Oracle® GoldenGate

Windows and UNIX 管理者ガイド

11*g* リリース 2 パッチ・セット 1 (11.2.1.0.1)

B69762-01 (原本部品番号 : E29397-01)

2012 年 11 月

Oracle GoldenGate Windows and UNIX 管理者ガイド 11g リリース 2 パッチ・セット 1 (11.2.1.0.1)

B69762-01 (原本部品番号 : E29397-01)

目次

はじめに **Oracle GoldenGate のドキュメントについて** 7
このマニュアルの表記規則 8
Oracle GoldenGate のヘルプの取得 8

第 1 章 Oracle GoldenGate の概要 10
Oracle GoldenGate でサポートされる処理方法およびデータベース 10
Oracle GoldenGate アーキテクチャの概要 11
プロセス・タイプの概要 17
グループの概要 17
コミット順序番号 (CSN) の概要 18

第 2 章 Oracle GoldenGate のグローバリゼーション・サポート 19
キャラクタ・セットの維持 19
Unicode およびネイティブ文字の使用 20

第 3 章 Manager およびネットワーク通信の構成 21
Manager プロセスの概要 21
Manager へのローカル通信用ポートの割当て 21
ファイアウォール経由でのリモート接続に使用するポートの管理 21
インターネット・プロトコルの選択 22
推奨される Manager パラメータ 23
Manager パラメータ・ファイルの作成 23
Manager の起動 24
Manager の停止 25

第 4 章 Oracle GoldenGate プロセス・インタフェースのスタート・ガイド 26
GGSCI コマンドライン・インタフェースの使用 26
UNIX バッチ・スクリプトおよびシェル・スクリプトの使用 28
Oracle GoldenGate のパラメータ・ファイルの使用 28
オブジェクト名でサポートされる文字 36
Oracle GoldenGate の入力におけるオブジェクト名の指定 38
パラメータ・ファイルにおける名前およびリテラルの SQL-92 ルールの適用 43

第 5 章　ライブ・レポートのための Oracle GoldenGate の使用 44
レポート構成の概要 44
レポート構成を選択する場合の考慮事項 45
標準レポート構成の作成 46
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 48
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 51
カスケード・レポート構成の作成 56
第 6 章　リアルタイム・データ分散のための Oracle GoldenGate の使用 64
データ分散構成の概要 64
データ分散構成の考慮事項 64
データ分散構成の作成 66
第 7 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 70
データ・ウェアハウス構成の概要 70
データ・ウェアハウス構成の考慮事項 70
データ・ウェアハウス構成の作成 72
第 8 章　ライブ・スタンバイ・データベース管理のための Oracle GoldenGate の構成 77
ライブ・スタンバイ構成の概要 77
ライブ・スタンバイ構成の考慮事項 78
ライブ・スタンバイ構成の作成 79
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 86
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 89
第 9 章　アクティブ / アクティブ型高可用性のための Oracle GoldenGate の構成 93
アクティブ / アクティブ構成の概要 93
アクティブ / アクティブ構成の考慮事項 93
データ・ループの防止 96
アクティブ / アクティブ構成の作成 99
競合の管理 104
Oracle GoldenGate CDR 機能による競合処理 105
Oracle GoldenGate CDR の構成 106
CDR の例 1: USEMAX、OVERWRITE、DISCARD によるすべての競合タイプの処理 113
CDR の例 2: USEDELTA および USEMAX による UPDATEROWEXISTS の処理 119
CDR の例 3: USEDELTA、USEMAX および IGNORE による UPDATEROWEXISTS の処理 122

第 10 章　Oracle GoldenGate のセキュリティの構成 125
Oracle GoldenGate のセキュリティ・オプションの概要 125
証跡またはファイルの暗号化 126
データベース・ユーザー・パスワードの暗号化 130
TCP/IP を通じて送信されるデータの暗号化 133
暗号化鍵の生成 134
コマンド・セキュリティの構成 135
ターゲット・システムからの接続開始の使用 137
第 11 章　データのマッピングおよび操作 141
サポートの制限 141
マッピングおよびデータ統合を制御するパラメータ 141
異なるデータベース間のマッピング 141
データのマッピングおよび変換の実行場所の決定 142
データ・マッピング時のグローバリゼーションに関する考慮事項 142
列のマッピング 145
行の選択 152
変更前の値の取得 157
列の選択 157
SQL 操作の選択および変換 157
トランザクション履歴の使用 158
データのテストおよび変換 159
トークンの使用 165
第 12 章　Oracle GoldenGate の処理エラーへの対処 167
Oracle GoldenGate のエラー処理の概要 167
Extract エラーの処理 167
DML 操作中の Replicat エラーの処理 167
DDL 操作中の Replicat エラーの処理 171
TCP/IP エラーの処理 171
更新されたエラー・メッセージの管理 172
Oracle GoldenGate エラーの解決 172
第 13 章　レプリケートされたデータとメタデータとの関連付け 173
同一のメタデータを想定するための Oracle GoldenGate の構成 173
異なるメタデータを想定するための Oracle GoldenGate の構成 174
同じ定義と異なる定義の組合せを使用するための Oracle GoldenGate の構成 181

第 14 章　オンライン変更同期の構成 183
オンライン変更同期の概要 183
特定のトポロジに適合させるための変更同期の構成 184
グループのネーミング規則 184
チェックポイント表の作成 184
オンライン Extract グループの作成 186
証跡の作成 189
オンライン抽出用のパラメータ・ファイルの作成 191
オンライン Replicat グループの作成 193
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 194
オンライン処理の制御 196
プロセス・グループの削除 198
第 15 章　初期データ・ロードの実行 199
初期データ・ロード方法の概要 199
初期ロードでのパラレル処理の使用 200
初期ロードの前提条件 200
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 202
ファイルから Replicat へのデータのロード 203
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 209
Oracle GoldenGate ダイレクト・ロードを使用したデータのロード 214
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 219
Teradata ロード・ユーティリティを使用したデータのロード 223
第 16 章　Oracle GoldenGate 処理のカスタマイズ 225
カスタム処理の概要 225
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 225
Oracle GoldenGate マクロを使用した作業の簡略化および自動化 232
ユーザー・イグジットを使用した Oracle GoldenGate 機能の拡張 238
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 244
第 17 章　Oracle GoldenGate 処理の監視 249
Oracle GoldenGate 監視ツールの概要 249
GGSCI での情報コマンドの使用 249
Extract リカバリの監視 250
ラグの監視 251
処理量の監視 252

エラー・ログの使用 254
プロセス・レポートの使用 255
廃棄ファイルの使用 257
システム・ログの使用 258
時間の差異の調整 259
NonStop システムへのイベント・メッセージの送信 259
監視およびチューニングに関するヘルプの取得 260

第 18 章　管理操作の実行 261

管理操作の概要 261
アプリケーション・パッチの実行 261
プロセス・グループの追加 262
トランザクション・ログの初期化 269
システムの停止 270
データベース属性の変更 270
証跡ファイルのサイズの変更 276

第 19 章　リバース・ユーティリティによるデータ変更の取消し 278

リバース・ユーティリティの概要 278
リバース・ユーティリティの制限 279
リバース・ユーティリティの構成 279
反転処理のためのプロセス・グループおよび証跡の作成 282
リバース・ユーティリティの実行 284
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 285

付録 1　サポートされるキャラクタ・セット 286

付録 2　サポートされるロケール 294

付録 3　Oracle GoldenGate 証跡について 297

証跡リカバリ・モード 297
証跡ファイルのヘッダー・レコード 297
証跡のレコード形式 298
Oracle GoldenGate レコードの例 298
レコード・ヘッダー領域 300
レコード・データ領域 302
トークン領域 303
Oracle GoldenGate の操作タイプ 304
Oracle GoldenGate 証跡のヘッダー・レコード 307

付録 4　　コミット順序番号について 309
用語集 312
索引 325

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 Monitor 管理者ガイド*』:Oracle GoldenGate レプリケーション・コンポーネントを監視するために *Oracle GoldenGate Monitor* 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 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 および Oracle GoldenGate Monitor

Oracle GoldenGate グラフィカル・クライアント・インタフェースのヘルプを取得するには、各アプリケーション内で**「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

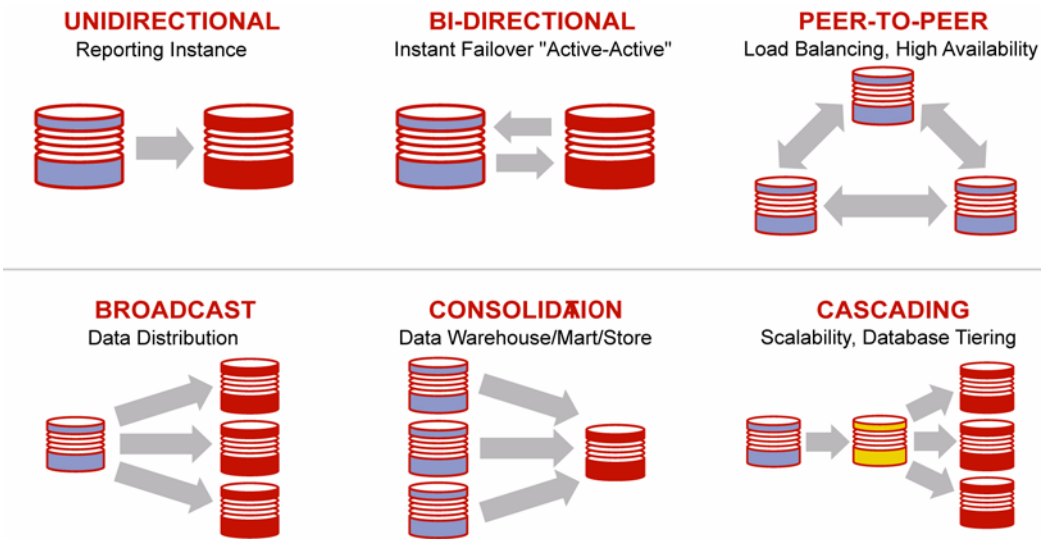
Oracle GoldenGate でサポートされる処理方法およびデータベース

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)
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

[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、次の目的で構成できます。

- あるデータベースからデータ・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、トランザクション DML[1]操作と DDL 変更 (サポートされるデータベースの場合) を継続的に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。

Oracle GoldenGate は、次のコンポーネントで構成されます。

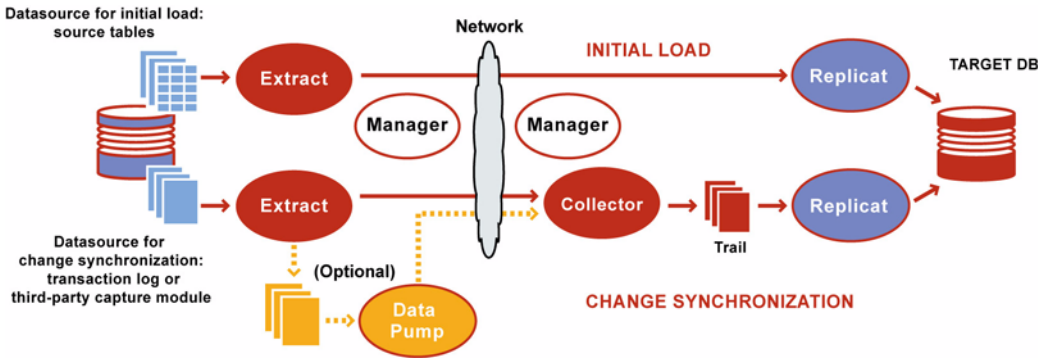
- Extract
- データ・ポンプ

[1] INSERT、UPDATE、DELETE

- 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 DML 操作および DDL 操作の同期に対応する 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静的データセットを直接抽出 (取得) 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 DML 操作および DDL 操作を取得します。

Extract が取得するのは次のいずれかの*データソース*からです。

- ソース表 (実行するのが初期ロードの場合)。
- *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、データ操作を抽出して Extract に渡すコンポーネントが用意されます。

変更同期用に構成されている場合、Extract は、Extract 構成内のオブジェクトに対して実行される DML 操作および DDL 操作を取得します。Extract は、操作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、それらの操作を格納します。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操作は破棄されます。Extract がコミットを受信すると、トランザクションはディスクの*証跡*と呼ばれる一連のファイルに永続化され、ターゲット・システムへの伝

播を待つキューに入ります。各トランザクションの操作はすべて、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、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 つの永続性証跡を使用する)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 Extract プロセスどうしを区別するには、各プロセスにグループ名を割り当てます (17 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、取得したデータ操作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。一方で、データ・ポンプを使用す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、データ操作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。

一般に、データ・ポンプは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*でデータ・ポンプを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操作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、Extract はメモリーを使い果た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、抽出したデータ操作を各ソース・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。

Replicat の概要

Replicat プロセスは、ターゲット・システムで実行されるもので、そのシステムの証跡を読み取り、DML 操作または DDL 操作を再構成してターゲット・データベースに適用します。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、静的データ・コピーをターゲット・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 :** 変更同期用に構成されている場合、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、レプリケートされたソース操作をターゲット・オブジェクトに適用します。データ整合性を維持するため、Replicat は、レプリケートされた操作を、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。データ整合性を維持するため、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。Replicat プロセスどうしを区別するには、各プロセスにグループ名を割り当てます (17 ページの「グループの概要」を参照)。

Replicat は、レプリケートされた操作をターゲット・データベースに適用する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、取得した変更のレコードをディスク上の*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、中間システム、ターゲット・システムのいずれか、またはこれらを組み合せた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 (15 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、データを処理して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 とデータ・ポンプ Extract が証跡に書き込みます。証跡に書き込めるのは一度に 1 つの Extract プロセスのみであり、各 Extract が証跡にリンクしている必要があります。

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: 前の Extract(通常はプライマリ Extract) にリンクしているローカル証跡から DML 操作および DDL 操作を抽出し、必要に応じてさらに処理を行い、後続の Oracle GoldenGate プロセス (通常は Replicat だが、必要に応じて別のデータ・ポンプの場合もある) によって読み取られる証跡にデータを転送します。
- Replicat: 証跡を読み取り、レプリケートされた DML 操作および DDL 操作をターゲット・データベースに適用します。

証跡の作成およびメンテナンス

証跡ファイル自体は処理中に必要に応じて作成されますが、ADD RMTTRAIL コマンドまたは ADD EXTTRAIL コマンドで Oracle GoldenGate 構成に追加するときに、2 文字の証跡名を指定します。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。

証跡ファイル全体が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。新しいファイルを作成すると、各ファイルが 2 文字の証跡名を継承し、それに一意の 6 桁の順序番号 (000000 ～ 999999) が追加されます (たとえば、c:\ggs\dirdat\tr000001 のようになります)。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。

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。

証跡と証跡に含まれるレコードの詳細は、297 ページの付録 3 を参照してください。

抽出ファイルの概要

一部の構成では、Oracle GoldenGate は抽出したデータを証跡ではなく*抽出ファイル*に格納します。抽出ファイルは、単一のファイルとすることも、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。チェックポイントは、同期のためにマークされたデータ変更が、実際に Extract により取得されて Replicat によりターゲットに適用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。Extract はコミットされたトランザクションのみを取得するので、オープン・トランザクションのいずれかがコミットされた場合に備えて、すべてのオープン・トランザクションの操作を追跡する必要があります。そのため、トランザクション・ログ内で現在読取りを行っている場所を示すチェックポイントと、最も古いオープン・トランザクションの開始位置 (現在のログまたは以前のいずれかのログ内) を記録することが求められます。

停止後に再処理が必要になるトランザクション・ログの量を抑えるため、Extract は、処理の現在の状態およびデータ (長時間実行トランザクションの状態およびデータがあればそれも含む) を特定の間隔でディスクに永続化します。この間隔の後、Extract が停止した場合は、直前の間隔における位置または最後のチェックポイントからリカバリを実行できるので、最も古いオープンの長時間実行トランザクションが最初に現れたログ位置まで戻る必要はありません。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。

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。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。チェックポイント表を使用すると、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ので、データベース・リカバリ後の一貫性が保証されます。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。

必要に応じて開始ポイントから再実行できる非継続的なタイプの構成 (初期ロードなど) の場合、チェックポイントは不要です。

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 プロセスの起動
- 動的プロセスの起動
- プロセスのポート番号の管理
- 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

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 プロセスおよび TCP/IP 接続の構成の詳細は、第 3 章を参照してください。

Collector の概要

Collector は、継続的なオンライン変更同期がアクティブである場合に、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。Collector は次の処理を実行します。

- リモート Extract から Manger への接続リクエストが発生すると、スキャンを行って使用可能なポートにバインドし、リクエスト元の Extract プロセスに割り当てるポート番号を Manager に送信します。

- Extract によって送信された抽出済のデータベース変更を受信し、証跡ファイルに書き込みます。ネットワーク接続が必要なときは Manager が Collector を自動的に起動するので、Oracle GoldenGate ユーザーが Collector を操作する必要はありません。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、使用する Extract ごとに Collector が 1 つ存在します。Collector は、関連する Extract プロセスが終了すると終了します。

