに指定されている IOType が次のいずれかである場合にのみ証跡に生成されます。

3: 削除
5: 挿入
10: 更新

それぞれの完全レコード・イメージの形式は、そのレコードが元のファイルまたは表を直接読み取るプログラムから取得された場合と同じです。SQL 表、日時フィールド、NULL などのデータは、プログラムがそのデータをアプリケーション・バッファに選択する場合とまったく同じように書き込まれます。日時フィールドは内部的に 8 バイトのタイムスタンプとして表現されますが、外部形式は最大 26 バイトの文字列として表現可能です。Enscribe レコードは、元のファイルに存在するとおりの形式で取得されます。

操作タイプが Insert または Update の場合、イメージには操作*後*のレコードの内容が含まれます (変更後イメージ)。操作タイプが Delete の場合、イメージには操作*前*のレコードの内容が含まれます (変更前イメージ)。

Enscribe データベースから生成されるレコードでは、元のファイルの AUDITCOMPRESS 属性が ON に設定されていなければ、完全レコード・イメージが出力されます。AUDITCOMPRESS が ON の場合、元のファイルによって更新操作が取得されると、常に圧縮更新レコードが生成されます。(完全イメージは、FETCHCOMPS パラメータを使用することで、Extract プロセスによって取得できます。)

圧縮レコード形式

デフォルトでは、Windows および UNIX システムのプロセスによって書き込まれる証跡レコードは、常に圧縮されます。圧縮レコードの形式は次のとおりです。

```
<column index><column length><column data>[...]
```

条件 :

- <column index> は、ソース表内の列の序数索引です (2 バイト)。
- <column length> は、データの長さです (2 バイト)。
- <column data> は、NULL または VARCHAR の長さインジケータを含むデータです。

NonStop プラットフォームから書き込まれる Enscribe レコードは、圧縮されないことがあります。圧縮 Enscribe レコードの形式は次のとおりです。

```
<field offset><field length><field value>[...]
```

条件 :

- <field offset> は、変更された値の元のレコード内のオフセットです (2 バイト)。
- <field length> は、データの長さです (2 バイト)。
- <field data> は、NULL または VARCHAR の長さインジケータを含むデータです。

圧縮 Enscribe レコードの最初のフィールドは、主キーまたはシステム・キーです。

トークン領域

証跡レコードには、2 つのトークン領域も含まれることがあります。1 つは内部使用を目的としており、ここでは説明しません。もう 1 つはユーザー・トークン領域です。ユーザー・トークンは、ターゲット列へのレプリケーションまたは他の目的のために取得されて証跡レコードに格納される環境値です。使用される場合、これらのトークンはレコードのデータ部分の後に続き、ログダンプで確認すると次のように表示されます。

```
TKN-HOST          : syshq
TKN-GROUP         : EXTORA
TKN-BA_IND        : AFTER
TKN-COMMIT_TS     : 2011-01-24 17:08:59.000000
TKN-POS           : 3604496
TKN-RBA           : 4058
TKN-TABLE         : SOURCE.CUSTOMER
TKN-OPTYPE        : INSERT
TKN-LENGTH        : 57
TKN-TRAN_IND      : BEGIN
```

Oracle GoldenGate の操作タイプ

次に、Oracle GoldenGate の操作タイプの一部を示します。Oracle GoldenGate に新機能が追加されると、タイプが追加されることがあります。最新のリストを確認するには、ログダンプ・ユーティリティの SHOW RECTYPE コマンドを使用してください。

表 53　Oracle GoldenGate の操作タイプ

タイプ	説明	プラットフォーム
1-Abort	トランザクションが中断されました。	NSK TMF
2-Commit	トランザクションがコミットされました。	NSK TMF
3-Delete	レコードまたは行が削除されました。Delete レコードには、通常、完全レコード・イメージが含まれます。ただし、COMPRESSDELETES パラメータが使用されている場合は、キー列のみが示されます。	すべて
4-EndRollback	データベース・ロールバックが終了しました。	NSK TMF
5-Insert	レコードまたは行が挿入されました。Insert レコードには、完全レコード・イメージが含まれます。	すべて
6-Prepared	ネットワーク・トランザクションのコミット準備が完了しました。	NSK TMF
7-TMF-Shutdown	TMF が停止しました。	NSK TMF
8-TransBegin	現在使用されていません。	NSK TMF
9-TransRelease	現在使用されていません。	NSK TMF

表 53　Oracle GoldenGate の操作タイプ (続き)

タイプ	説明	プラットフォーム
10-Update	レコードまたは行が更新されました。Update レコードには、完全レコード・イメージが含まれます。注意 : レコード・ヘッダーのパーティション・インジケータが 4 の場合、そのレコードは FieldComp 形式 (「15-FieldComp」を参照) であり、更新は圧縮されています。	すべて
11-UpdateComp	TMF AuditComp 形式のレコードまたは行が更新されました。この形式では、変更されたバイトのみが示されます。<2-byte offset><2-byte length> 形式の 4 バイトの記述子が、各データ・フラグメントの先頭に配置されます。byte offset は、ソース表内の列の序数索引です。length はデータの長さです。	NSK TMF
12-FileAlter	データベース・ファイルの属性が変更されました。	NSK
13-FileCreate	データベース・ファイルが作成されました。	NSK
14-FilePurge	データベース・ファイルが削除されました。	NSK
15-FieldComp	SQL 表の行が更新されました。この形式では、変更されたバイトのみが示されます。変更されていない列の変更前イメージは、データベースでは記録されません。<2-byte offset><2-byte length> 形式の 4 バイトの記述子が、各データ・フラグメントの先頭に配置されます。byte offset は、ソース表内の列の序数索引です。length はデータの長さです。レコード・ヘッダーのパーティション・インジケータの 4 は、FieldComp 形式を示します。	すべて
16-FileRename	ファイル名が変更されました。	NSK
17-AuxPointer	新規データを保持する AUX 証跡および読取りを開始する場所に関する情報が含まれます。	NSK TMF
18-NetworkCommit	ネットワーク・トランザクションがコミットされました。	NSK TMF
19-NetworkAbort	ネットワーク・トランザクションが中断されました。	NSK TMF
90-(GGS)SQLCol	SQL 表に 1 つ以上の列が追加されたか、属性が変更されました。	NSK
100-(GGS)Purgedata	ファイルからすべてのデータが削除されました (PURGEDATA)。	NSK
101-(GGS)Purge(File)	ファイルが消去されました。	NSK 非 TMF
102-(GGS)Create(File)	ファイルが作成されました。Oracle GoldenGate レコードには、ファイル属性が含まれます。	NSK 非 TMF
103-(GGS)Alter(File)	ファイルが変更されました。Oracle GoldenGate レコードには、変更されたファイル属性が含まれます。	NSK 非 TMF

表 53　Oracle GoldenGate の操作タイプ (続き)

タイプ	説明	プラットフォーム
104-(GGS)Rename(File)	ファイル名が変更されました。Oracle GoldenGate レコードには、元の名前と新しい名前が含まれます。	NSK 非 TMF
105-(GGS)Setmode	SETMODE 操作が実行されました。Oracle GoldenGate レコードには、SETMODE の情報が含まれます。	NSK 非 TMF
106-GGSChangeLabel	CHANGELABEL 操作が実行されました。Oracle GoldenGate レコードには、CHANGELABEL の情報が含まれます。	NSK 非 TMF
107-(GGS)Control	CONTROL 操作が実行されました。Oracle GoldenGate レコードには、CONTROL の情報が含まれます。	NSK 非 TMF
115 および 117 (GGS)KeyFieldComp(32)	主キーが更新されました。Oracle GoldenGate レコードには、キーの変更前イメージと、キーおよび行の変更後イメージが含まれます。データは FieldComp 形式 (圧縮) です。つまり、変更されていない列の変更前イメージは、データベースでは記録されません。	Windows および UNIX
116-LargeObject 116-LOB	RAW 列、BLOB 列、CLOB 列または LOB 列を示します。このタイプのデータは、複数のレコードにわたって格納されます。	Windows および UNIX
132-(GGS) SequenceOp	順序に対する操作を示します。	Windows および UNIX
160 - DDL_Op	DDL 操作を示します。	Windows および UNIX
161- RecordFragment	ベース・レコードを超えており、複数のレコードにわたって格納する必要のある大きな行の一部であることを示します。	Windows および UNIX
200-GGSUnstructured Block 200-BulkIO	BULKIO 操作が実行されました。Oracle GoldenGate レコードには、RAW DP2 ブロックが含まれます。	NSK 非 TMF
201 から 204	これらは、NonStop トレース・レコードの異なるタイプです。トレース・レコードは、Oracle GoldenGate サポートのアナリストによって使用されます。説明は次のとおりです。 ◆ ARTYPE_FILECLOSE_GGS 201: ソース・アプリケーションが、構造化されていない I/O に対して開かれていたファイルを閉じました。Replicat によって使用されます。 ◆ ARTYPE_LOGGERTS_GGS 202: ロガー・ハートビート・レコード。	NSK 非 TMF

表 53　Oracle GoldenGate の操作タイプ (続き)

タイプ	説明	プラットフォーム
	◆ ARTYPE_EXTRACTERTS_GGS 203: 未使用。 ◆ ARTYPE_COLLECTORTS_GGS 204: 未使用。	
205-GGSComment	ログダンプ・ユーティリティによって作成されたコメント・レコードを示します。コメント・レコードは、ログダンプによって、その SAVE コマンドでファイルに保存されるデータの最初と最後に作成されます。	すべて
249 から 254	これらは、NonStop トレース・レコードの異なるタイプです。トレース・レコードは、Oracle GoldenGate サポートのアナリストによって使用されます。説明は次のとおりです。 ◆ ARTYPE_LOGGER_ADDED_STATS 249: ソース・アプリケーションがロガーで開いている状態を閉じたときにロガーによって作成される統計レコード (SENDERSTATS が有効化されており、統計がログ証跡に書き込まれる場合)。 ◆ ARTYPE_LIBRARY_OPEN 250: BASELIB によって書き込まれ、アプリケーションがファイルを開いたことを示します。 ◆ ARTYPE_LIBRARY_CLOSE 251: BASELIB によって書き込まれ、アプリケーションがファイルを閉じたことを示します。 ◆ ARTYPE_LOGGER_ADDED_OPEN 252: 未使用。 ◆ ARTYPE_LOGGER_ADDED_CLOSE 253: 未使用。 ◆ ARTYPE_LOGGER_ADDED_INFO 254: ロガーによって書き込まれ、後続のレコードで I/O を実行したソース・アプリケーションに関する情報が含まれます (SENDERSTATS が有効化されており、統計がログ証跡に書き込まれる場合)。トレース・レコードのファイル名は、アプリケーションのオブジェクト・ファイルです。トレース・データには、アプリケーション・プロセスの名前と、その実行で使用されていたライブラリ (存在する場合) の名前が含まれます。	NSK 非 TMF