  注意 Collector は、必要に応じて手動で実行できます。これは、通常の *動的* Collector と対比して、*静的* Collector と呼ばれます。複数の Extract プロセスで 1 つの静的 Collector を共有できますが、1 対 1 の比率が最適です。静的 Collector を使用すると、プロセスを確実に特定のポートで動作させることができます。静的 Collector 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。Manager によるポート割当て方法の詳細は、第 3 章を参照してください。

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。この構成の詳細は、137 ページを参照してください。

プロセス・タイプの概要

Oracle GoldenGate は、要件に応じて、次の処理タイプでの構成が可能です。

- *オンライン* Extract プロセスまたは Replicat 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。ソース・オブジェクトとターゲット・オブジェクトの同期を維持するために、DML 操作および DDL 操作 (サポートされる場合) を継続的に抽出してレプリケートするには、オン ライン・プロセスを使用します。このプロセス・タイプには、EXTRACT パラメータと REPLICAT パラメータが適用されます。
- *ソース表* Extract プロセスは、別のデータベースへの初期ロードの準備として、現在の静的データセットをソース・オブジェクトから直接抽出します。このプロセス・タイプでは、チェックポイントは使用されません。このプロセス・タイプには、SOURCEISTABLE パラメータが適用されます。
- *特別実行* Replicat 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データを適用します。Replicat の特別実行は初期データ・ロードで使用しますが、オンライン Extract と一緒に使用して、証跡内のデータ変更をバッチ形式で (継続的にではなく、1 日 1 回など) 適用することもできます。このプロセス・タイプでは、同じ開始ポイントと終了ポイントを使用して最初から実行しなおせるため、チェックポイントは保持されません。このプロセス・タイプには、SPECIALRUN パラメータが適用されます。
- *リモート・タスク*は、特殊なタイプの初期ロード・プロセスであり、Extract が TCP/IP 経由で直接 Replicat と通信します。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。このタスクは、RMTTASK パラメータを使用して Extract パラメータ・ファイルに定義します。

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。使用可能なグループ名については、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』の各コマンドの説明を参照してください。

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。これにより、トランザクションがデータベースにコミットされた時点を一意に識別します。

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 の詳細とデータベースごとの CSN 値のリストは、309 ページの付録 4 を参照してください。

第 2 章

Oracle GoldenGate のグローバリゼーション・サポート

Oracle GoldenGate のグローバリゼーション・サポートにより、ネイティブ言語のエンコーディングでのデータ処理が可能になります。このサポートには次のような特長があります。

キャラクタ・セットの維持

データをネイティブ言語のエンコーディングで処理するために、Oracle GoldenGate では、パラメータ・ファイルの解析とコマンドライン・インタープリタの処理を行う際に、データベースのキャラクタ・セット・エンコーディングとオペレーティング・システム・ロケールが考慮されます。

データベース構造メタデータのキャラクタ・セット

Oracle GoldenGate では、カタログ、スキーマ、表および列の名前は、ソース・データベースおよびターゲット・データベースのキャラクタ・セット・エンコーディングに従って、ネイティブ言語で処理されます。この処理は、パラメータ・ファイルとコマンド・インタープリタにも適用されます (それらはオペレーティング・システム・ロケールに従って処理されます)。これらのオブジェクトは、クライアント・インタフェース全般、コンソールおよびファイルに、ローカライズ形式で表示されます。

文字型データのキャラクタ・セット

Oracle GoldenGate の適用プロセス (Replicat) では、文字型の列にデータが含まれている場合に、あるキャラクタ・セットから別のキャラクタ・セットへのデータ変換がサポートされます。キャラクタ・セット変換のサポートは、TABLE 文または MAP 文の COLMAP 句または USEDEFAULTS 句に対して実行される列から列へのマッピングに限定されています。列変換関数、SQLEXEC または TOKENS 機能ではサポートされません。

キャラクタ・セット、キャラクタ・セット間の変換、およびデータ・マッピングの詳細は、141 ページの「データのマッピングおよび操作」を参照してください。

データベース接続のキャラクタ・セット

Extract プロセスおよび Replicat プロセスは、データベースへの接続時にセッション・キャラクタ・セットを使用します。Oracle の場合、セッション・キャラクタ・セットは Extract プロセスのデータベース・キャラクタ・セットと同様に設定されており、Replicat プロセスの NLS_LANG を使用して決定されます。Sybase、Teradata および MySQL の場合、セッション・キャラクタ・セットは SOURCEDB および TARGETDB の SESSIONCHARSET オプション、または GLOBALS ファイルでグローバルに設定される SESSIONCHARSET パラメータから取得されます。その他のデータベース・タイプの場合は、プログラムによって取得されます。また、Oracle GoldenGate プロセスは、Oracle GoldenGate とデータベースの間

の通信やデータ転送 (SQL 問合せ、フェッチ、データ適用など) にもセッション・キャラクタ・セットを使用します。

テキスト入力およびテキスト出力のキャラクタ・セット

Oracle GoldenGate では、次の場所で、ホスト・オペレーティング・システムのデフォルト・キャラクタ・セットでのテキスト入力およびテキスト出力がサポートされます。

- コンソール
- コマンドラインでの入力および出力
- FORMATASCII、FORMATSQL、FORMATXML の各パラメータ、パラメータ・ファイル、データ定義ファイル、エラー・ログ、プロセス・レポート、廃棄ファイルなどのテキスト・ファイル、および Oracle GoldenGate ユーザーが Oracle GoldenGate 環境の構成、実行、監視に使用するその他の判読可能なファイル。

プラットフォームで、必要なキャラクタ・セットがオペレーティング・システムのデフォルトとしてサポートされていない場合は、次のパラメータを使用してキャラクタ・セットを指定できます。

- CHARSET パラメータ (プロセスがパラメータ・ファイルの読取りに使用するキャラクタ・セットを指定する場合)。
- DEFSFILE パラメータの CHARSET オプション (特定のキャラクタ・セットでデータ定義ファイルを生成する場合)。

GGSCI コマンド・コンソールは、キーボードと OBEY の入力、およびコンソール出力に関して、常にローカル・オペレーティング・システムのキャラクタ・セットで動作します。

Unicode およびネイティブ文字の使用

Oracle GoldenGate では、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文字を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、オペレーティング・システムで必要な文字がサポートされていない場合に、または必要に応じて他の目的で使用できます。詳細は、144 ページの「特定の文字に対するエスケープ・シーケンスの使用」を参照してください。

第 3 章

Manager およびネットワーク通信の構成

この章では、次の操作の手順について説明します。

- Manager プロセスの構成。
- ローカルおよびリモートのネットワーク通信に使用するポートの指定。Oracle GoldenGate のポートはすべて構成可能です。

Manager プロセスの概要

Oracle GoldenGate を構成して実行するためには、Oracle GoldenGate のソース・システムとターゲット・システムのすべて、および中間システム (構成内で使用している場合) で Manager プロセスを実行する必要があります。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
- Collector プロセスの起動
- プロセスのポート番号の管理
- 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 へのローカル通信用ポートの割当て

Manager プロセスは、Oracle GoldenGate のインストール環境ごとに、他のローカル Oracle GoldenGate プロセスと通信を行うための専用ポートを必要とします。このポートを指定するには、Manager パラメータ・ファイルの PORT パラメータを使用します。次のガイドラインに従ってください。

- Manager のデフォルト・ポート番号は 7809 です。デフォルト・ポート番号 (使用可能であれば、こちらを推奨) または任意の別のポートを指定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。
- 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。

ファイアウォール経由でのリモート接続に使用するポートの管理

Oracle GoldenGate のターゲットの場所でファイアウォールを使用している場合、リモート Oracle

GoldenGate プロセスからの動的 TCP/IP 通信を受信するには、ターゲット・システムに追加のポートが必要です。具体的には次のポートです。

- リモートのオンライン Extract プロセスから伝播されたトランザクション・データを受信するためにローカルの Manager が起動する *各* Collector プロセス用に 1 ポート。Extract プロセスがターゲットにデータを送信すると、ターゲットの Manager が専用の Collector プロセスを起動します。
- ローカルの Manager がリモート・タスクの一環として起動する *各* Replicat プロセス用に 1 ポート。リモート・タスクは初期ロードに使用されるもので、RMTTASK パラメータで指定します。このポートは、リモート Extract プロセスからの着信リクエストを受信するために使用されます。
- ローカル Oracle GoldenGate 構成の拡張に必要な場合は、追加で数ポート。
- 他の Oracle GoldenGate 製品が Oracle GoldenGate のローカル・インスタンスとやり取りする場合は、その製品用のポート (各製品のドキュメントを参照)。

これらのポートを指定するには、Manager パラメータ・ファイルの DYNAMICPORTLIST パラメータを使用します。次のガイドラインに従ってください。

- 次の形式を自由に組み合せて、最大 5000 のポートを指定できます。

```
7830, 7833, 7835
7830-7835
7830-7835, 7839
```

- 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。
- システム上の Manager インスタンスごとに異なるポート・リストを使用する必要があります。

必須パラメータではありませんが、最高のパフォーマンスを得るために DYNAMICPORTLIST を使用することを強くお薦めします。Collector プロセスの役割は、使用可能なポートを見つけてバインドすることですが、有効なポートの既知のリストがあれば、このプロセスが迅速化されます。DYNAMICPORTLIST がない (または、DYNAMICPORTLIST に十分な数のポートが指定されていない) 場合、Collector はポート 7840 を使用してリモート・リクエストを行おうとします。7840 が使用できない場合、Collector は使用可能なポートが見つかるまで番号を 1 ずつ増やします。これが原因で、リモート・リクエストの受入れが遅れる可能性があります。Collector が DYNAMICPORTLIST リスト内のポートを使い果たすと、次の処理が発生します。

- Manager は、プロセス・レポートおよび Oracle GoldenGate ggserr ログにエラーを記録します。
- Collector は、Oracle GoldenGate tcperrs ファイルのルールに基づいて再試行します。tcperrs ファイルの詳細は、171 ページの「TCP/IP エラーの処理」を参照してください。

PORT および DYNAMICPORTLIST 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。

インターネット・プロトコルの選択

デフォルトでは、Oracle GoldenGate は接続が成功する可能性を最大限に高めるために、次の優先順位でソケットを選択します。

- IPv6 デュアルスタック
- IPv4(IPv6 デュアルスタックが使用できない場合)
- IPv6

ネットワーク内にデュアルスタック・モードをサポートしていない IPv6 ネットワーク機器がある場合、USEIPV6 パラメータを使用すると、Oracle GoldenGate にすべての接続で IPv6 を使用させることができます。これは、Oracle GoldenGate インスタンスのすべてのプロセスに適用される GLOBALS パラメータの 1 つです。USEIPV6 を使用するときは、接続障害が発生しないように、ネットワーク全体を IPv6 対応にする必要があります。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。

推奨される Manager 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。これらのパラメータとその他の Manager 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。

- 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。このパラメータはクラスタ構成では必須であり、システムの起動後ただちに Oracle GoldenGate アクティビティを開始する必要がある場合に便利です。(Manager が起動ルーチンの一部であることが必要です。) 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。このパラメータはクラスタ構成では必須ですが、どのような構成でも処理を確実に継続するのに便利です。
- 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ため、証跡ファイルが大量のディスク領域を消費する可能性があります。最適な結果を得るため、PURGEOLDEXTRACTS は Extract パラメータや Replicat パラメータとしてではなく、Manager パラメータとして使用してください。
- STARTUPVALIDATIONDELAY | STARTUPVALIDATIONDELAYCSECS: Manager がプロセスの実行ステータスを検証するまでの遅延時間を設定します。起動時に検証を行うことで、Oracle GoldenGate ユーザーは、エラー・メッセージやプロセス・レポートが生成される前に、失敗したプロセスを認識できます。
- USERID: Oracle GoldenGate の DDL サポートを使用する場合は必須です。

Manager パラメータ・ファイルの作成

必須のポート情報とオプションのパラメータで 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。Oracle GoldenGate のパラメータ・ファイルの詳細は、26 ページの第 4 章を参照してください。

注意 **Oracle GoldenGate がクラスタ内に位置する場合は、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 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てください。クラスタへの Oracle GoldenGate のインストールの詳細は、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。**

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 (GGSCI) 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. Manager プロセスに使用するパラメータを 1 行に 1 つずつ追加します。パラメータが複数行にわたる場合は、各改行の前にアンパサンド (&) を使用します。

4. ファイルを保存して閉じます。

例 次に示すのは、必須および推奨のパラメータを使用した UNIX システムでの Manager パラメータ・ファイルの例です。

```
PORT 7809
DYNAMICPORTLIST 7810-7820, 7830
AUTOSTART ER t*
AUTORESTART ER t*, RETRIES 4, WAITMINUTES 4
STARTUPVALIDATIONDELAY 5
PURGEOLDEXTRACTS /ogg/dirdat/tt*, USECHECKPOINTS, MINKEEPHOURS 2
```

これらのパラメータとその他の Manager 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。

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 コマンドライン・インタフェース。
- 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。

Manager をオペレーティング・システムのコマンド・シェルから起動する手順

```
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
```

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 および Linux(z/OS 上の USS を含む) 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。

- 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。
- 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 クラスタでは、クラスタ・アドミニストレータから 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 を停止しようとすると、クラスタ・モニターがその操作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。GGSCI を介して起動リクエストが何度も行われると、最終的には Manager クラスタ・リソースの起動しきい値を超えてしまい、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。

第 4 章

Oracle GoldenGate プロセス・インタフェースのスタート・ガイド

Oracle GoldenGate ユーザーは、次の方法でプロセスに指示を送ります。

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

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・インタフェースのグローバリゼーション・サポート

コマンド入力と関連のコンソール出力はすべて、ローカル・オペレーティング・システムのデフォルト・キャラクタ・セットで表示されます。ローカル・オペレーティング・システムのキャラクタ・セットと互換性のない文字を指定するには、UNICODE 表記法を使用します。たとえば、表の名前にユーロ記号が含まれている場合、次のコマンドは同等です。

```
ADD TRANDATA €1
ADD TRANDATA \u20AC1
```

詳細は、144 ページの「特定の文字に対するエスケープ・シーケンスの使用」を参照してください。

注意　Oracle GoldenGate のグループ名は大 / 小文字が区別されません。

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 ファイルでは、ローカル・オペレーティング・システムのキャラクタ・セットが認識されます。そのキャラクタ・セットと互換性のない文字を指定するには、UNICODE 表記法を使用します。詳細は、144 ページの「特定の文字に対するエスケープ・シーケンスの使用」を参照してください。

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。スクリプト・ファイルは、オペレーティング・システムのキャラクタ・セットでエンコードされている必要があります。オペレーティング・システムのキャラクタ・セットがサポートしていない文字には、UNICODE 表記法を使用できます。スクリプトを作成する前に、26 ページの「コマンド・インタフェースのグローバリゼーション・サポート」を参照してください。

スクリプトを入力する手順

オペレーティング・システムのコマンドラインから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 Oracle GoldenGate プロセスによって読み取られるプレーン・テキスト・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。

パラメータ・ファイルのグローバリゼーション・サポート

Oracle GoldenGate では、ローカル・オペレーティング・システムのデフォルト・キャラクタ・セットでパラメータ・ファイルが作成されます。ローカル・プラットフォームで、必要なキャラクタ・セットがオペレーティング・システムのデフォルトとしてサポートされていない場合は、CHARSET パラメータをグローバルに、またはプロセス単位で使用して、パラメータ・ファイルのキャラクタ・セットを指定できます。

キャラクタ・セットの非互換性による問題を回避するため、パラメータ・ファイルは、関連プロセスが実行されるサーバーで作成または編集します。あるシステム (Windows ラップトップなど) で作成したファイルを、Oracle GoldenGate がインストールされている、オペレーティング・システムのキャラクタ・セットが異なる UNIX サーバーに転送するのは避けてください。Oracle GoldenGate には、パラメータ・ファイルを別のシステムで作成する必要がある場合に、キャラクタ・セットの非互換性の問題を解決するためのツールがいくつか用意されています。

- CHARSET パラメータを使用すると、互換性のあるキャラクタ・セットをパラメータ・ファイルに指定できます。このパラメータは、指定したキャラクタ・セットでファイルを記述できるようにするもので、パラメータ・ファイルの 1 行目に置く必要があります。ファイルを別のシステムに転送したら、そのシステムでは編集しないでください。

- UNICODE 表記法を使用すると、ファイルが使用されるオペレーティング・システムのキャラクタ・セットと互換性のない文字を指定できます。144 ページの「特定の文字に対するエスケープ・シーケンスの使用」を参照してください。

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。

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・ファイルは、Oracle GoldenGate 環境に必要な場合と必要でない場合があります。

GLOBALS を使用する場合、GGSCI などの Oracle GoldenGate プロセスを起動する前にこのファイルが存在している必要があります。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、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。34 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。コメントの先頭にはハイフンが 2 つ付いています。