Oracle GoldenGate 証跡のヘッダー・レコード

Oracle GoldenGate 証跡にあるトランザクション関連のレコードに加え、各証跡ファイルにはファイル・ヘッダーが含まれます。

ファイル・ヘッダーは、データ・レコードに先行する証跡ファイルの先頭部分にレコードとして格納されます。証跡のヘッダーに格納されているレコードに関する情報によって、Oracle GoldenGate プロセスは、各レコードが Oracle GoldenGate の現行リリースでサポートされる形式であるかどうかを判断できます。

証跡のヘッダー・フィールドは、トークンとして格納されます。トークンの形式は、Oracle GoldenGate のすべてのリリースで同じです。あるリリースの Oracle GoldenGate でいずれかのトークンがサポートされない場合、そのトークンは無視されます。以前のリリースの Oracle GoldenGate との互換性を確保するため、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。

証跡のヘッダーは、ログダンプ・ユーティリティの FILEHEADER コマンドを使用して表示できます。ファイル・ヘッダーのトークンの詳細は、『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*』のログダンプの説明を参照してください。

用語集

次に、このマニュアルに含まれる用語について説明します。

用語	定義
異常終了	*異常な終了*。コンピュータ・システム上で実行されているプロセスの障害または予期しない終了。
変更後イメージ	挿入または更新の実行後におけるデータベース内の行の値。
エイリアス Extract	ソース・システムよりセキュアなネットワーク・ゾーン内に存在するターゲット・システムで動作する Extract グループ。エイリアス Extract の目的は、信頼度の低いソースに対してターゲットから TCP/IP 接続を開始することです。一度接続が確立されると、データはソース・システムで動作するパッシブ Extract グループによって通常どおり処理され、ネットワークを通じて転送されます。
追加モード	証跡に対するデフォルトの書込み方法。この方法では、Extract は、障害後に古いデータを上書きせずに、証跡ファイルに再読取りデータを追加します。
アーカイブ・ログ専用モード (ALO)	Extract の操作モード。プロセスは、本番データベース・システムまたはスタンバイ・データベース・システムのアーカイブ・トランザクション・ログから排他的に読取りを行うように構成されます。
バッチ Replicat 処理モード	バッチ・モードでは、Replicat は、同様の SQL 文を配列に編成し、それらを高速に適用します。Replicat は、メモリー・キュー内に複数の文をバッチとしてまとめ、各バッチを 1 回のデータベース操作で適用します。このモードの動作は、BATCHSQL パラメータによって制御されます。「標準 Replicat 処理モード」も参照してください。
監査証跡	レプリケーションおよびリカバリの目的で、データベースに行われた変更を格納する NonStop Server システムのファイル。
バッチ実行	開始と終了が明確な 1 回かぎりの処理実行 (特定の終了ポイントを持たないオンライン変更同期などの継続処理と対比されます)。
変更前イメージ	SQL 操作がデータベース内の行に対して実行される前に、その行に存在していた値。
双方向同期	複数のデータベースおよびサーバー全体で負荷分散が行われます。この環境では、通常、異なるユーザーが同じデータセットを変更可能で、それらの変更が Oracle GoldenGate によって同期されます。
BLOB	「*LOB*」を参照してください。

用語	定義
制限リカバリ	Extract リカバリ・システムの一部。制限リカバリによって、Extract が予期せずに停止して再起動する場合に、Extract の停止時点で存在していたオープン・トランザクションの数やそれらの経過時間にかかわらず、効率的なリカバリが保証されます。制限リカバリにより、Extract が停止した時点までリカバリして通常の処理を再開するまでに要する最大時間の上限が設定されます。
コール元	ユーザー・イグジット・ルーチンを実行する Oracle GoldenGate プロセス。
正規形式	Oracle GoldenGate が証跡または抽出ファイルにデータを格納する場合に使用するデータ形式。この形式によって、異機種データベース間でデータを高速かつ正確に交換できます。
カスケード同期	データがソース・システムから 1 つ以上の中間システムに送信され、さらにそれらのシステムから 1 つ以上の同期状態の他のシステムに送信される Oracle GoldenGate 構成。
変更同期	あるシステムのデータベースで行われたデータ変更を、1 つ以上の他のシステムに存在する同様のデータセットと同期するプロセス。
チェックポイント・ファイル	Oracle GoldenGate プロセスによって生成されたチェックポイントを格納するディスク上のファイル。
チェックポイント表	Replicat のチェックポイントを保持するターゲット・データベースに作成される表。オプションで、ディスク上の標準のチェックポイント・ファイルと組み合せて使用されます。
チェックポイント	Oracle GoldenGate プロセスの現在の読取り位置と書込み位置を記録する内部インジケータ。チェックポイントは、オンライン変更同期でデータの正確性とフォルト・トレランスを保証するために、Extract プロセスおよび Replicat プロセスによって使用されます。
CLOB	「*LOB*」を参照してください。
CMDSEC ファイル	GGSCI コマンド権限のルールを格納した Oracle GoldenGate ファイル。
Collector	TCP/IP を通じて Extract プロセスからデータを受信し、そのデータをターゲット・システムの証跡または抽出ファイルに書き込むプロセス。
衝突	Oracle GoldenGate によってレプリケートされたデータ変更がターゲット表に適用される場合に、ターゲット行が欠落または重複していると発生するエラー。
列	データベース表によって記述されるエンティティに割り当てられた一連の属性のうちの 1 つ。たとえば、「従業員」というエンティティには、名前、住所および電話番号の列を使用できます。
列マップ	「マップ」を参照してください。
列変換関数	データを選択または操作する目的で比較、テスト、計算などの処理を実行する Oracle GoldenGate 組込みの処理関数。