図 4 Extract のサンプル・パラメータ・ファイル

```
-- Extract group name
EXTRACT capt
-- Extract database user login, with password encryption specifications
USERID ogg, PASSWORD AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC &
    AES128, ENCRYPTKEY securekey1
-- Discard file
DISCARDFILE /ggs/capt.dsc, PURGE
-- Remote host to where captured data is sent in encrypted format:
RMTHOST sysb, MGRPORT 7809, ENCRYPT AES192 KEYNAME mykey
-- Encryption specification for trail data
ENCRYPTTRAIL AES192 KEYNAME mykey1
-- Remote trail on the remote host
RMTTRAIL /ggs/dirdat/aa
-- TABLE statements that identify data to capture.
TABLE FIN.*;
TABLE SALES.*;
```

前述の例は、大 / 小文字が区別されない Oracle データベースを反映したもので、オブジェクト名が TABLE 文に大文字で指定されています。大 / 小文字が区別されない Oracle データベースの場合、パラメータ・ファイルに名前をどのように入力しても (大文字でも、小文字でも、大文字と小文字が混在していても) 違いは生じません。他のデータベースの場合は、オブジェクト名の大 / 小文字が問題になることがあります。詳細は、38 ページの「Oracle GoldenGate の入力におけるオブジェクト名の指定」を参照してください。

図 5 Replicat のサンプル・パラメータ・ファイル

```
-- Replicat group name
REPLICAT deliv
-- Replicat database user login, with password encryption specifications
USERID ogg, PASSWORD AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC &
    AES128, ENCRYPTKEY securekey1
-- File containing definitions of source objects
SOURCEDEFS /ggs/dirdef/defs
-- Discard file
DISCARDFILE /ggs/deliv.dsc, PURGE
-- Decryption specification for encrypted trail
DECRYPTTRAIL AES192 KEYNAME mykey1
-- Error handling rules
REPERROR DEFAULT, ABEND
-- Ignore INSERT operations
IGNOREINSERTS
-- MAP statement to map source objects to target objects and
-- specify column mapping
MAP "fin"."accTAB", TARGET "fin"."accTAB",
COLMAP ("Account" = "Acct",
"Balance" = "Bal",
"Branch" = "Branch");
-- Get INSERT operations
GETINSERTS
-- MAP statement to map source objects to target objects and
-- filter to apply only the 'NY' branch data.
MAP "fin"."teller", TARGET "fin"."tellTAB",
WHERE ("Branch" = 'NY');
```

図 5 に示した Replicat のサンプルでの一重引用符と二重引用符の使用方法に注意してください。オブジェクト名の大 / 小文字を区別するには引用符が必要になるデータベース (Oracle など) の場合、大 / 小文字が区別されるオブジェクト名はパラメータ・ファイルでも二重引用符で囲む必要があります。大 / 小文字が区別される列名を二重引用符で囲んで指定するには、GLOBALS ファイルで USEANSISQLQUOTES パラメータを使用する必要があります。使用しないと、二重引用符で囲んだ文字列はリテラルとして解釈されます。名前およびリテラルの指定方法の詳細は、38 ページの「Oracle GoldenGate の入力におけるオブジェクト名の指定」を参照してください。

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。

GGSCI を使用するのは、パラメータ・ファイルをオペレーティング・システムのキャラクタ・セットで記述する (CHARSET パラメータを使用しない) 場合のみです。それ以外の場合は、GGSCI 外部のテキスト・エディタを使用します。CHARSET 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。

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
```

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 ogg, PASSWORD &
    AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC &
    AES128 KEYNAME mykey1
RMTHOST sysb, MGRPORT 8040, ENCRYPT AES192 KEYNAME mykey
ENCRYPTTRAIL AES 192 KEYNAME mykey2
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。ただし、次のものはセミコロンで終わるため、アンパサンドの指定は可能ですが必須ではなく、例外となります。

  MAP

  TABLE

  SEQUENCE

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

テキスト・エディタを使用して GGSCI の外部でパラメータ・ファイルを作成できますが、次の点に注意してください。

- 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。.prm というファイル拡張子を使用します。たとえば、extfin.prm や mgr.prm のようになります。
- 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」も参照してください。

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 17 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。

警告 キャラクタ・セットがローカル・オペレーティング・システムのものとは異なる既存のパラメータ・ファイル (CHARSET オプションを使用して別のキャラクタ・セットを指定したファイルなど) は、VIEW PARAMS を使用して表示しないでください。内容が破損する可能性があります。そのようなパラメータ・ファイルは、GGSCI の外部で表示してください。

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。

警告 ローカル・オペレーティング・システムのものとは異なるキャラクタ・セットを使用した既存のパラメータ・ファイル (CHARSET オプションを使用して別のキャラクタ・セットを指定したファイルなど) は、EDIT PARAMS コマンドを使用して表示や編集を行わないでください。内容が破損する可能性があります。そのようなパラメータ・ファイルは、GGSCI の外部で表示してください。

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 (*) および疑問符 (?) のワイルドカードを使用できます。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。詳細は、40 ページの「オブジェクト名でのワイルドカードの使用」を参照してください。

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。GLOBALS パラメータ・ファイルでは OBEY はサポートされていません。

OBEY パラメータを含むパラメータ・ファイルで CHARSET パラメータを使用する場合、参照先のパラメータ・ファイルは CHARSET のキャラクタ・セットを継承しません。CHARSET のキャラクタ・セットは参照先ファイル内のワイルドカードを使用したオブジェクト名を読み取りますが、参照先ファイルにおけるその他すべてのマルチバイト指定については、エスケープ・シーケンス (\uXXXX) を使用する必要があります。CHARSET 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。

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、232 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 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。

オブジェクト名でサポートされる文字

Oracle GoldenGate は、オブジェクト名および列名でほとんどの文字をサポートしています。次に示すサポートされる文字とサポートされない文字のリストは、Oracle GoldenGate でサポートされているすべてのデータベースを対象としていますが、リストのすべての文字がサポートされるかどうかはデータベース・プラットフォームによって異なります。

サポートされる特殊文字

Oracle GoldenGate は、次の特殊文字も含めて、データベースでサポートされるすべての文字をサポートしています。パラメータ・ファイルでは、これらの特殊文字を含むオブジェクト名を二重引用符で囲む必要があります。

文字	説明
*	アスタリスク (パラメータ・ファイルで使用する場合は、* のように、バック・スラッシュでエスケープする必要があります。)
?	疑問符 (パラメータ・ファイルで使用する場合は、\? のように、バック・スラッシュでエスケープする必要があります。)
/	フォワード・スラッシュ
\	バック・スラッシュ (パラメータ・ファイルで使用する場合は、\\ とする必要があります。)
@	アットマーク (サポートされますが、データベースでリソース・ロケータとして使用されることが多い文字です。オブジェクト名では問題を生じることがあります。)

文字	説明
#	番号記号
$	ドル記号
%	パーセント記号 (パラメータ・ファイルで使用する場合は、%% とする必要があります。)
^	カレット記号
( )	開き丸カッコと閉じ丸カッコ
_	アンダースコア
-	ダッシュ
	空白

サポートされない特殊文字

次の文字は、オブジェクト名およびキー以外の列名ではサポートされません。

文字	説明
{}	開き中カッコと閉じ中カッコ
[ ]	開き大カッコと閉じ大カッコ
=	等号
+	プラス記号
!	感嘆符
~	チルド
|	パイプ
&	アンパサンド
:	コロン
;	セミコロン
,	カンマ
' '	一重引用符
" "	二重引用符
'	アクセント記号 (発音区別符)

文字	説明
.	ピリオド
<	小なり記号
>	大なり記号

Oracle GoldenGate の入力におけるオブジェクト名の指定

パラメータ・ファイル (TABLE 文、MAP 文など)、列変換関数、コマンド、その他の入力でデータベース・オブジェクト名を指定する際には、次のルールに従ってください。

- オブジェクト名は、任意の長さで、サポートされる任意のキャラクタ・セットを使用して指定できます。サポートされる文字については、36 ページの「オブジェクト名でサポートされる文字」を参照してください。サポートされるキャラクタ・セットについては、286 ページの「サポートされるキャラクタ・セット」を参照してください。
- オブジェクト名がパラメータ・ファイルで完全修飾されていない場合 (表名として EMP のみが指定されている場合など)、Oracle GoldenGate ではログイン・セッションのデフォルト・スキーマ名が使用されます。たとえば、FIN が現在のログイン・セッションのデフォルト・スキーマである場合、FIN.EMP が使用されます。
- 空白や記号などの特殊文字が含まれているオブジェクト名は、二重引用符で囲んで指定します。
- 大 / 小文字が区別されるデータベースのオブジェクト名は、それをホスト・データベースに格納するときと同じ大 / 小文字で指定します。データベースのタイプによっては、大 / 小文字の区別の有無がデータベースのレベルごとに異なる場合がありますので注意してください (スキーマは大 / 小文字が区別されるが、表は区別されないなど)。大 / 小文字を区別するには引用符が必要になるデータベースの場合は、大 / 小文字を区別する各オブジェクトを修飾名の中で引用符で囲んでください。

  正 : `TABLE "Sales"."ACCOUNT"`

  誤 : `TABLE "Sales.ACCOUNT"`

- Oracle GoldenGate では、マッピング目的での必要に応じて、大 / 小文字が区別されない名前を格納時の大 / 小文字に変換します。

次の表は、オブジェクト名における大 / 小文字の区別のサポート状況を、サポートされるデータベースごとに簡単にまとめたものです。この種のサポートの詳細は、データベースのドキュメントを参照してください。

注意　サポートされるすべてのデータベースにおいて、パスワードは、関連オブジェクト名が引用符で囲まれているかどうかに関係なく、常に大 / 小文字を区別するものとして扱われます。

表 2　データベースごとのオブジェクト名の大 / 小文字の区別

データベース	大 / 小文字を区別するために引用符が必要か	引用符で囲んでいないオブジェクト名	引用符で囲んだオブジェクト名
DB2	必要。大 / 小文字を区別するかどうかを引用符の有無で識別します。	大 / 小文字が区別されず、大文字で格納されます。	大 / 小文字が区別され、大文字と小文字の混在で格納されます。

表 2　データベースごとのオブジェクト名の大 / 小文字の区別（続き）

データベース	大 / 小文字を区別するために引用符が必要か	引用符で囲んでいないオブジェクト名	引用符で囲んだオブジェクト名
Oracle	必要。大 / 小文字を区別するかどうかを引用符の有無で識別します。	大 / 小文字が区別されず、大文字で格納されます。	大 / 小文字が区別され、大文字と小文字の混在で格納されます。
SQL/MX	必要。大 / 小文字を区別するかどうかを引用符の有無で識別します。	大 / 小文字が区別されず、大文字で格納されます。	大 / 小文字が区別され、大文字と小文字の混在で格納されます。
PostgreSQL	必要。大 / 小文字を区別するかどうかを引用符の有無で識別します。	大 / 小文字が区別されず、小文字で格納されます。	大 / 小文字が区別され、大文字と小文字の混在で格納されます。
MySQL (大 / 小文字が区別されるデータベース)	不要 ◆ 常に大 / 小文字が区別され、大文字と小文字の混在で格納されます。 ◆ 列、トリガーおよびプロシージャの名前は大 / 小文字が区別されません。	影響なし	影響なし
MySQL (大 / 小文字が区別されないデータベース)	不要 ◆ 常に大 / 小文字が区別されず、大文字と小文字の混在で格納されます。 ◆ 列、トリガーおよびプロシージャの名前は大 / 小文字が区別されません。	影響なし	影響なし
◆ **SQL Server** ◆ **Sybase** (データベースは大 / 小文字を区別するものとして作成)	不要 常に大 / 小文字が区別され、大文字と小文字の混在で格納されます。	影響なし	影響なし
◆ **SQL Server** ◆ **Sybase** (データベースは大 / 小文字を区別しないものとして作成)	不要 常に大 / 小文字が区別されず、大文字と小文字の混在で格納されます。	影響なし	影響なし

表 2　データベースごとのオブジェクト名の大 / 小文字の区別 (続き)

データベース	大 / 小文字を区別するために引用符が必要か	引用符で囲んでいないオブジェクト名	引用符で囲んだオブジェクト名
Teradata	不要 常に大 / 小文字が区別されず、大文字と小文字の混在で格納されます。	影響なし	影響なし
TimesTen	不要 常に大 / 小文字が区別されず、大文字で格納されます。	大 / 小文字が区別されず、大文字で格納されます。	大 / 小文字が区別されず、大文字で格納されます。

オブジェクト名でのワイルドカードの使用

ワイルドカードは、表および順序の名前に使用できます。所有者名 (スキーマなど) にはワイルドカードを使用しないでください。Oracle GoldenGate のパラメータでは、2 種類のワイルドカードを適宜使用して、1 つの文に複数のオブジェクトを指定できます。

- 疑問符 (?) は 1 文字に置き換わります。たとえば、TAB*n*(*n* は 0 ～ 9) という名前の表を含むスキーマでは、HQ.TAB? というワイルドカード指定を行うと、HQ.TAB0、HQ.TAB1、HQ.TAB2 ～ HQ.TAB9 が返されます (これ以外は返されません)。このワイルドカードは、DEFGEN ではサポートされません。
- アスタリスク (*) は、任意の数の文字 (ゼロ・シーケンスを含む) を表します。たとえば、HQ.T* と指定すると、HQ.TOTAL、HQ.T123、HQ.T などのオブジェクトが返されます。
- TABLE 文および MAP 文では、ソース・オブジェクト名の中でアスタリスクと疑問符のワイルドカード文字を組み合せることができます。

ソース・オブジェクトにワイルドカードを使用する場合のルール

ソース・オブジェクトには、アスタリスクを単独で、または部分的な名前とともに使用できます。たとえば、次のソース指定はいずれも有効です。

- `TABLE HQ.*;`
- `MAP HQ.T_*;`
- `MAP HQ.T_*, TARGET HQ.*;`

TABLE、MAP、SEQUENCE の各パラメータでは、ワイルドカードを解決するために、大 / 小文字の区別の有無とデータベースのロケールが考慮されます。大 / 小文字を区別する、または区別しないものとして作成されたデータベースの場合、名前と大 / 小文字がワイルドカードと正確に照合されます。たとえば、データベースで大 / 小文字が区別される場合、SCHEMA.TABLE は SCHEMA.TABLE に一致し、Schema.Table は Schema.Table に一致します。データベースで大 / 小文字が区別されない場合、照合でも大 / 小文字は区別されません。

引用符を使用して大 / 小文字を区別することで、大 / 小文字が区別されるオブジェクト名と区別されないオブジェクト名を同じデータベース・インスタンス内に持つことができるデータベースの場合、ワイルドカードは異なる働きをします。アスタリスク・ワイルドカードを TABLE 文でソース名として単独で使用した場合、そのアスタリスクは、引用符で囲まれているかどうかに関係なく、任意の文字に一致します。次の文はいずれも同じ結果になります。

```
TABLE *;
TABLE "*";
```

同様に、単独で使用した疑問符ワイルドカードは、引用符で囲まれているかどうかに関係なく、任意の 1 文字に一致します。次の指定はいずれも同じ結果になります。

```
TABLE ?;
TABLE "?";
```

疑問符またはアスタリスクのワイルドカードを他の文字とともに使用する場合は、大 / 小文字の区別の有無がワイルドカード以外の文字にも適用されますが、ワイルドカードは大 / 小文字が区別される名前と区別されない名前の両方に一致します。

- 次の TABLE 文では、小文字の abc で始まる表名がすべて取得されます。引用符で囲んだ名前の大 / 小文字は維持され、大 / 小文字を区別する照合が適用されます。ワイルドカードは大 / 小文字が区別される文字と区別されない文字の両方に一致するため、abcA と abca を含む表名が取得されます。

  ```
  TABLE "abc*";
  TABLE "abc?";
  ```

- 次の TABLE 文では、部分的な名前は大 / 小文字が区別されず (引用符なし)、このデータベースに大文字で格納されるため、大文字の ABC で始まる表名がすべて取得されます。ただし、ワイルドカードは大 / 小文字が区別される文字と区別されない文字の両方に一致するため、この例では ABCA と ABCa を含む表名が取得されます。

  ```
  TABLE abc*;
  TABLE abc?;
  ```

ターゲット・オブジェクトにワイルドカードを使用する場合のルール

MAP 文の TARGET 句でワイルドカードを使用する場合、ターゲット・オブジェクトがターゲット・データベースに存在する必要があります (ただし、DDL レプリケーションがサポートされ、使用されている場合を除きます。その場合、新しいスキーマとそのオブジェクトを作成時にレプリケートできます。)

ターゲット・オブジェクトには、アスタリスクのみを使用できます。アスタリスク・ワイルドカードを部分的な名前とともに使用すると、Replicat はワイルドカードを対応するソース・オブジェクトの完全な名前に置き換えます。したがって、次のような指定は*誤り*です。