用語	定義
コミット	トランザクションを終了させ、そのトランザクション内の SQL 文で実行された変更を永続化するトランザクション制御文。
コミット順序番号 (CSN)	CSN は、トランザクション一貫性とデータ整合性を維持する目的でトランザクションを識別するために Oracle GoldenGate が作成する識別子です。CSN によって、トランザクションがデータベースにコミットされた特定の時点が一意に識別されます。CSN の構成および値は、トランザクションを生成したデータベースのタイプに応じて異なります。CSN は、データベースがトランザクションを識別する際に使用する一意の情報を取得して、それを内部的に一連のバイト列として表現します。ただし、Oracle GoldenGate は、CSN をプラットフォームに依存しない方法で処理します。
圧縮更新	SQL の更新操作を記録する方法。この方法では、更新の結果として変更された列値のみがトランザクション・ログに記録されます。
競合解決	同じ SQL 操作が (ほぼ) 同時に 2 つ以上のデータベースの同じ行に適用される場合の処理ルールおよびエラー処理ルールを提供する、双方向同期で使用される手順。
統合同期	異なるデータを 2 つ以上のデータベースから 1 つの中央データベース (データ・ウェアハウスなど) にレプリケートするプロセス。
変換	「*トランスフォーメーション*」を参照してください。
データ定義ファイル	「*ソース定義ファイル*」および「*ターゲット定義ファイル*」を参照してください。
データ・ポンプ	抽出ファイルまたは証跡から読取りを行うセカンダリ Extract プロセス。証跡へのデータ移入は、データソースから読取りを行うプライマリ Extract プロセスによって行われます。
データソース	Oracle GoldenGate によって処理されるデータ変更のコンテナ。次のデータソースを使用できます。 ◆ データベースのトランザクション・ログ ◆ ベンダー・アクセス・モジュール

用語	定義
データソース名 (DSN)	DSN では、データベースに対する ODBC 接続を定義します。DSN は、データベースに応じて、データベース名、データベース・ディレクトリ、データベース ODBC ドライバ名、データベース認証情報などの情報で構成されます。DSN によって、アプリケーションはアプリケーション・プログラム内に必須情報をエンコードせずにデータベースに接続できるため、Oracle GoldenGate などの外部アプリケーションでは DSN が必要です。 DSN には次の 3 つのタイプがあります。 ◆ システム DSN は、マシンにアクセスする任意のエンティティで使用できます。これはシステム構成内に格納されます。 ◆ ユーザー DSN は、特定のユーザーのみが使用できます。これはシステム構成内に格納されます。 ◆ ファイル DSN は、.dsn 拡張子付きのテキスト・ファイルに格納されます。これは、必要な ODBC ドライバがインストールされている異なるシステム間で共有できます。
データ型	個々のデータについて、そのデータの種類およびそのデータに実行可能な操作の種類を識別する属性。たとえば、整数データ型は数値を、文字データ型は文字を格納します。
DDL	*データ定義言語*。データベースの構造を定義するデータであり、行、列、表、索引およびデータベース詳細 (ファイルの場所、ユーザー、権限、記憶域パラメータなど) が含まれます。
DEFGEN	データ定義ファイルを生成する Oracle GoldenGate ユーティリティ。
廃棄ファイル	失敗した SQL 操作に関する情報が含まれる Oracle GoldenGate ファイル。このファイルは、レコードを処理できない場合に作成されますが、DISCARDFILE パラメータがパラメータ・ファイル内に存在し、ファイルの場所が指定されている必要があります。
DML	*データ操作言語*。データベースのデータを取得および操作します。SQL の場合、その操作は選択、挿入、更新および削除です。
DSN	「データソース名 (DSN)」を参照してください。
動的 Collector	Manager プロセスによって自動的に起動される Collector プロセス。静的 Collector と対比されます。
EMSCLNT	Windows などのサポートされるオペレーティング・システムで発生した Oracle GoldenGate のシステム・エラー・メッセージを NonStop Server の EMS(イベント管理サブシステム) サーバーに配信する Oracle GoldenGate ユーティリティ。
ENCKEYS ファイル	暗号化鍵を格納した Oracle GoldenGate 参照ファイル。
暗号化	解読するためのパスワードまたは復号化コードを所有するユーザー以外には判読不可能な形式にデータをエンコードする方法。

用語	定義
エラー・ログ	Oracle GoldenGate によって生成されたイベント、メッセージ、エラーおよび警告の処理を示すファイル。名前は ggserr.log で、Oracle GoldenGate のルート・ディレクトリに配置されます。
イベント・マーカー・システム	Oracle GoldenGate をカスタマイズして、フィルタ基準に適合するレコードに基づいて処理中に特定のアクションを実行するシステム。たとえば、特定のレコードが検出された場合にそのレコードをスキップしたり、Oracle GoldenGate プロセスを停止できます。 「イベント・レコード」も参照してください。
イベント・レコード	特定のフィルタ基準を満たし、処理中に特定のアクションを起動するために使用されるトランザクション・ログ内のレコード。「イベント・マーカー・システム」も参照してください。
例外マップ	エラー処理専用で使用される特別な MAP パラメータ。このパラメータは、エラーの後にのみ実行されて、エラー・データを例外表に送信します。
例外表	失敗した SQL 操作に関する情報が例外マップの結果として書き込まれるデータベース表。エラー処理で使用されます。
Extract	データソース、ソース表、ローカル証跡またはローカル・ファイルからデータを読み取る Oracle GoldenGate プログラム。Extract は、ターゲット・システムに配信するためにデータを処理します。*プライマリ Extract* は、データソースまたはデータベース表を読み取り、*データ・ポンプ Extract* は、プライマリ Extract によってデータを移入されたローカル証跡を読み取ります。
抽出ファイル	Oracle GoldenGate によって書き込まれるファイルであり、バッチ実行または初期ロードによる後続の処理を一時的に待機しているデータが格納されます。
抽出	後続の処理またはターゲット・データベースへの転送 (あるいはその両方) に備えてデータベース表またはデータソースからデータを読み取る処理。
フェッチ	トランザクション・ログのレコードを処理する場合に Extract プロセスによってデータベースに発行される問合せ。フェッチは、SQL 操作を完了するために必要なデータ値がレコードに存在しない場合に必要です。
ファイル・ヘッダー	「ヘッダー」を参照してください。
フィルタリング	抽出またはレプリケーションのためにデータを選択および除外するルールを使用すること。
関数	アプリケーションまたはルーチン内で実行できるコード部分。「*列変換関数*」も参照してください。
GGSCI	*GoldenGate ソフトウェア・コマンド・インタフェース*。Oracle GoldenGate を構成、制御および監視するコマンドを発行するための主要インタフェースです。
GLOBALS ファイル	Extract や Replicat などのプロセスに固有のランタイム・パラメータとは対照的に、Oracle GoldenGate インスタンスに全体的に適用されるパラメータを格納した Oracle GoldenGate のルート・ディレクトリにあるテキスト・ファイル。

用語	定義
グループ	*プロセス・グループ*とも呼ばれます。グループは、Oracle GoldenGate プロセス (Extract または Replicat) と、そのプロセスに関連するパラメータ・ファイル、チェックポイント・ファイルおよびその他のファイルで構成されます。
ヘッダー	ヘッダーには次の種類があります。 ◆ **レコード・ヘッダー :** レコードのトランザクション環境に関する情報を格納した Oracle GoldenGate の証跡ファイルに含まれるレコードの先頭部分にある領域。 ◆ **ファイル・ヘッダー :** 証跡の各ファイルの先頭部分または抽出ファイルの先頭部分にある領域。このヘッダーには、Oracle GoldenGate のリリースなど、ファイル自体の情報が含まれます。
異機種	データの交換が、異なるタイプのアプリケーション間、異なるタイプのデータベース間、異なるオペレーティング・システム間、またはこれらの組合せを対象に行われるデータ環境。
同機種	データの交換が、同一タイプのアプリケーション、データベースおよびオペレーティング・システム間で行われるデータ環境。
初期ロード	2 つのデータベースを同一にするためにソース・データをターゲット・データベースに複製すること。
中間システム	ソース・システムとターゲット・システム間の中継場所として機能するネットワーク上のシステム。このシステムは、トランスフォーメーションなどの追加処理アクティビティのホストに指定できます。
キー	表の行の一意の識別子として使用されるその表の 1 つ以上の列。Oracle GoldenGate では、ターゲット・データベースでの適切な行の検出とソース・データベースからのフェッチにこのキーが使用されます。Oracle GoldenGate でキーとして指定できるのは、主キー、一意キー、代替キー、または表のすべての列 (定義された識別子がない場合) です。
KEYCOLS	Oracle GoldenGate が表の特定の行を検索するために一意の識別子として使用する 1 つ以上の列を定義する TABLE 文または MAP 文の句。
KEYGEN	暗号化鍵を生成する Oracle GoldenGate ユーティリティ。
ラグ	Extract ラグは、Extract によってレコードが処理された時刻と、データソースにおけるそのレコードのタイムスタンプとの間の差異です。 Replicat ラグは、Replicat によって証跡の最後のレコードが処理された時刻と、証跡におけるそのレコードのタイムスタンプとの間の差異です。
待機時間	変更がソース・データで発生した時点と、その変更がターゲット・データに反映された時点との間の時間的差異。