```
TABLE HQ.T_*, TARGET RPT.T_*;
MAP HQ.T_*, TARGET RPT.T_*;
```

前述の指定では、ターゲット指定内のワイルドカードが T_TEST(ソース・オブジェクトの名前) に置き換わるため、ターゲット名全体では T_T_TEST<xxx> となります。これにより、次のような結果が生じます。

- HQ.T_TEST1 が RPT.T_T_TEST1 にマップされます。
- HQ.T_TEST2 が RPT.T_T_TEST2 にマップされます。

(以降同様)

次の例は、アスタリスク・ワイルドカードの正しい使用方法を示しています。

```
MAP HQ.T_*, TARGET RPT.*;
```

これにより、次のような正しい結果が得られます。

- HQ.T_TEST1 が RPT.T_TEST1 にマップされます。
- HQ.T_TEST2 が RPT.T_TEST2 にマップされます。

(以降同様)

名前のフォールバック・マッピング

Oracle GoldenGate には、ソース名をターゲット名にマップできない場合に備えて、フォールバック・マッピングのメカニズムが用意されています。大 / 小文字が区別されるソース・オブジェクトと完全に一致するものがターゲット上に見つからない場合、Replicat はソース名を、ターゲット上の大文字または小文字 (データベースのタイプによる) の同じ名前にマップしようとします。詳細は、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』の NAMEMATCHIGNORECASE パラメータの説明を参照してください。

11.2.1 より前のバージョンの証跡からのワイルドカード・マッピング

Oracle GoldenGate 11.2.1 より前のバージョンの証跡ファイルから読み取るように Replicat が構成されている場合、ターゲットのマッピングは、下位互換性を確保するために次の方法で行われます。

- 引用符で囲んだオブジェクト名は大 / 小文字が区別されます。
- 引用符で囲んでいないオブジェクト名は大 / 小文字が区別されません。

次の例では、大 / 小文字が区別される表名 abc が、ターゲット abc にマップされます。この処理が発生するのは、11.2.1 より前の Extract により、大 / 小文字が区別される構成の SQL Server と Sybase について証跡が書き込まれた場合のみです。この例では、ターゲット・データベースが Oracle、DB2 または SQL/MX で、ターゲット・データベース内に大 / 小文字が区別される abc はないが、表 ABC はあるという場合、名前のフォールバック・マッピングが実行されます。(「名前のフォールバック・マッピング」を参照してください。)

```
MAP "abc", TARGET *;
```

次の例では、大 / 小文字が区別されない表名 abc が、ターゲットの表名 ABC にマップされます。旧リリースの Oracle GoldenGate では、大 / 小文字が区別されないオブジェクト名は大文字で証跡に格納されていたため、ターゲットの表名は常に大文字でした。大 / 小文字が区別されない名前変換の場合、比較はロケールを考慮せずに、大文字 (A から Z の文字のみ) で US-ASCII で行われます。

```
MAP abc, TARGET *;
```

オブジェクト名におけるリテラルとしてのアスタリスクまたは疑問符

オブジェクトの名前自体にアスタリスクまたは疑問符が含まれている場合、次の例に示すように、名前全体をエスケープして二重引用符で囲む必要があります。

```
TABLE HT."\?ABC";
```

ワイルドカードの解決方法

オブジェクト名がワイルドカードで指定されている場合、デフォルトでは、ソース・オブジェクトの 1 行目の処理と同時に、そのオブジェクトの解決が行われます。(これに対し、オブジェクト名が明示的に指定されている場合は、プロセスの起動時に解決が行われます。) ワイルドカード解決のルールを変更するには、WILDCARDRESOLVE パラメータを使用します。デフォルトは DYNAMIC です。

ワイルドカード指定からのオブジェクトの除外

TABLEEXCLUDE パラメータと MAPEXCLUDE パラメータを使用すると、ワイルドカードによるオブジェクト選択を、明示的なオブジェクト除外と組み合せて使用できます。

パラメータ・ファイルにおける名前およびリテラルの SQL-92 ルールの適用

Oracle GoldenGate のパラメータ・ファイル、変換関数、ユーザー・イグジットおよびコマンドでは、二重引用符で囲んだ文字はデフォルトで文字列リテラルとして扱われます。デフォルトでは、引用符で囲んだ列名の大 / 小文字の区別は認識されません (Oracle など、大 / 小文字を区別するには引用符が必要になるデータベースの場合)。たとえば、次の例は正しく指定されています (PRODUCT_CODE は列名、その他の文字列はリテラルです)。

```
@CASE (PRODUCT_CODE, "CAR", "A car", "TRUCK", "A truck")
```

二重引用符で囲んだ、大 / 小文字が区別されるオブジェクト名をサポートするには、GLOBALS パラメータ・ファイルで USEANSISQLQUOTES パラメータを使用します。このパラメータにより、SQL-92 ルールが適用されます。USEANSISQLQUOTES を使用する場合、列名は二重引用符で囲んで指定し、リテラル文字列は一重引用符で囲んで指定する必要があります。次の例では、Product_Code は Oracle データベース内の大 / 小文字が区別される列名であり、その他の文字列はリテラルです。

```
@CASE ("Product_Code", 'CAR', 'A car', 'TRUCK', 'A truck')
```

USEANSISQLQUOTES 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。

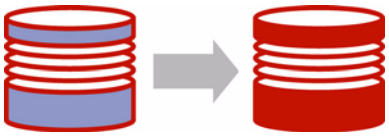
注意 TABLE、SEQUENCE、MAP の各文は、引用符で囲んだ表名および順序名をデフォルトでサポートしているため、USEANSISQLQUOTES を使用する必要はありません。

第 5 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

- 46 ページの「標準レポート構成の作成」
- 48 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」
- 51 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」
- 56 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。64 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。

- 141 ページの「データのマッピングおよび操作」
- 225 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、93 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 の構成」を参照してください。

追加情報

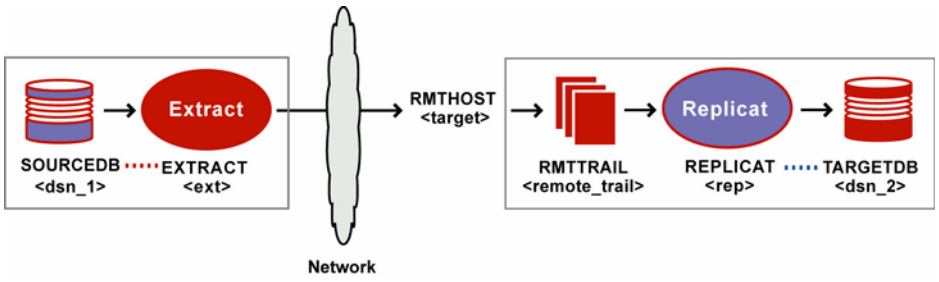
- システムおよびデータベースの構成の追加要件は、使用中のデータベースのタイプに対応する Oracle GoldenGate の*インストレーションおよびセットアップ・ガイド*を参照してください。
- Teradata の Extract 構成の追加要件は、『Oracle GoldenGate *Teradata インストレーションおよびセットアップ・ガイド*』を参照してください。
- Oracle GoldenGate の変更取得および配信グループの構成の詳細は、183 ページの「オンライン変更同期の構成」を参照してください。
- Oracle GoldenGate のコマンドとパラメータの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 3 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 | INTEGRATED TRANLOG}, BEGIN <time>
[, THREADS <n>]
```

   - TRANLOG および INTEGRATED TRANLOG については、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』を参照してください。INTEGRATED TRANLOG により、Oracle データベースに対する統合取得が可能になりました。

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>
    [<encryption options>]]
-- Specify the name or IP address of the target system and
-- optional encryption across TCP/IP:
RMTHOST <target>, MGRPORT <portnumber>, ENCRYPT <encryption options>
-- Specify the remote trail and encryption options on the target system:
ENCRYPTTRAIL [<encryption options>]
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 3 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、184 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>][, PASSWORD <pw>
    [<encryption options>]]
-- Specify error handling rules:
-- Specify decryption if trail is encrypted:
DECRYPTTRAIL <encryption options>
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

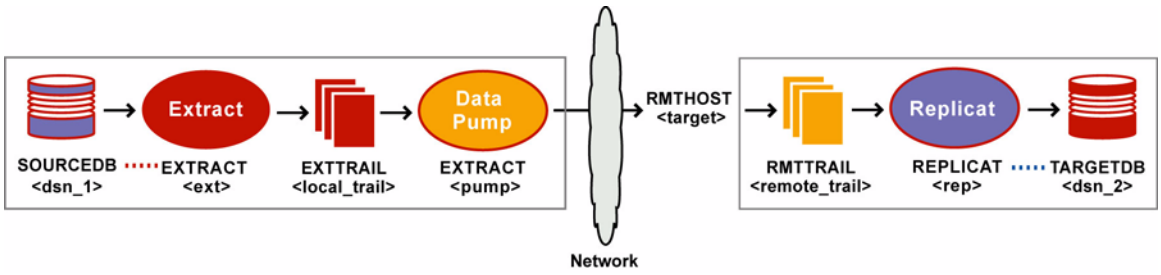
ソース・システムにデータ・ポンプを追加して、プライマリ Extract を TCP/IP 機能から分離することで、記憶域の柔軟性を向上し、プライマリ Extract からフィルタリングおよび変換処理のオーバーヘッドを削減できます。

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



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 3 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 | INTEGRATED TRANLOG}, BEGIN <time>
[, THREADS <n>]
```

   - TRANLOG および INTEGRATED TRANLOG については、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』を参照してください。INTEGRATED TRANLOG により、Oracle データベースに対する統合取得が可能になりました。

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>
    [<encryption options>]]
-- Specify the local trail that this Extract writes to and
-- optional encryption:
ENCRYPTTRAIL <encryption options>
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 if using NOPASSTHRU:
[SOURCEDB <dsn_1>][,USERID <user>][, PASSWORD <pw>
    [<encryption options>]]
-- Specify decryption if input trail is encrypted:
DECRYPTTRAIL <encryption options>
-- Specify the name or IP address of the target system
-- and optional encryption of data over TCP/IP:
RMTHOST <target>, MGRPORT <portnumber>, ENCRYPT <encryption options>
-- Specify the remote trail and encryption options on the target system:
ENCRYPTTRAIL [<encryption options>]
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 3 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、184 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。

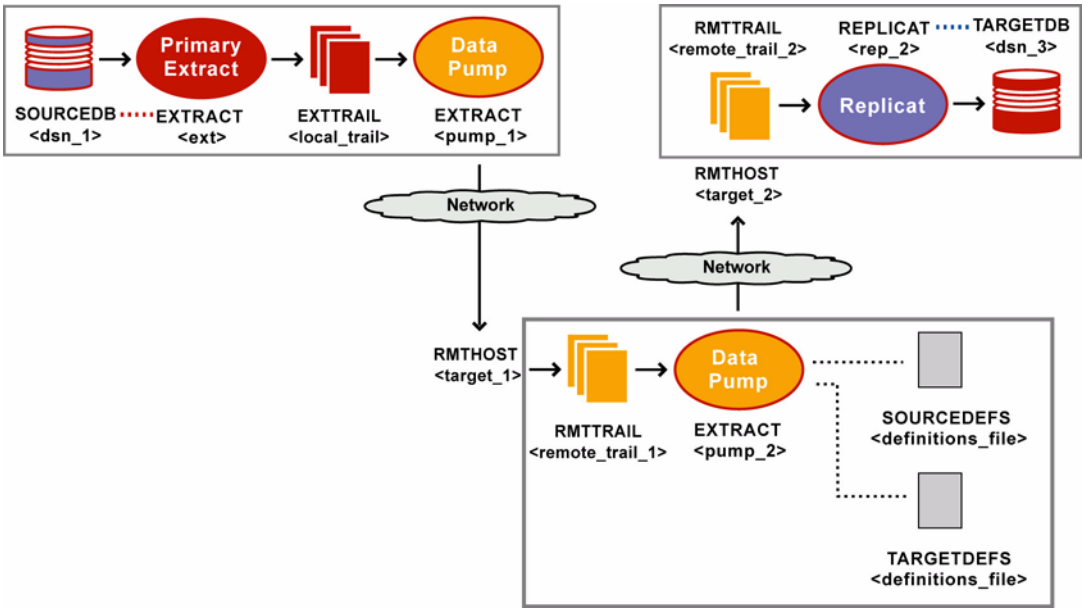
13. ターゲットで、EDIT PARAMS コマンドを使用して Replicat グループのパラメータ・ファイルを作成します。次のパラメータと、データベース環境に適用する他のパラメータを含めます。

```
-- 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>][, PASSWORD <pw>
    [<encryption options>]]
-- Specify decryption if input trail is encrypted:
DECRYPTTRAIL <encryption options>
-- 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

中間システムを、ソース・システムとターゲット・システムの中継場所として使用できます。この構成では、ソース・システムのデータ・ポンプは、取得した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送信します。ターゲットの Replicat はリモート証跡を読み取り、データをターゲット・データベースに適用します。

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



このトポロジについて検討する際、次の点に注意してください。

- 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、各システム間が直接接続されていない場合に役立ちます。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。
- この構成を使用すると、記憶域の柔軟性が増し、ソースまたはターゲットを補うことができます。
- この構成は、すべてのシステムのキャラクタ・セットが同じ場合、データのフィルタリングや変換に使用できます。キャラクタ・セットが異なる場合、データ・ポンプはキャラクタ・セット間の変換を行えません。ターゲットで変換とトランスフォーメーションを行うよう Replicat を構成する必要があります。
- キャラクタ・セットが同じ場合、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。定義ファイルと変換の詳細は、第 13 章を参照してください。
- この構成は、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。

ソース・システム

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。

Manager プロセスを構成する手順

1. ソースで、第 3 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 | INTEGRATED TRANLOG}, BEGIN <time>
   [, THREADS <n>]
   ```

   - TRANLOG および INTEGRATED TRANLOG については、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』を参照してください。INTEGRATED TRANLOG により、Oracle データベースに対する統合取得が可能になりました。

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>
    [<encryption options>]]
-- Specify the local trail that this Extract writes to and
-- encryption options:
ENCRYPTTRAIL <encryption options>
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:
[SOURCEDB <dsn_1>][,USERID <user>][, PASSWORD <pw>
    [<encryption options>]]
-- Specify decryption if input trail is encrypted:
DECRYPTTRAIL <encryption options>
-- Specify the name or IP address of the intermediary system
-- and optional encryption of data over TCP/IP:
RMTHOST <target_1>, MGRPORT <portnumber>, ENCRYPT <encryption options>
-- Specify remote trail and encryption options on intermediary system:
ENCRYPTTRAIL [<encryption options>]
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 3 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>
-- Note that no database login parameters are required in this case.
-- 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 decryption if input trail is encrypted:
DECRYPTTRAIL <encryption options>
-- Specify the name or IP address of the target system
-- and optional encryption of data over TCP/IP:
RMTHOST <target_2>, MGRPORT <portnumber>, ENCRYPT <encryption options>
-- Specify the remote trail and encryption options on the target system:
ENCRYPTTRAIL [<encryption options>]
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 3 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、184 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>][, PASSWORD <pw>
    [<encryption options>]]
-- Specify decryption if input trail is encrypted:
DECRYPTTRAIL <encryption options>
-- Specify error handling rules:
REPERROR (<error>, <response>)
-- Specify tables for delivery:
MAP <owner>.<table>, TARGET <owner>.<table>[, DEF <template name>];
```

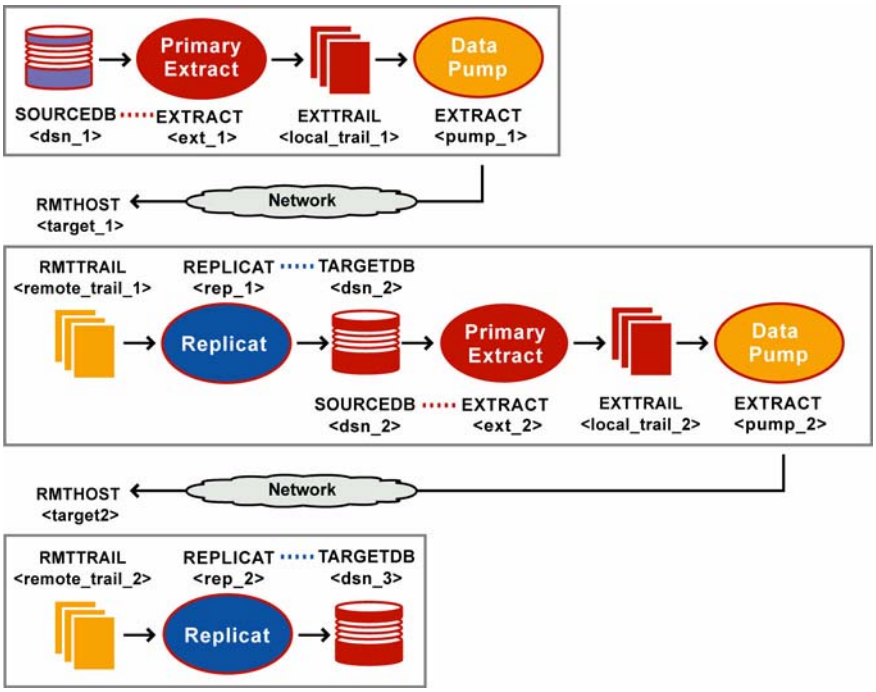
カスケード・レポート構成の作成

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。
- 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、51 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



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、2 番目の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)。

このトポロジについて検討する際、次の点に注意してください。

- この構成は、すべてのシステムのキャラクタ・セットが同じ場合、データのフィルタリングや変換に使用できます。キャラクタ・セットが異なる場合、データ・ポンプはキャラクタ・セット間の変換を行えません。ターゲットで変換とトランスフォーメーションを行うよう Replicat を構成する必要があります。
- キャラクタ・セットが同じ場合、2 番目のシステムでデータ・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最初のシステムでソース定義ファイルを作成し、2 番目のシステムに転送する必要があります。さらに、2 番目のシステムでソース定義ファイルを作成し、3 番目のシステムに転送する必要があります。定義ファイルと変換の詳細は、第 13 章を参照してください。
- 2 番目のシステムで、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 グループを構成する必要があります。この動作を制御する Extract パラメータは、IGNOREAPPLOPS および GETREPLICATES です。

ソース・システム

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。

ソースの Manager プロセスを構成する手順

1. ソースで、第 3 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 | INTEGRATED TRANLOG}, BEGIN <time>
   [, THREADS <n>]
   ```

   - TRANLOG および INTEGRATED TRANLOG については、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』を参照してください。INTEGRATED TRANLOG により、Oracle データベースに対する統合取得が可能になりました。

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>
       [<encryption options>]]
   -- Specify the local trail that this Extract writes to
   -- and specify encryption options:
   ENCRYPTTRAIL <encryption options>
   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 if using NOPASSTHROUGH:
[SOURCEDB <dsn_1>][,USERID <user>][, PASSWORD <pw>
    [<encryption options>]]
-- Specify decryption if input trail is encrypted:
DECRYPTTRAIL <encryption options>
-- Specify the name or IP address of second system in cascade
-- and optional encryption of data over TCP/IP:
RMTHOST <target_1>, MGRPORT <portnumber>, ENCRYPT <encryption options>
-- Specify the remote trail and encryption options on the second system:
ENCRYPTTRAIL <encryption options>
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 3 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、184 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>][, PASSWORD <pw>
    [<encryption options>]]
-- Specify decryption if input trail is encrypted:
DECRYPTTRAIL <encryption options>
-- 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 | INTEGRATED TRANLOG}, BEGIN <time>
[, THREADS <n>]
```

- TRANLOG および INTEGRATED TRANLOG については、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』を参照してください。INTEGRATED TRANLOG により、Oracle データベースに対する統合取得が可能になりました。

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>
    [<encryption options>]]
-- Specify the local trail that this Extract writes to
-- and encryption options:
ENCRYPTTRAIL <encryption options>
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 if using NOPASSTHRU:
[SOURCEDB <dsn_2>][, USERID <user>][, PASSWORD <pw>
    [<encryption options>]]
-- Specify decryption if input trail is encrypted:
DECRYPTTRAIL <encryption options>
-- Specify the name or IP address of third system in cascade
-- and optional encryption of data over TCP/IP:
RMTHOST <target_2>, MGRPORT <portnumber>, ENCRYPT <encryption options>
-- Specify the remote trail and encryption options on the third system:
ENCRYPTTRAIL <encryption options>
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 3 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、184 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。

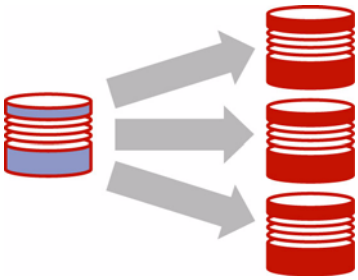
```
-- Identify the Replicat group:
REPLICAT <rep_2>
-- State whether or not source and target definitions are identical:
SOURCEDEFS <full_pathname> | ASSUMETARGETDEFS
-- Specify database login information as needed for the database:
[TARGETDB <dsn_3>][, USERID <user>][, PASSWORD <pw>
    [<encryption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

第 6 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。フィルタリングと変換のオプションについては、141 ページの「データのマッピングおよび操作」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。

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、93 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 の構成」を参照してください。

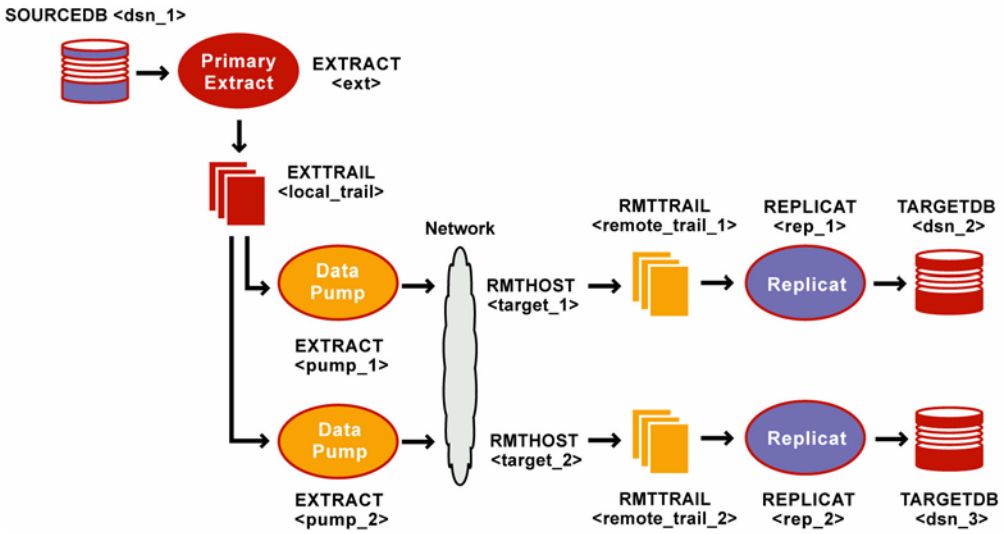
追加情報

- 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、183 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 3 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 | INTEGRATED TRANLOG}, BEGIN <time>,
[, THREADS]
```

   - TRANLOG および INTEGRATED TRANLOG については、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』を参照してください。INTEGRATED TRANLOG により、Oracle データベースに対する統合取得が可能になりました。

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>
    [<encryption options>]]
-- Specify the local trail that this Extract writes to
-- and encryption options:
ENCRYPTTRAIL <encryption options>
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 if using NOPASSTHRU:
[SOURCEDB <dsn_1>][,USERID <user>][, PASSWORD <pw>
    [<encryption options>]]
-- Specify decryption options if input trail is encrypted.
DECRYPTTRAIL <encryption options>
-- Specify the name or IP address of the first target system
-- and optional encryption of data over TCP/IP:
RMTHOST <target_1>, MGRPORT <portnumber>, ENCRYPT <encryption options>
-- Specify remote trail and encryption options on first target system:
ENCRYPTTRAIL <encryption options>
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 if using NOPASSTHRU:
[SOURCEDB <dsn_1>][,USERID <user>][, PASSWORD <pw>
    [<encryption options>]]
-- Specify decryption options if input trail is encrypted.
DECRYPTTRAIL <encryption options>
-- Specify the name or IP address of the second target system
-- and optional encryption of data over TCP/IP:
RMTHOST <target_2>, MGRPORT <portnumber>, ENCRYPT <encryption options>
-- Specify remote trail and encryption options on second target system:
ENCRYPTTRAIL <encryption options>
RMTTRAIL <remote_trail_2>
-- Allow mapping, filtering, conversion or pass data through as-is:
[PASSTHRU | NOPASSTHRU]
-- 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

9. 各ターゲットで、第 3 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、184 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>][, PASSWORD <pw>
    [<encryption options>]]
-- Specify decryption options if input trail is encrypted.
DECRYPTTRAIL <encryption options>
-- 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>][, PASSWORD <pw>
    [<encryption options>]]
-- Specify decryption options if input trail is encrypted.
DECRYPTTRAIL <encryption options>
-- Specify error handling rules:
REPERROR (<error>, <response>)
-- Specify tables for delivery:
MAP <owner>.<table>, TARGET <owner>.<table>[, DEF <template name>];
```

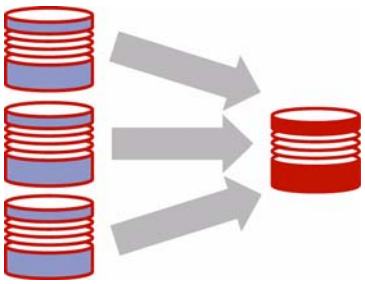
- どの Replicat グループに対しても任意の数の MAP 文を使用できます。特定の Replicat グループに対するすべての MAP 文では、そのグループにリンクされた証跡に含まれる同じオブジェクトを指定する必要があります。

第 7 章

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、93 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 の構成」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。フィルタリングと変換のオプションについては、141 ページの「データのマッピングおよび操作」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* を参照してください。

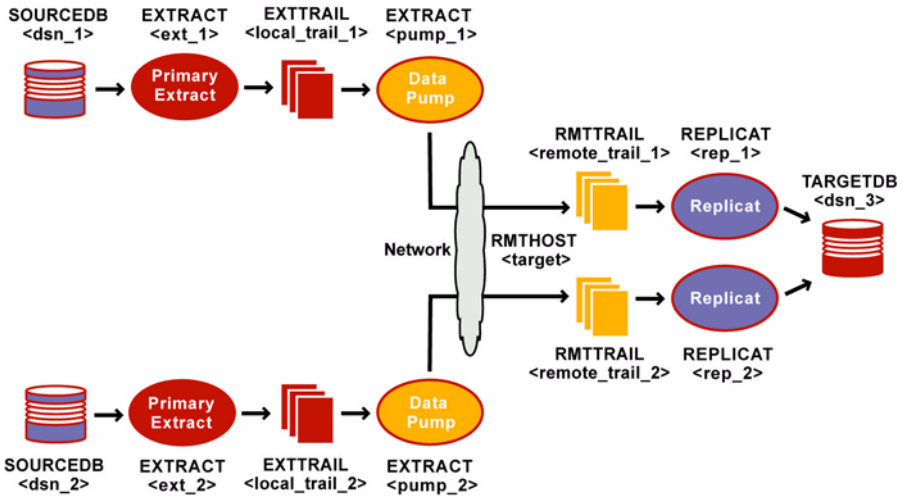
追加情報

- 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、183 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 3 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 | INTEGRATED TRANLOG}, BEGIN <time>
   [, THREADS <n>]
   ```

   Extract_2

   ```
   ADD EXTRACT <ext_2>, (TRANLOG | INTEGRATED TRANLOG}, BEGIN <time>
   [, THREADS <n>]
   ```

   - TRANLOG および INTEGRATED TRANLOG については、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』を参照してください。INTEGRATED TRANLOG により、Oracle データベースに対する統合取得が可能になりました。

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>
    [<encryption options>]]
-- Specify the local trail that this Extract writes to
-- and encryption options:
ENCRYPTTRAIL <encryption options>
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>
    [<encryption options>]]
-- Specify the local trail that this Extract writes to
-- and encryption options:
ENCRYPTTRAIL <encryption options>
-- 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>
       [<encryption options>]]
   -- Specify decryption if the input trail is encrypted.
   DECRYPTTRAIL <encryption options>
   -- Specify the name or IP address of the target system
   -- and optional encryption of data over TCP/IP:
   RMTHOST <target>, MGRPORT <portnumber>, ENCRYPT <encryption options>
   -- Specify the remote trail and encryption options on the target system:
   ENCRYPTTRAIL <encryption options>
   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>
    [<encryption options>]]
-- Specify decryption if the input trail is encrypted.
DECRYPTTRAIL <encryption options>
-- Specify the name or IP address of the target system
-- and optional encryption of data over TCP/IP:
RMTHOST <target>, MGRPORT <portnumber>, ENCRYPT <encryption options>
-- Specify the remote trail and encryption options on the target system:
ENCRYPTTRAIL <encryption options>
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。

ターゲット・システム

Manager プロセスを構成する手順

9. 第 3 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>][, PASSWORD <pw>
    [<encryption options>]]
-- Specify decryption if the input trail is encrypted.
DECRYPTTRAIL <encryption options>
-- 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>][, PASSWORD <pw>
    [<encryption options>]]
-- Specify decryption if the input trail is encrypted.
DECRYPTTRAIL <encryption options>
-- Specify error handling rules:
REPERROR (<error>, <response>)
-- Specify tables for delivery:
MAP <owner>.<table>, TARGET <owner>.<table>[, DEF <template name>];
```

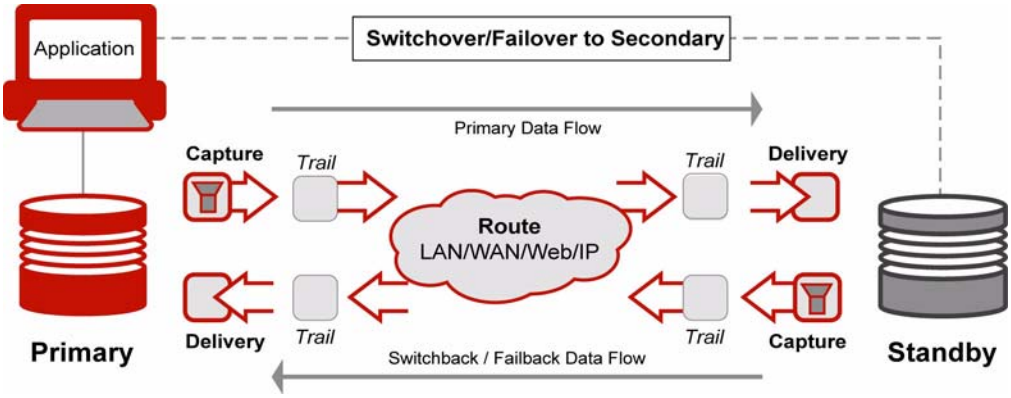
- どの Replicat グループに対しても任意の数の MAP 文を使用できます。特定の Replicat グループに対するすべての MAP 文では、そのグループにリンクされた証跡に含まれる同じオブジェクトを指定する必要があります。

第 8 章

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

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。93 ページの第 9 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。

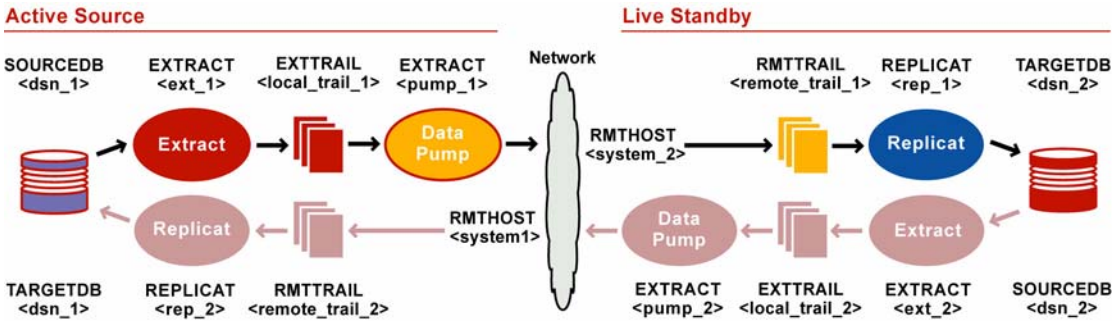
追加情報

- 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、183 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、184 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 3 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。
2.