用語	定義
LOB	*ラージ・オブジェクト*。大きすぎて文字フィールドに収まらない非構造化オブジェクト (Microsoft Word 文書や映像 / 音声ファイルなど) を表すデータベースのデータ型です。LOB のサブセットとして、文字データを格納する CLOB(キャラクタ・ラージ・オブジェクト) と、バイナリ・データを格納する BLOB(バイナリ・ラージ・オブジェクト) があります。
ログベース抽出	データベースのトランザクション・ログからデータ変更を抽出する方法。
論理名	ストアド・プロシージャの*実行*インスタンスを表すそのプロシージャの名前。プロシージャの実際の名前と対比されます。たとえば、lookup という名前のプロシージャの論理名は、lookup1 や lookup2 などになります。
LUW	*Linux*、*UNIX*、*Windows*。これらのどのプラットフォームでも実行されるアプリケーションを説明する頭字語 (DB2 LUW など)。
マクロ	パラメータおよびコマンドの実装などのタスクを自動化するコンピュータ・プログラム。
Manager	Oracle GoldenGate 処理のための制御プログラム。
マップ	ソース・データのセットとターゲット・データのセットとの間の対応付け。マップには、データの選択基準と変換基準を含めることができます。これらのマップは、Replicat の MAP パラメータで指定します。
MAP 文	ソース表とターゲット表間の関係と、それらの表の処理ルールを指定する Replicat パラメータ。
マーカー	Extract および Replicat の処理に関連してアプリケーション固有のイベントを識別するために、NonStop Server の監査証跡に挿入されるレコード。「イベント・マーカー・システム」も参照してください。
標準 Replicat 処理モード	Replicat のデフォルトの処理モード。標準モードでは、Replicat は、複数のソース・トランザクションによる操作を (トランザクション順に) 蓄積し、それらをターゲットの 1 つのトランザクション内のグループとして適用することでパフォーマンスを向上します。GROUPTRANSOPS パラメータによってこのトランザクション内の操作の数を制御できますが、その境界は、グループの最後のトランザクションによるすべての操作が含まれるように Replicat によって自動的に調整される可能性があります。「バッチ Replicat 処理モード」および「ソース Replicat 処理モード」も参照してください。
オブジェクト	このドキュメントにおいては、*オブジェクト*という語は、データの格納 (表など)、所有権および権限の定義 (ロールなど)、他のオブジェクトに対するアクションの実行 (トリガーなど) といった目的でユーザーが認識および作成できるデータベースの任意の論理コンポーネントを示します。
オブジェクト・レコード	Oracle GoldenGate で処理するために構成された表および他のデータベース・オブジェクトの属性 (列 ID やデータ型など) を格納したファイル。

用語	定義
ODBC	*Open Database Connectivity*。アプリケーションが統一された方法で異なるタイプのデータベースに接続できるようにする標準インタフェースの頭字語です。ODBC の目的は、データベースに接続するプロセスをプログラミング言語、データベース・システムおよびオペレーティング・システムから分離することです。
オンライン変更同期	Extract プロセスと Replicat プロセスが、Oracle GoldenGate ユーザーによって停止されないかぎり、データ変更を同期するために継続的に実行される Oracle GoldenGate の処理方法。オンライン・プロセスでは、証跡にチェックポイントが保持されます。
オンライン Extract	オンライン変更同期のために構成された Extract グループ。
オンライン処理	「*オンライン変更同期*」を参照してください。
オンライン Replicat	オンライン変更同期のために構成された Replicat グループ。
操作	単一の作業単位。通常は、データに行われる SQL 変更またはデータベースのオブジェクト構造に行われる変更を示しますが、コンピュータ・プロセスによって実行される任意の作業を示すこともあります。
Oracle GoldenGate Director	Oracle GoldenGate ユーザーによる Oracle GoldenGate プロセスの監視および管理を可能にするグラフィカル・ユーザー・インタフェース・ソフトウェア。Oracle GoldenGate Director には、次のコンポーネントがあります。 **Oracle GoldenGate Director Administrator:** 管理者が Oracle GoldenGate のユーザーおよびインスタンスを定義するために使用するユーティリティ。 **Oracle GoldenGate Director Server:** Oracle GoldenGate プロセスに関するデータを収集するソフトウェア・モジュール。 **Oracle GoldenGate Director Client:** Oracle GoldenGate Director に対するインタフェースとしてユーザーのシステムにインストールされるソフトウェア。 **Oracle GoldenGate Director Web:** Oracle GoldenGate Director に対するブラウザベースのユーザー・インタフェース (ソフトウェアのインストール不要)。
Oracle GoldenGate ロールバック	変更前イメージを使用して、データベースに加えられた変更を元に戻すユーティリティ。
上書きモード	10.0 より前のバージョンの Oracle GoldenGate で使用されていた証跡へのデータの書込み方法。このモードでは、Extract は、リカバリ時に証跡ファイルの最後にデータを追加するのではなく、既存のデータを上書きします。
所有者	データベース・オブジェクトが組織階層の一部として割り当てられるデータベースの論理ネームスペース。データベース・オブジェクトの所有権はデータベース・タイプごとに異なる方法で管理されるため、このドキュメントで使用される*所有者*という語は、オブジェクト名の修飾子としてデータベースによって認識されるすべてのエンティティ (通常はユーザー名またはスキーマ名) を示します。たとえば、修飾された Oracle 表名の scott.emp において、所有者は scott です。