```
ADD EXTRACT <ext_1>, {TRANLOG | INTEGRATED TRANLOG}, BEGIN <time>
[, THREADS <n>]
```

   - TRANLOG および INTEGRATED TRANLOG については、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』を参照してください。INTEGRATED TRANLOG により、Oracle データベースに対する統合取得が可能になりました。

3. 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。

4. 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>
    [<encryption options>]]
-- Specify the local trail that this Extract writes to
-- and encryption options:
ENCRYPTTRAIL <encryption options>
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>
    [<encryption options>]]
-- Specify decryption if the input trail is encrypted:
DECRYPTTRAIL <encryption options>
-- Specify the name or IP address of the standby system
-- and optional encryption of data over TCP/IP:
RMTHOST <system_2>, MGRPORT <portnumber>, ENCRYPT <encryption options>
-- Specify the remote trail and encryption options on the standby system:
ENCRYPTTRAIL <encryption options>
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 機能、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、184 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>][, PASSWORD <pw>
    [<encryption options>]]
-- Specify decryption if the input trail is encrypted:
DECRYPTTRAIL <encryption options>
-- 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 | INTEGRATED TRANLOG}, BEGIN <time>
[, THREADS <n>]
```

   - TRANLOG および INTEGRATED TRANLOG については、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』を参照してください。INTEGRATED TRANLOG により、Oracle データベースに対する統合取得が可能になりました。

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>
    [<encryption options>]]
-- Specify local trail and encryption options that this Extract writes to:
ENCRYPTTRAIL <encryption options>
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>
    [<encryption options>]]
-- Specify decryption if the input trail is encrypted:
DECRYPTTRAIL <encryption options>
-- Specify the name or IP address of the active source system
-- and optional encryption of data over TCP/IP:
RMTHOST <system_1>, MGRPORT <portnumber>, ENCRYPT <encryption options>
-- Specify remote trail and encryption options on active source system:
ENCRYPTTRAIL <encryption options>
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>][, PASSWORD <pw>
    [<encryption options>]]
-- Specify decryption if the input trail is encrypted:
DECRYPTTRAIL <encryption options>
-- 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た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 | INTEGRATED TRANLOG}, BEGIN <time>
   [, THREADS <n>]
   ```

   - TRANLOG および INTEGRATED TRANLOG については、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』を参照してください。INTEGRATED TRANLOG により、Oracle データベースに対する統合取得が可能になりました。

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>
   ```

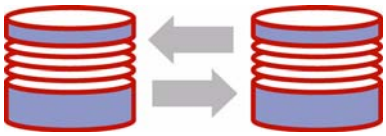
5. **プライマリ**・システムで、アプリケーションへのユーザー・アクセスを許可します。

第 9 章

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

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。

- 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。

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。

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。

比較のために WHERE 句で使用できる列に関連したその他のオプションについては、104 ページの「競合の管理」の内容を参照してください。

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すべて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。Oracle GoldenGate には、挿入、更新および削除の競合に対する競合解決サポートが組み込まれています。104 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、183 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)

Replicat によって他のデータベース・タイプに適用される SQL の取得を防止するには、次のパラメータを使用します。

- 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

Replicat トランザクションを識別するように Extract を構成するには、Extract がデータを取得するデータベースについて、次の手順に従います。

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。

Oracle

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>
  ```

  注意 TRANLOGOPTIONS を使用するかわりに、GGSCI で ADD TRACETABLE コマンドを使用してトレース表を作成できます。ただし、この方法を取ると、TRANLOGOPTIONS オプションを使用する場合より処理のオーバーヘッドが大きくなります。詳細は、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』の ADD TRACETABLE の説明を参照してください。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。

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

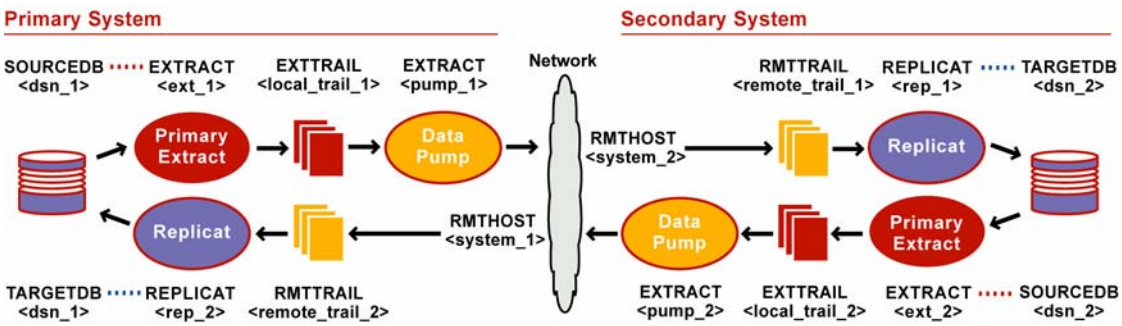
Teradata データベースに適用される Replicat トランザクションを識別する必要はありません。

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、262 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、184 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 3 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 | INTEGRATED TRANLOG}, BEGIN <time>
   [, THREADS <n>]
   ```

   - TRANLOG および INTEGRATED TRANLOG については、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』を参照してください。INTEGRATED TRANLOG により、Oracle データベースに対する統合取得が可能になりました。

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>
    [<encryption options>]]
-- Specify the local trail that this Extract writes to
-- and encryption options:
ENCRYPTTRAIL <encryption options>
EXTTRAIL <local_trail_1>
-- Exclude Replicat transactions. Uncomment ONE of the following:
-- DB2 z/OS and LUW, Oracle, and Sybase:
-- TRANLOGOPTIONS EXCLUDEUSER <Replicat_user>
-- Oracle alternative to EXCLUDEUSER:
-- EXCLUDEUSERID <Oracle uid>
-- Oracle alternative to the TRANLOGOPTIONS options:
-- TRACETABLE <trace_table_name>
-- SQL Server and Sybase:
-- TRANLOGOPTIONS EXCLUDETRANS <transaction_name>
-- SQL/MX:
-- TRANLOGOPTIONS FILTERTABLE <checkpoint_table_name>
-- Teradata:
-- SQLEXEC "SET SESSION OVERRIDE REPLICATION ON;"
-- SQLEXEC "COMMIT;"
-- Specify API commands if Teradata:
VAM <library name>, PARAMS ("<param>" [, "<param>"] [, ...])
-- Capture before images for conflict resolution:
GETBEFORECOLS (ON <operation> {ALL | KEY | KEYINCLUDING (<col list>) | ALLEXCLUDING
(<col list>)})
-- 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>
    [<encryption options>]]
-- Specify decryption options if input trail is encrypted:
DECRYPTTRAIL <encryption options>
-- Specify the name or IP address of the secondary system
-- and optional encryption of data over TCP/IP:
RMTHOST <system_2>, MGRPORT <portnumber>, ENCRYPT <encryption options>
-- Specify remote trail and encryption options on secondary system:
ENCRYPTTRAIL <encryption options>
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>][, PASSWORD <pw>
    [<encryption options>]]
-- Specify decryption options if input trail is encrypted:
DECRYPTTRAIL <encryption options>
-- Specify error handling rules:
REPERROR (<error>, <response>)
-- Specify tables for delivery and  conflict-resolution routines:
MAP <owner>.*, TARGET <owner>.*,COMPARECOLS (ON <operation> {ALL | KEY |
KEYINCLUDING (<col list>) | ALLEXCLUDING (<col list>)}), RESOLVECONFLICT
(<conflict type> (<resolution name>, <resolution type> COLS (<col>[,...]));
```

```
-- 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 | INTEGRATED TRANLOG}, BEGIN <time>
   [, THREADS <n>]
   ```

   - TRANLOG および INTEGRATED TRANLOG については、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』を参照してください。INTEGRATED TRANLOG により、Oracle データベースに対する統合取得が可能になりました。

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>
       [<encryption options>]]
   -- 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, Oracle, and Sybase:
   -- TRANLOGOPTIONS EXCLUDEUSER <Replicat_user>
   -- Oracle alternative to EXCLUDEUSER:
   -- EXCLUDEUSERID <Oracle uid>
   -- Oracle alternative to the TRANLOGOPTIONS options:
   -- TRACETABLE <trace_table_name>
   -- 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:
GETBEFORECOLS (ON <operation> {ALL | KEY | KEYINCLUDING (<col list>) | ALLEXCLUDING
(<col list>)})
-- 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>
       [<encryption options>]]
   -- Specify decryption options if input trail is encrypted:
   DECRYPTTRAIL <encryption options>
   -- Specify the name or IP address of the primary system
   -- and optional encryption of data over TCP/IP:
   RMTHOST <system_1>, MGRPORT <portnumber>, ENCRYPT <encryption options>
   -- Specify the remote trail and encryption options on the primary system:
   ENCRYPTTRAIL <encryption options>
   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>][, PASSWORD <pw>
       [<encryption options>]]
   -- Specify decryption options if input trail is encrypted:
   DECRYPTTRAIL <encryption options>
   -- Specify error handling rules:
   REPERROR (<error>, <response>)
   -- Specify tables for delivery and  conflict-resolution routines:
   MAP <owner>.*, TARGET <owner>.*,COMPARECOLS (ON <operation> {ALL | KEY |
   KEYINCLUDING (<col list>) | ALLEXCLUDING (<col list>)}), RESOLVECONFLICT
   (<conflict type> (<resolution name>, <resolution type> COLS (<col>[,...]));
   -- 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。

- Replicat が、一意性整合性制約 (PRIMARY KEY 制約や UNIQUE 制約など) に違反する挿入操作または更新操作を適用すると、**一意性競合**が発生します。この競合タイプの例としては、2 つの異なるデータベースから 2 つのトランザクションが発生し、各トランザクションが同じ主キー値で表に行を挿入する場合があげられます。
- Replicat が、同じ行への別の更新と競合する更新を適用すると、**更新競合**が発生します。更新競合が発生するのは、異なるデータベースから生じた 2 つのトランザクションがほぼ同時に同じ行を更新した場合です。証跡レコードに格納されている古い値 (変更前の値) とターゲット・データベース内の同じ行の現行値との間に差異があると、Replicat は更新競合を検出します。
- 2 つのトランザクションが異なるデータベースから生じ、一方が行を削除するのと同時にもう一方が同じ行を更新または削除すると、**削除競合**が発生します。この場合、その行は存在せず、更新も削除もできません。主キーが存在しないため、Replicat は行を見つけられません。

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。

3 つのデータベースが関与するさらに複雑な例をもとに、順序にまつわるさらに複雑な競合について説明します。A、B、C という 3 つのデータベースがあるとします。あるユーザーがデータベース A に行を挿入すると、その行はデータベース B にレプリケートされます。次に、別のユーザーがその行をデータベース B で変更すると、行の変更がデータベース C にレプリケートされます。B からの行の変更が、データベース A からの行の挿入より前にデータベース C に到着した場合、C は競合を検出します。

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。

競合が避けられない場合は、Oracle GoldenGate の競合検出および解決 (CDR) 機能または独自に開発した方法によって、競合をすみやかに特定し、可能なかぎり自動的に解決する必要があります。独自の方法は、SQLEXEC とユーザー・イグジット機能を通じて Oracle GoldenGate 処理に統合できます。

Oracle GoldenGate CDR 機能による競合処理

Oracle GoldenGate の競合検出および解決 (CDR) は、次の処理を行う基本的な競合解決ルーチンを備えています。

- INSERT の一意性競合を解決します。

- 行は存在するが、1 つ以上の列の変更前イメージがデータベースの現行値と異なる場合に発生する、UPDATE の「データが見つからない」競合を解決します。
- 行が存在しない場合に発生する、UPDATE の「データが見つからない」競合を解決します。
- 行は存在するが、1 つ以上の列の変更前イメージがデータベースの現行値と異なる場合に発生する、DELETE の「データが見つからない」競合を解決します。
- 行が存在しない場合に発生する、DELETE の「データが見つからない」競合を解決します。

競合検出および解決 (CDR) を使用するには、データベースが Windows、Linux、UNIX のいずれかのシステムに位置していることが必要です。これは、NonStop プラットフォーム上のデータベースに対してはサポートされません。

CDR は、明示的な変換を行わずに単純な SQL と比較することのできるデータ型をサポートしています。

- NUMERIC
- DATE
- TIMESTAMP
- CHAR/NCHAR
- VARCHAR/NVARCHAR

つまり、これらの列タイプは、COMPARECOLS パラメータおよび GETBEFORECOLS パラメータとともに、また RESOLVECONFLICT パラメータの USEMIN オプションまたは USEMAX オプションで解決列として使用できます。RESOLVECONFLICT の USEDELTA オプションに使用できるのは、NUMERIC 列のみです。 CDR は、LOB、抽象データ型 (ADT) またはユーザー定義型 (UDT) を含む列に使用しないでください。

Replicat が BATCHSQL モードで動作する場合、競合解決は実行されません。BATCHSQL モードで競合が発生すると、Replicat は GROUPTRANSOPS モードに戻り、さらに単一トランザクション・モードに戻ります。競合検出は 3 つのモードすべてで行われます。詳細は、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』の BATCHSQL パラメータの説明を参照してください。

Oracle GoldenGate CDR の構成

ソース・データベース、ターゲット・データベースおよび Oracle GoldenGate を競合検出および解決のために構成するには、次の手順を実行します。手順は次のとおりです。

- 変更前イメージのロギングの有効化
- 競合解決のための Oracle GoldenGate パラメータ・ファイルの構成
- エラー処理のための Oracle GoldenGate パラメータ・ファイルの構成

変更前イメージのロギングの有効化

更新や削除の際に競合検出および解決の基準とするため、列の変更前イメージをログに記録するようソース・データベースを構成する必要があります。たとえば、タイムスタンプ列や在庫量の変更前イメージをログに記録できます。

- 一部のデータベースは、変更前イメージをログ・レコードに記録できないため、サプリメンタル・ロギングを使用してこの処理を行うよう構成する必要があります。ほとんどのサポート対象データベースでは、ADD TRANDATA コマンドをこの目的で使用できます。
- NonStop SQL/MX の場合、表を作成または変更して NOAUDITCOMPRESS 属性を持たせます。
- Oracle GoldenGate でサポートされるすべてのプラットフォーム上の DB2 データベースの場合、GETUPDATEBEFORES、NOCOMPRESSUPDATES、NOCOMPRESSDELETES の各パラメータを使用して、変更前イメージと変更後イメージ全体を CDR で使用できるようにします。

これらのパラメータおよびコマンドの詳細は、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』を参照してください。これらのパラメータの追加情報は、113 ページ以降の例を参照してください。

競合解決のための Oracle GoldenGate パラメータ・ファイルの構成

競合検出および解決をサポートするには、次のパラメータが必要です。

1. Extrat の TABLE パラメータの GETBEFORECOLS オプションを使用し、データベースがログに記録する変更前イメージを取得して証跡に書き込むよう Extract に指示します。Oracle GoldenGate でサポートされるすべてのプラットフォーム上の DB2 データベースの場合、DB2 でサポートされない GETBEFORECOLS パラメータではなく、GETUPDATEBEFORES パラメータを使用します。
2. Replicat パラメータ・ファイル内の MAP パラメータの COMPARECOLS オプションを使用して、Replicat の WHERE 句で変更前の値とともに使用する列を指定します。変更前の値は、更新および削除の競合を検出するために、ターゲット・データベースの現行値と比較されます。(デフォルトでは、Replicat は WHERE 句で主キーのみを使用しますが、これでは競合検出には不十分な場合があります)。
3. MAP パラメータの RESOLVECONFLICT オプションを使用して、様々な操作や競合タイプに対する競合解決ルーチンを指定します。RESOLVECONFLICT を MAP 文で複数回使用すると、競合タイプごとに異なる解決を指定できます。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。

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。これらのパラメータの追加情報は、113 ページ以降の例を参照してください。

エラー処理のための Oracle GoldenGate パラメータ・ファイルの構成

CDR をエラー処理と併用して、解決済のエラーと CDR で解決できなかったエラーを取得する必要があります。

1. REPERROR パラメータを使用して、CDR で解決できないエラーを処理するためのルール、または CDR で処理しないエラーに関するルールを割り当てます。一部のエラーを REPERROR で処理し、残りを CDR で処理するのが適当なこともありますが、同じ競合を処理するように REPERROR と CDR を構成した場合は、CDR が優先されます。INSERTMISSINGUPDATES パラメータと HANDLECOLLISIONS パラメータも、CDR で処理されない一部のエラーを処理するために使用できます。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。

   注意 HANDLECOLLSIONS、INSERTMISSINGUPDATES、REPERROR を使用する場合、これらのパラメータによるエラー処理の前に競合解決が実行されます。

2. (オプション) 例外表を作成します。例外 MAP 文とともに例外表を使用する場合 (手順 3 を参照)、Replicat は競合が発生しているすべての操作を (競合が解決したかどうかにかかわらず)、例外表にマップされる例外 MAP 文に送信します。UPDATE 例外と DELETE 例外を Replicat で処理する場合は、この表の主キーを省略してください。そうしないと、整合性制約エラーが発生する可能性があります。

少なくとも、例外表にターゲット表と同じ列を含める必要があります。これらの行には、Replicat がターゲットに適用した (または適用しようとした) 各行イメージが格納されます。

さらに、追加の列を定義してその他の情報を取得し、データをトランザクション・コンテキストに当てはめるのに役立てることができます。Oracle GoldenGate には、例外 MAP 文を通じてこの情報を取得するためのツールが用意されています (手順 3 を参照)。そのような列には次のものがあります (これだけに限りません)。

- 証跡レコードの変更前イメージ。これは、<col1>_before や <col2>_before などの名前を持つ、ターゲット列の複製です。
- ターゲット列の現行値。これは、<col1>_current や <col2>_current などの名前を持つ、ターゲット列の複製です。
- ターゲット表の名前
- 競合のタイムスタンプ
- 操作タイプ
- データベースのエラー番号
- (オプション) データベースのエラー・メッセージ
- 競合が解決されたかどうか

3. 例外 MAP 文を作成して、例外データを例外表にマップします。例外 MAP 文には次の要素が含まれます。
   - (必須)INSERTALLRECORDS オプション。このパラメータを指定すると、すべての列値が例外表にマップされるように、すべてのマップ済操作が INSERT に変換されます。
   - (必須)EXCEPTIONSONLY オプション。このパラメータを指定すると、エラーを生成する操作がマップされます (成功した操作はマップされません)。
   - (オプション)COLMAP 句。例外表の列の名前および定義がソース表のものと同一であり、例外表にその列のみが含まれている場合、COLMAP は不要です。ただし、名前または定義が異なる場合、あるいは例外表に追加の列があり、それらの列に追加データを移入する場合は、COLMAP 句を使用してすべての列をマップします。

追加データを例外表にマップするためのツール

次に示すのは、追加列に値を移入するために COLMAP 句で使用できるツールです。

- ソース列の名前および定義が例外表のターゲット列のものと同一である場合、明示的にマッピングを行う名前のかわりに USEDEFAULTS キーワードを使用できます。それ以外の場合は、次の例のように、それらの列を COLMAP 句でマッピングする必要があります。

```
COLMAP (exceptions_col1 = col1, [...])
```

- ソース行の変更前イメージを例外表の変更前の列にマップするには、証跡レコードから変更前イメージを取得する before.<column> 構文を使用します。

```
COLMAP (USEDEFAULTS, <exceptions_col1> = before.<source_col1>, &
<exceptions_col2> = before.<source_col2>, [...])
```

- ターゲット行の現行イメージを例外表の現行の列にマップするには、SQLEXEC 問合せを使用してイメージを取得し、COLMAP 句で <queryID>.<column> 構文を使用して問合せの結果を例外表の列にマップします。

  ```
  COLMAP (USEDEFAULTS, name_current = <queryID>.name, phone_current =
  <queryID>.phone, [...])
  ```

- タイムスタンプ、データベース・エラー、その他の環境情報をマップするには、該当する Oracle GoldenGate 列変換関数を使用します。たとえば、次の例では実行時の現行タイムスタンプをマップしています。

  ```
  res_date = @DATENOW()
  ```

例外 MAP の COLMAP でこれらの機能を組み合せて詳細な例外表にデータを移入する方法は、110 ページの「例外表の追加の列を含むサンプル例外マッピング」を参照してください。

ここに示したパラメータおよび列変換関数の使用方法と構文の詳細は、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』を参照してください。

ソース列とターゲット列のみを含むサンプル例外マッピング

次のサンプル・パラメータ・ファイルは、113 ページ以降の CDR の例で使用されるソース表およびターゲット表のエラー処理と単純な例外マッピングを示しています。この例でマップするのはソース列とターゲット列であり、追加の列はマップしません。次の理由から、この例の例外 MAP 文には COLMAP 句は不要です。

- ソースとターゲットの例外列は、名前と定義がそれぞれ同一です。
- 例外表に他の列がありません。

注意 この例では、Replicat パラメータ・ファイルに必要なその他のパラメータ (プロセス名やログイン資格証明、特定のデータベース・タイプに必要なオプションのパラメータなど) を意図的に省略してあります。改行を使用してパラメータ文を複数の行に分割するときは、各行の末尾でアンパサンド (&) を使用します。

```
   -- REPERROR error handling: DEFAULT represents all error types.DISCARD
   -- writes operations that could not be processed to a discard file.
REPERROR (DEFAULT, DISCARD)
   -- Specifies the discard file.
DISCARDFILE ./dirrpt/discards.dsc, PURGE
   -- The regular MAP statement with the CDR parameters
MAP fin.src, TARGET fin.tgt, &
COMPARECOLS (ON UPDATE ALL, ON DELETE ALL), &
RESOLVECONFLICT (UPDATEROWEXISTS, (DEFAULT, USEMAX (last_mod_time)), &
RESOLVECONFLICT (INSERTROWEXISTS, (DEFAULT, USEMAX (last_mod_time)), &
RESOLVECONFLICT (DELETEROWEXISTS, (DEFAULT, OVERWRITE)), &
RESOLVECONFLICT (UPDATEROWMISSING, (DEFAULT, OVERWRITE)), &
RESOLVECONFLICT (DELETEROWMISSING, (DEFAULT, DISCARD)), &
);
   -- Starts the exceptions MAP statement by mapping the source table to the --
   exceptions table.
```

```
MAP fin.src, TARGET fin.exception, &
    -- directs Replicat only to map operations that caused the error specified
    -- in REPERROR.
EXCEPTIONSONLY, &
    -- directs Replicat to convert all the exceptions to inserts into the
    -- exceptions table.This is why there cannot be a primary key constraint
    -- on the exceptions table.
INSERTALLRECORDS &
;
```

例外表の追加の列を含むサンプル例外マッピング

次のサンプル・パラメータ・ファイルは、113 ページ以降の CDR の例で使用されるソース表およびターゲット表のエラー処理と複雑な例外マッピングを示しています。この例では、例外表にソース表と同じ行があり、コンテキスト・データを取得するための追加の列もあります。

注意 この例では、Replicat パラメータ・ファイルに必要なその他のパラメータ (プロセス名やログイン資格証明、特定のデータベース・タイプに必要なオプションのパラメータなど) を意図的に省略してあります。改行を使用してパラメータ文を複数の行に分割するときは、各行の末尾でアンパサンド (&) を使用します。

```
    -- REPERROR error handling: DEFAULT represents all error types.DISCARD
    -- writes operations that could not be processed to a discard file.
REPERROR (DEFAULT, DISCARD)
    -- Specifies the discard file.
DISCARDFILE ./dirrpt/discards.dsc, PURGE
    -- The regular MAP statement with the CDR parameters
MAP fin.src, TARGET fin.tgt, &
COMPARECOLS (ON UPDATE ALL, ON DELETE ALL), &
RESOLVECONFLICT (UPDATEROWEXISTS, (DEFAULT, USEMAX (last_mod_time)), &
RESOLVECONFLICT (INSERTROWEXISTS, (DEFAULT, USEMAX (last_mod_time)), &
RESOLVECONFLICT (DELETEROWEXISTS, (DEFAULT, OVERWRITE)), &
RESOLVECONFLICT (UPDATEROWMISSING, (DEFAULT, OVERWRITE)), &
RESOLVECONFLICT (DELETEROWMISSING, (DEFAULT, DISCARD)), &
);
    -- Starts the exceptions MAP statement by mapping the source table to the --
    exceptions table.
MAP fin.src, TARGET fin.exception, &
    -- directs Replicat only to map operations that caused the error specified
    -- in REPERROR.
EXCEPTIONSONLY, &
    -- directs Replicat to convert all the exceptions to inserts into the
    -- exceptions table.This is why there cannot be a primary key constraint
    -- on the exceptions table.
INSERTALLRECORDS &
    -- SQLEXEC query to select the values from the target record before the
    -- Replicat statement is applied. These are mapped to the "*_target"
    -- columns later.
SQLEXEC (id qry, query "select name, phone, address, salary, balance, & comment,
last_mod_time from fin.tgt where name = :p1", PARAMS(p1 = name )), &
    -- Start of the column mapping, specifies use default column definitions.
```

```
COLMAP ( &
    -- USEDEFAULTS maps the source columns to the target exceptions columns
    -- that receive the "after" image that Replicat applied or tried to apply.
    -- In this case, USEDEFAULTS can be used because the names and definitions -- of
    the source and target exceptions columns are identical; otherwise
    -- the columns must be mapped explicitly in the COLMAP clause.
USEDEFAULTS, &
    -- captures the timestamp when the resolution was performed.
res_date = @DATENOW(), &
    -- captures and maps the DML operation type.
optype = @GETENV("LASTERR", "OPTYPE"), &
    -- captures and maps the database error number that was returned.
dberrnum = @GETENV("LASTERR", "DBERRNUM"), &
    -- captures and maps the database error that was returned.
dberrmsge = @GETENV("LASTERR", "DBERRMSG"), &
    -- captures and maps the name of the target table
tabname = @GETENV("GGHEADER", "TABLENAME"), &
    -- If the names and definitions of the source columns and the target
    -- exceptions columns were not identical, the columns would need to
    -- be mapped in the COLMAP clause instead of using USEDEFAULTS:
-- name_after = name, &
-- phone_after = phone, &
-- address_after = address, &
-- salary_after = salary, &
-- balance_after = balance, &
-- comment_after = comment, &
-- last_mod_time_after = last_mod_time &
    -- maps the before image of each column from the trail to a column in the
    -- exceptions table.
name_before = before.name, &
phone_before = before.phone, &
address_before = before.address, &
salary_before = before.salary, &
balance_before = before.balance, &
comment_before = before.comment, &
last_mod_time_before = before.last_mod_time, &
    -- maps the results of the SQLEXEC query to rows in the exceptions table
    -- to show the current image of the row in the target.
name_current = qry.name, &
phone_current = qry.phone, &
address_current = qry.address, &
salary_current = qry.salary, &
balance_current = qry.balance, &
comment_current = qry.comment, &
last_mod_time_current = qry.last_mod_time) &
;
```

例外表の作成と例外マッピングの使用の詳細は、167 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。

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。

CDR 統計の表示

CDR 機能には、競合解決の結果を表示するために次の方法が用意されています。

レポート・ファイル

Replicat は、CDR 統計をレポート・ファイルに書き込みます。

```
Total CDR conflicts7
    CDR resolutions succeeded            6
    CDR resolutions failed               1
    CDR INSERTROWEXISTS conflicts        1
    CDR UPDATEROWEXISTS conflicts        4
    CDR DELROWEXISTS conflicts           1
    CDR DELROWMISSING conflicts          1
```

GGSCI

CDR 統計は、STATS REPLICAT コマンドを REPORTCDR オプションとともに使用して、GGSCI から表示することもできます。

```
STATS REPLICAT <group>, REPORTCDR
```

列変換関数

次の CDR 統計を取得し、必要に応じて、例外表にマップしたり、列変換関数からの入力を受け付ける他の Oracle GoldenGate パラメータで使用できます。

- Replicat で検出した競合の数
- Replicat で解決した解決の数
- Replicat で解決できなかった解決の数

これらの統計を取得するには、"STATS" または "DELTASTATS" 情報タイプを指定して @GETENV 列変換関数を使用します。結果は、現在の Replicat セッションに基づいています。Replicat が停止して再起動すると、統計はリセットされます。

特定の表またはワイルドカードで指定された表のセットについて統計が返されます。

```
@GETENV("STATS","TABLE","SCHEMA.TABLNAME","CDR_CONFLICTS")
@GETENV("STATS","TABLE","SCHEMA.TABLNAME","CDR_RESOLUTIONS_SUCCEEDED")
@GETENV("STATS","TABLE","SCHEMA.TABLNAME","CDR_RESOLUTIONS_FAILED")
```

Replicat パラメータ・ファイルのすべての MAP 文のすべての表について統計が返されます。

```
@GETENV("STATS","CDR_CONFLICTS")
@GETENV("STATS","CDR_RESOLUTIONS_SUCCEEDED")
@GETENV("STATS","CDR_RESOLUTIONS_SUCCEEDED")
```

前述の例の "STATS" 情報タイプを "DELTASTATS" に置き換えると、前回の "DELTASTATS" の実行以降の数が返されます。

@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。

CDR の例 1: USEMAX、OVERWRITE、DISCARD によるすべての競合タイプの処理

この例で使用する表

各例では同一の Oracle データベースを想定しています。

```
CREATE TABLE tgt(
    name varchar2(30) primary key,
    phone varchar2(10),
    address varchar2(100),
    salary number,
    balance number,
    comment varchar2(100),
    last_mod_time timestamp);
```

ソース・データベースでは、すべての列が補足的にログに記録されます。

```
ADD TRANDATA scott.src, COLS (name, phone, address, salary, balance, comment,
last_mod_time);
```

競合解決を指定した MAP 文