用語	定義
パラメータ	コンピュータ・プログラム (Oracle GoldenGate のようなアプリケーション、ストアド・プロシージャ、マクロ、スクリプトまたは他の処理命令のコードなど) の入力値または出力値。
パラメータ・ファイル	Oracle GoldenGate プロセスの動作を制御するパラメータを格納したファイル。パラメータ・ファイルのデフォルトの場所は、Oracle GoldenGate のインストール・ディレクトリの dirprm ディレクトリです。
パススルー・データ・ポンプ	データ定義を参照する必要性を避けるために、PASSTHRU パラメータで構成されたデータ・ポンプ。これにより、処理が高速化し、データベースが存在しない中間システムでポンプを使用できます。
パススルー Extract	「*パススルー・データ・ポンプ*」を参照してください。
パッシブ Extract	エイリアス Extract がターゲットで使用されている場合に、ソース・システムで動作する Extract プロセス。この Oracle GoldenGate 構成が必要になるのは、ターゲットがよりセキュアなネットワーク・ゾーン内に存在するためにセキュリティ・ルールで (通常の Extract が行うような) ソース・システムからの TCP/IP 接続の開始が許可されない場合です。パッシブ Extract は、使用中はデータ・ポンプになります。それ以外の場合、プライマリ Extract になります。
プライマリ Extract	データソースから、または直接データベース表から読取りを行う Extract グループ。プライマリ Extract は、後からデータ・ポンプ Extract によって読み取られるローカル証跡に書込みを行うことができます。または、TCP/IP を通じてデータをターゲット・システムに送信できます。
主キー	現在および将来の表に存在する (可能性のある) すべての行を一意に識別する 1 つ以上の列で構成された整合性制約。1 つの表には 1 つの主キーのみを指定できます。主キーには、暗黙的な NOT NULL 制約が含まれます。
プロセス・レポート	プロセス構成と実行時の統計およびイベントに関する情報を提供する、Extract、Replicat および Manager に対して生成されるレポート。プロセス・レポートのデフォルトの場所は、Oracle GoldenGate のインストール・ディレクトリの dirrpt ディレクトリです。
レコード	データベースの行に対して実行された単一の SQL 操作に関する情報を格納するトランザクション・ログまたは証跡の情報単位。*レコード*という語は、表の特定の行に含まれる情報を説明する場合にも使用されます。
レコード・ヘッダー	「ヘッダー」を参照してください。
リモート・ファイル	リモート・システムの抽出ファイル。
リモート証跡	リモート・システムの証跡。
Replicat	データをターゲット表に適用するか、データを別のアプリケーションまたは宛先に移動する Oracle GoldenGate プロセス。
レプリケーション	ソース・データベースの操作を再作成してターゲット・データベースに適用するプロセス。

用語	定義
レポート	「*プロセス・レポート*」を参照してください。
レポート・ファイル	「*プロセス・レポート*」を参照してください。
ロールバック	コミットされていないトランザクション内の SQL 文によって実行されたデータ変更を元に戻すアクション。
ロールオーバー	一連のファイル内 (証跡など) の 1 つのファイルを閉じ、同じ一連のファイル内の新規ファイルを開くこと。
ルーチン	値の取得と返却を行ってレスポンスを戻す関数をコールする、Oracle GoldenGate などのアプリケーション内で実行されるコード部分。「*ユーザー・イグジット*」も参照してください。
行	データベース表内に格納されるエンティティ (従業員など) の単一のインスタンスに関する情報。たとえば、John Doe に関する情報は、1 つの行に格納されますが、その行は会社の John およびその他の従業員に関する情報を格納するより広い範囲の行のコレクションに含まれます。行は、一般的に *レコード*とも呼ばれます。
ソース	Oracle GoldenGate が抽出を行う元のデータの場所 (*ソース・データベース*や *ソース・システム*など)。
ソース定義ファイル	ソース表の定義を格納したファイルであり、ターゲット・システムに転送されます。このファイルは、ソース表とターゲット表が異なる場合に、データ変換のために Replicat プロセスによって使用されます。
ソース Replicat 処理モード	ソース処理モードでは、Replicat は、ソースで使用された範囲と同じトランザクション境界内で SQL 操作を適用します。「標準 Replicat 処理モード」も参照してください。
特別実行	「*バッチ実行*」を参照してください。
文	コンピュータ・プログラミング言語における基本命令 (SQL 文、パラメータ文、コマンド文など)。
静的 Collector	Manager プロセスによって自動的に起動されるかわりに、Oracle GoldenGate ユーザーによって手動で起動される Collector プロセス。
ストアド・プロシージャ	データベースに格納され、ビジネス・ルールの適用、アプリケーション・ロジックの追加、または他の必要な作業の実行のためにプロセスまたはアプリケーションによって必要時にコールされる SQL、PL/SQL または Java の文のグループ。
代替キー	表の行を一意に識別できるその表内の任意の列で構成された一意識別子。代替キーは、表の定義では設定されません。TABLE 文または MAP 文に KEYCOLS 句を指定することで作成します。

用語	定義
同期	2 つ以上のデータセットの一貫性を相互に確立または維持するプロセス。一貫性を確保するため、一方のセットは、他方と同一になるか、他方を再編成、再フォーマットまたは拡張したバージョンになりますが、情報それ自体の本質は維持されます。
表	行および列で構成されたデータベースの記憶域の論理単位。行と列の組合せによって、特定のエンティティ (従業員など) のインスタンスと、そのエンティティの属性 (名前や住所など) が識別されます。
TABLE 文	データベースからデータを抽出する 1 つ以上のソース表を指定する Extract パラメータ。
Teradata アクセス・モジュール (TAM)	Teradata データベースの Change Data Capture(CDC) コンポーネントと Extract プロセス間のインタフェース。これにより、Oracle GoldenGate は、Teradata レプリケーション・コンポーネントと通信できます。
ターゲット	Oracle GoldenGate によって処理されるデータの宛先 (*ターゲット・データベース*や*ターゲット・システム*など)。
ターゲット定義ファイル	ターゲット表の定義を格納したファイル。このファイルは、ソース・システムに転送され、ソース表とターゲット表が異なる場合に、データ変換のために Extract プロセスによって使用されます。
タスク	Extract プロセスが、Collector プロセスまたは証跡を使用せずに、TCP/IP を通じて Replicat プロセスと直接通信する、特殊なタイプのバッチ実行。
トークン	Oracle GoldenGate の証跡ファイルに含まれるレコードのヘッダー部分に格納されたユーザー定義の情報。トークン・データを使用して、Oracle GoldenGate による情報の配信方法をカスタマイズできます。
トレース表	Oracle Database で Oracle GoldenGate が使用するために作成される特別な表。この表をパラメータ設定と組み合せて使用し、双方向同期構成でレプリケートされたデータがソースに戻されることを防止します。
証跡	後続の処理に備えて Oracle GoldenGate が一時的にデータを格納するディスク上の一連のファイル。Oracle GoldenGate は、オンライン変更同期のために証跡にチェックポイントを記録します。
トランザクション	開始および終了のトランザクション制御文のセット内で論理的な作業単位として実行される 1 つ以上の SQL 操作 (または文) のグループ。トランザクション内の各 SQL 文は、すべて一体として実行に成功する必要があります。それ以外の場合、どの文も実行できません。トランザクションは、データおよび構造の整合性を確保するデータベース機能のシステムの一部です。
トランザクション・ログ	データのリカバリまたはレプリケーションを行う目的で、データベースに対して実行されたすべての SQL 変更操作を記録する一連のファイル。

用語	定義
トランスフォーメーション	「変換」とも呼ばれます。ターゲットの表またはアプリケーションで必要とされる形式にソース・データを処理するプロセス (日付の変換や算術計算の実行など) です。Oracle GoldenGate 列変換関数を使用してトランスフォーメーションを実行できます。
単方向同期	データ変更がソースからターゲットへと一方向にレプリケートされる構成。双方向構成のように同じデータが変更されてソースに戻されることはありません。
一意キー	現在および将来の表に存在する (可能性のある) すべての行を一意に識別する 1 つ以上の列で構成された整合性制約。暗黙的な NOT NULL 制約が含まれない点で、主キーとは異なります。1 つの表には複数の一意キーを指定できます。
作業単位	データベースで論理的な単位として実行されるデータ操作のセット。すべての操作が成功する必要があり、それ以外の場合はどの操作も実行できません。IBM の用語では、*作業単位*という語は、他のタイプのデータベースにおけるトランザクションという語と同義です。
ユーザー・イグジット	カスタム処理 (データの変換、データベース・イベントに対するレスポンス、無効なデータの修復など) を実行するために Oracle GoldenGate の処理中にコールされる C プログラミング・コードで記述されたユーザー作成プログラム。
ベンダー・アクセス・モジュール (VAM)	特定の種類のデータベースと通信するために Oracle GoldenGate プロセス・モジュールによって使用される API インタフェース。
VAM 証跡	トランザクション・ログと同様に、必要に応じて自動的に作成およびエージングされる一連のファイル。同時トランザクションによるデータ操作は、発生した順に時系列に記録されますが、必ずしもトランザクション順ではありません。Teradata の最大保護コミット・プロトコルをサポートするために使用されます。
ワイルドカード	不明または未指定の文字 (文字セット) のプレースホルダ。ワイルドカードは、パラメータまたはコマンド文で複数の名前を指定する手段です。Oracle GoldenGate では、任意の数の不明な文字を表すアスタリスク・ワイルドカード (*) がサポートされます。

索引

E

F

G

H

I

K

L

M

Q

R

S

T

U

V

W

ア

イ

ウ

エ

オ

力

キ

ク

ケ

コ

サ

シ

ス

セ

ソ

タ

チ

ツ

テ

ト

ネ

ハ

ヒ

フ

ヘ

ホ

マ

メ

モ

ユ

ラ

リ

ル

レ

ロ

ワ