```
MAP fin.src, TARGET fin.tgt,
    COMPARECOLS (ON UPDATE ALL, ON DELETE ALL),
    RESOLVECONFLICT (UPDATEROWEXISTS, (DEFAULT, USEMAX (last_mod_time)),
    RESOLVECONFLICT (INSERTROWEXISTS, (DEFAULT, USEMAX (last_mod_time)),
    RESOLVECONFLICT (DELETEROWEXISTS, (DEFAULT, OVERWRITE)),
    RESOLVECONFLICT (UPDATEROWMISSING, (DEFAULT, OVERWRITE)),
    RESOLVECONFLICT (DELETEROWMISSING, (DEFAULT, DISCARD)),
    );
```

MAP 文の説明

- COMPARECOLS では、更新および削除の Replicat WHERE 句において、証跡レコード内のすべての列の変更前イメージを使用します。
- DEFAULT では、すべての競合タイプの列グループとしてすべての列を使用します。そのため、解決はすべての列に適用されます。
- INSERTROWEXISTS 競合には、USEMAX 解決を使用します。つまり、挿入時に行が存在する場合は、last_mod_time 列を解決列として使用し、証跡の値とデータベースの値のどちらが大きいかを判別します。証跡の値の方が大きい場合、レコードを適用しますが、挿入を更新に変更します。データベースの値の方が大きい場合、レコードを無視します。
- UPDATEROWEXISTS 競合には、USEMAX 解決を使用します。つまり、更新時に行が存在する場合は、last_mod_time 列を解決列として使用します。証跡の値の方が大きい場合、更新を適用します。

- DELETEROWEXISTS 競合には、OVERWRITE 解決を使用します。つまり、削除操作時に行が存在する場合は、削除を適用します。
- UPDATEROWMISSING 競合には、OVERWRITE 解決を使用します。つまり、更新時に行が存在しない場合は、更新を挿入に変更して適用します。
- DELETROWMISSING 競合には、DISCARD 解決を使用します。つまり、削除操作時に行が存在しない場合は、証跡レコードを破棄します。

エラー処理

例外表に対するエラー処理の例は、107 ページの「エラー処理のための Oracle GoldenGate パラメータ・ファイルの構成」を参照してください。

USEMAX 解決による INSERTROWEXISTS の処理

この例では、証跡とデータベースのレコードに対応する変更前イメージと変更後イメージを使用して USEMAX 解決を説明しています。ここでは、ソースとターゲットに行が存在するが、一部またはすべての行の値が異なる場合の挿入の解決方法を示します。

表 3　USEMAX 解決による INSERTROWEXISTS 競合の処理

イメージ	SQL	コメント
証跡内の変更前イメージ	None (row was inserted on the source).	
証跡内の変更後イメージ	name='Mary' phone='1234567890' address='Oracle Pkwy' salary=100 balance=100 comment=NULL last_mod_time='9/1/10 3:00'	last_mod_time='9/1/10 3:00 は、解決列の変更後イメージです。変更後イメージが存在するため、これが解決の特定に使用されます。
ターゲット・データベースのイメージ	name='Mary' phone='111111' address='Ralston' salary=200 balance=500 comment='aaa' last_mod_time='9/1/10 1:00'	last_mod_time='9/1/10 1:00 は、ターゲットの解決列の現行イメージであり、証跡内の解決列の値はこれと比較されます。

表 3　USEMAX 解決による INSERTROWEXISTS 競合の処理 (続き)

イメージ	SQL	コメント
Replicat によって適用され、競合を検出する初期 INSERT	SQL バインド変数 : 1)'Mary' 2)'1234567890' 3)'Oracle Pkwy' 4)100 5)100 6)NULL 7)'9/1/10 3:00'	この SQL は、Mary に対する一意性競合を戻します。
競合を解決するために Replicat によって適用される UPDATE	SQL バインド変数 : 1)'1234567890' 2)'Oracle Pkwy' 3)100 4)100 5)NULL 6)'9/1/10 3:00' 7)'Mary' 8)'9/1/10 3:00'	INSERTROWEXISTS に対して USEMAX が指定されているため、Replicat は挿入を更新に変換し、証跡レコードの last_mod_time の値をデータベースの値と比較します。レコードの値の方が大きいため、証跡ファイル内の列の変更後イメージがターゲットに適用されます。

USEMAX 解決による UPDATEROWEXISTS の処理

この例では、証跡とデータベースのレコードに対応する変更前イメージと変更後イメージを使用して USEMAX 解決を説明しています。ここでは、ソースとターゲットに行が存在するが、一部またはすべての行の値が異なる場合の更新の解決方法を示します。

表 4　USEMAX 解決による UPDATEROWEXISTS 競合の処理

イメージ	SQL	コメント
証跡内の変更前イメージ	name='Mary' phone='1234567890' address='Oracle Pkwy' salary=100 balance=100 comment=NULL last_mod_time='9/1/10 3:00'	last_mod_time='9/1/10 3:00 は、解決列の変更前イメージです。
証跡内の変更後イメージ	phone='222222' address='Holly' last_mod_time='9/1/10 5:00'	last_mod_time='9/1/10 5:00 は、解決列の変更後イメージです。変更後イメージが存在するため、これが解決の特定に使用されます。

表 4　USEMAX 解決による UPDATEROWEXISTS 競合の処理 (続き)

イメージ	SQL	コメント
ターゲット・データベースのイメージ	name='Mary' phone='1234567890' address='Oracle Pkwy' salary=100 balance=600 comment='com' last_mod_time='9/1/10 6:00'	last_mod_time='9/1/10 6:00 は、ターゲットの解決列の現行イメージであり、証跡内の解決列の値はこれと比較されます。
Replicat によって適用され、競合を検出する初期 UPDATE	SQL バインド変数 : 1)'222222' 2)'Holly' 3)'9/1/10 5:00' 4)'Mary' 5)'1234567890' 6)'Oracle Pkwy' 7)100 8)100 9)NULL 10)'9/1/10 3:00'	balance、comment および last_mod_time の値がターゲット内で異なるため、この SQL はデータが見つからないというエラーを戻します。 COMPARECOLS 文が ALL に設定されているため、すべての列が WHERE 句で使用されます。
競合を解決するために Replicat によって適用される UPDATE	SQL バインド変数 : 1)'Mary' 2)'222222' 3)'Holly' 4)100 5)100 6)NULL 7)'9/1/10 5:00' 8)'Mary' 9)'9/1/10 5:00'	証跡レコードの last_mod_time の変更後の値の方がデータベースの現行値より小さいため、データベースの値は維持されます。Replicat は、主キーと、9/1/10 5:00 より小さく設定された last_mod_time 値を含む WHERE で操作を適用します。この基準に一致する行はないため、データが見つからないというエラーが発生して文は失敗しますが、条件に一致しない場合は USEMAX 解決が失敗すると予想されることから、Replicat はエラーを無視します。

OVERWRITE 解決による UPDATEROWMISSING の処理

この例では、証跡とデータベースのレコードに対応する変更前イメージと変更後イメージを使用して OVERWRITE 解決を説明しています。ここでは、ターゲット行が存在しない場合の解決方法を示します。論理的な解決は、行をターゲットに上書きして両方のデータベースを再度同期させることであり、この方法が使用されます。

表 5　OVERWRITE 解決による UPDATEROWMISSING 競合の処理

イメージ	SQL	コメント
証跡内の変更前イメージ	name='Jane' phone='333' address='Oracle Pkwy' salary=200 balance=200 comment=NULL last_mod_time='9/1/10 7:00'	
証跡内の変更後イメージ	phone='4444' address='Holly' last_mod_time='9/1/10 8:00'	
ターゲット・データベースのイメージ	None (row for Jane is missing)	
Replicat によって適用され、競合を検出する初期 UPDATE	SQL バインド変数 : 1)'4444' 2)'Holly' 3)'9/1/10 8:00' 4)'Jane' 5)'333' 6)'Oracle Pkwy' 7)200 8)200 9)NULL 10)'9/1/10 7:00'	この SQL は、データが見つからないというエラーを戻します。COMPARECOLS 文が ALL に設定されているため、すべての列が WHERE 句で使用されます。
競合を解決するために Replicat によって適用される INSERT	SQL バインド変数 : 1)'Jane' 2)'4444' 3)'Holly' 4)200 5)200 6)NULL 7)'9/1/10 8:00'	OVERWRITE が解決であるため、更新は挿入に変更されます。列の変更後イメージがあれば使用され、ない場合は変更前イメージが使用されます。

DISCARD 解決による DELETEROWMISSING の処理

この例では、証跡とデータベースのレコードに対応する変更前イメージと変更後イメージを使用して DISCARD 解決を説明しています。ここでは、ターゲット行が存在しない場合の解決方法を示します。ソースに対する削除の場合、ターゲット行が存在しなくてもかまわない (ターゲット行はいずれにせよ削除する必要がある) ため、解決方法は証跡内の DELETE 操作を破棄することです。

表 6　DISCARD 解決による DELETEROWMISSING 競合の処理

イメージ	SQL	コメント
証跡内の変更前イメージ	name='Jane' phone='4444' address='Holly' salary=200 balance=200 comment=NULL last_mod_time='9/1/10 8:00'	
証跡内の変更後イメージ	None	
ターゲット・データベースのイメージ	None (row missing)	
Replicat によって適用され、競合を検出する初期 DELETE	SQL バインド変数 : 1)'Jane' 2)'4444' 3)'Holly' 4)200 5)200 6)NULL 7)'9/1/10 8:00'	この SQL は、データが見つからないというエラーを戻します。COMPARECOLS 文が ALL に設定されているため、すべての列が WHERE 句で使用されます。
競合を解決するために Replicat によって適用される SQL	None	DELETEROWMISSING の解決として DISCARD が指定されているため、証跡からの削除が廃棄ファイルに格納されます。

OVERWRITE 解決による DELETEROWEXISTS の処理

この例では、証跡とデータベースのレコードに対応する変更前イメージと変更後イメージを使用して OVERWRITE 解決を説明しています。ここでは、ソース行が削除されたがターゲット行は存在する場合の解決方法を示します。この場合、OVERWRITE 解決により、ターゲットに削除が適用されます。

表 7　OVERWRITE 解決による DELETEROWEXISTS 競合の処理

イメージ	SQL	コメント
証跡内の変更前イメージ	name='Mary' phone='222222' address='Holly' salary=100 balance=100 comment=NULL last_mod_time='9/1/10 5:00'	
証跡内の変更後イメージ	None	

表 7　OVERWRITE 解決による DELETEROWEXISTS 競合の処理 (続き)

イメージ	SQL	コメント
ターゲット・データベースのイメージ	name='Mary' phone='1234567890' address='Oracle Pkwy' salary=100 balance=600 comment=com last_mod_time='9/1/10 7:00'	行はターゲットに存在しますが、phone、address、balance、comment、last_mod_time の各列が証跡内の変更前イメージと異なっています。
Replicat によって適用され、競合を検出する初期 DELETE	SQL バインド変数 : 1)'Mary' 2)'222222' 3)'Holly' 4)100 5)100d 6)NULL 7)'9/1/10 5:00'	COMPARECOLS 文が ALL に設定されているため、すべての列が WHERE 句で使用されます。 変更前の値と現行値が異なるため、データが見つからないというエラーが発生します。
競合を解決するために Replicat によって適用される DELETE	SQL バインド変数 : 1)'Mary'	OVERWRITE が解決であるため、(整合性エラーを回避するために) 主キーのみを使用して DELETE が適用されます。

CDR の例 2: USEDELTA および USEMAX による UPDATEROWEXISTS の処理

この例で使用する表

各例では同一の Oracle データベースを想定しています。

```
CREATE TABLE tgt(
    name varchar2(30) primary key,
    phone varchar2(10),
    address varchar2(100),
    salary number,
    balance number,
    comment varchar2(100),
    last_mod_time timestamp);
```

ソース・データベースでは、すべての列が補足的にログに記録されます。

```
ADD TRANDATA scott.src, COLS (name, phone, address, salary, balance, comment,
last_mod_time);
```

MAP 文

```
MAP fin.src, TARGET fin.tgt,
    COMPARECOLS
    (ON UPDATE KEYINCLUDING (address, phone, salary, last_mod_time),
    ON DELETE KEYINCLUDING (address, phone, salary, last_mod_time)),
    RESOLVECONFLICT (
    UPDATEROWEXISTS,
    (delta_res_method, USEDELTA, COLS (salary)),
    (DEFAULT, USEMAX (last_mod_time)));
```

説明

- UPDATE にターゲット行は存在するが、キー以外の列が異なる UPDATEROWEXISTS 競合には、列に応じて 2 つの異なる解決を使用します。
  - delta_res_method 解決では、salary 列に対して USEDELTA 解決ロジックを使用し、値の変更が列の現行値に追加されるようにします。
  - DEFAULT では、last_mod_time 列を解決列として使用し、表 (デフォルトの列グループ) のその他すべての列に対して USEMAX 解決ロジックを使用します。この列は行が変更されるたびに現在の時間で更新され、証跡内のこの列の値がターゲットの値と比較されます。証跡レコード内の last_mod_time の値がターゲット・データベースの last_mod_time の現行値より大きい場合は、name、phone、address、balance、comment および last_mod_time の変更がターゲットに適用されます。
- COMPARECOLS では、主キー (name 列) と address、phone、salary、last_mod_time の各列を UPDATE 操作と DELETE 操作の競合検出のための比較列として使用します。(balance 列と comment 列は比較されません。)

エラー処理

例外表に対するエラー処理の例は、107 ページの「エラー処理のための Oracle GoldenGate パラメータ・ファイルの構成」を参照してください。

表 8　USEDELTA および USEMAX による UPDATEROWEXISTS の処理

イメージ	SQL	コメント
証跡内の変更前イメージ	name='Mary' phone='1234567890' address='Oracle Pkwy' salary=100 balance=100 comment=NULL last_mod_time='9/1/10 3:00'	last_mod_time='9/1/10 3:00 は、USEMAX 解決の解決列の変更前イメージです。 salary=100 は、USEDELTA 解決の変更前イメージです。
証跡内の変更後イメージ	phone='222222' address='Holly' salary=200 comment='new' last_mod_time='9/1/10 5:00'	last_mod_time='9/1/10 5:00 は、USEMAX の解決列の変更後イメージです。変更後イメージが存在するため、これが解決の特定に使用されます。

表 8　USEDELTA および USEMAX による UPDATEROWEXISTS の処理 (続き)

イメージ	SQL	コメント

ターゲット・データベースのイメージ	`name='Mary'`
	`phone='1234567890'`
	`address='Oracle Pkwy'`
	`salary=600`
	`balance=600`
	`comment='com'`
	`last_mod_time='9/1/10 4:00'`
		last_mod_time='9/1/10 4:00 は、ターゲットの解決列の現行イメージであり、証跡内の解決列の値はこれと比較されます。
		salary=600 は、USEDELTA 解決のターゲット列の現行イメージです。

Replicat によって適用され、競合を検出する初期 UPDATE	SQL バインド変数 :
	`1)'222222'`
	`2)'Holly'`
	`3)200`
	`4)'new'`
	`5)'9/1/10 5:00'`
	`6)'Mary'`
	`7)'1234567890'`
	`8)'Oracle Pkwy'`
	`9)100`
	`10)'9/1/10 3:00'`
		salary と last_mod_time の値が異なるため、この SQL はデータが見つからないというエラーを戻します。(comment と balance の値も異なりますが、これらの列は比較されません。)

USEDELTA を使用して salary の競合を解決するために Replicat によって適用される UPDATE。	SQL バインド変数 :
	`1)200`
	`2)100`
	`3)'Mary'`
		USEDELTA では、証跡内の salary の変更後イメージ (200) と証跡内の salary の変更前イメージ (100) の差が、ターゲットの salary の現行値 (600) に追加されます。結果は 700 です。
		`600 + (200 - 100) = 700`

USEMAX を使用してデフォルト列の競合を解決するために Replicat によって適用される UPDATE。	SQL バインド変数 :
	`1)'222222'`
	`2)'Holly'`
	`3)'new'`
	`4)'9/1/10 5:00'`
	`5)'Mary'`
	`6)'9/1/10 5:00'`
		USEMAX では、証跡レコード内の last_mod_time の変更後の値がデータベースの現行値より大きいため、証跡レコードの変更後の値で行が更新されます。
		salary 列は、USEDELTA 解決の UPDATE で解決されるため、ここでは設定していません。

CDR の例 3: USEDELTA、USEMAX および IGNORE による UPDATEROWEXISTS の処理

この例で使用する表

各例では同一の Oracle データベースを想定しています。

```
CREATE TABLE tgt(
    name varchar2(30) primary key,
    phone varchar2(10),
    address varchar2(100),
    salary number,
    balance number,
    comment varchar2(100),
    last_mod_time timestamp);
```

ソース・データベースでは、すべての列が補足的にログに記録されます。

```
ADD TRANDATA scott.src, COLS (name, phone, address, salary, balance, comment,
last_mod_time);
```

MAP 文

```
MAP fin.src, TARGET fin.tgt,
    COMPARECOLS
    (ON UPDATE ALLEXCLUDING (comment)),
    RESOLVECONFLICT (
    UPDATEROWEXISTS,
    (delta_res_method, USEDELTA, COLS (salary, balance)),
    (max_res_method, USEMAX (last_mod_time), COLS (address, last_mod_time)),
    (DEFAULT, IGNORE));
```

説明

- UPDATE にターゲット行は存在するが、キー以外の列が異なる UPDATEROWEXISTS 競合には、列に応じて 2 つの異なる解決を使用します。
  - delta_res_method 解決では、salary 列と balance 列に対して USEDELTA 解決ロジックを使用し、各値の変更が各列の現行値に追加されるようにします。
  - max_res_method 解決では、address 列と last_mod_time 列に対して USEMAX 解決ロジックを使用します。last_mod_time 列が解決列です。この列は行が変更されるたびに現在の時間で更新され、証跡内のこの列の値がターゲットの値と比較されます。証跡レコード内の last_mod_time の値がターゲット・データベースの last_mod_time の現行値より大きい場合は、address と last_mod_time の変更がターゲットに適用されます。そうでない場合、変更は無視され、ターゲット値が維持されます。
  - DEFAULT では、表 (デフォルトの列グループ) の残りの列 (phone と comment) に対して IGNORE 解決ロジックを使用します。Replicat では、これらの列の変更は常に無視されます。
- COMPARECOLS では、comment 列を除くすべての列を、UPDATE 操作の競合検出における比較列として使用します。comment は更新の WHERE 句では使用されませんが、証跡レコード内に変更前イメージを持つその他すべての列が使用されます。

エラー処理

例外表に対するエラー処理の例は、107 ページの「エラー処理のための Oracle GoldenGate パラメータ・ファイルの構成」を参照してください。

表 9　USEDELTA、USEMAX および IGNORE による UPDATEROWEXISTS の処理

イメージ	SQL	コメント

証跡内の変更前イメージ	`name='Mary'`
	`phone='1234567890'`
	`address='Oracle Pkwy'`
	`salary=100`
	`balance=100`
	`comment=NULL`
	`last_mod_time='9/1/10 3:00`
		last_mod_time='9/1/10 3:00 は、USEMAX 解決の解決列の変更前イメージです。
		salary=100 と balance=100 は、USEDELTA 解決の変更前イメージです。

証跡内の変更後イメージ	`phone='222222'`
	`address='Holly'`
	`salary=200`
	`comment='new'`
	`last_mod_time='9/1/10 5:00'`
		last_mod_time='9/1/10 5:00 は、USEMAX の解決列の変更後イメージです。変更後イメージが存在するため、これが解決の特定に使用されます。
		salary=200 は、USEDELTA 解決の唯一の変更後イメージです。balance については、変更前イメージが計算で使用されます。

ターゲット・データベースのイメージ	`name='Mary'`
	`phone='1234567890'`
	`address='Ralston'`
	`salary=600`
	`balance=600`
	`comment='com'`
	`last_mod_time='9/1/10 4:00'`
		last_mod_time='9/1/10 4:00 は、USEMAX のターゲットの解決列の現行イメージであり、証跡内の解決列の値はこれと比較されます。
		salary=600 と balance=600 は、USEDELTA のターゲット列の現行イメージです。

Replicat によって適用され、競合を検出する初期 UPDATE	SQL バインド変数 :
	`1)'222222'`
	`2)'Holly'`
	`3)200`
	`4)'new'`
	`5)'9/1/10 5:00'`
	`6)'Mary'`
	`7)'1234567890'`
	`8)'Oracle Pkwy'`
	`9)100`
	`10)100`
	`11)'9/1/10 3:00'`
		address、salary、balance、last_mod_time の各列の値が異なるため、この SQL はデータが見つからないというエラーを戻します。

表 9　USEDELTA、USEMAX および IGNORE による UPDATEROWEXISTS の処理 (続き)

イメージ	SQL	コメント
USEDELTA を使用して salary の競合を解決するために Replicat によって適用される UPDATE。	SQL バインド変数 : 1)200 2)100 3)'Mary'	salary には 100 の差がありますが、balance の値に変更がなかったため、更新の SQL では salary は不要です。USEDELTA では、証跡内の salary の変更後イメージ (200) と変更前イメージ (100) の差が、ターゲットの salary の現行値 (600) に追加されます。結果は 700 です。
USEMAX の競合を解決するために Replicat によって適用される UPDATE。	SQL バインド変数 : 1)'Holly' 2)'9/1/10 5:00' 3)'Mary' 4)'9/1/10 5:00'	証跡レコード内の last_mod_time の変更後の値がデータベースの現行値より大きいため、その列と address 列が証跡レコードの変更後の値で更新されます。 salary 列は、USEDELTA 解決の UPDATE で解決されるため、ここでは設定していません。
IGNORE に対し、Replicat によって適用される UPDATE。	SQL バインド変数 : 1)'222222' 2)'new' 3)'Mary'	DEFAULT 列グループ (phone と comment) に IGNORE が指定されているため、解決の SQL は適用されません。

第 10 章

Oracle GoldenGate のセキュリティの構成

Oracle GoldenGate のセキュリティ・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 10　Oracle GoldenGate のセキュリティ・オプション[1]

セキュリティ機能	保護の対象	説明
証跡ファイル暗号化 126 ページ	データ・リンク経由およびファイル内で証跡ファイルまたは抽出ファイルに書き込まれるレコード。	次のいずれかの方法を使用します。 ◆ 256 鍵バイト置換[2] ◆ 任意の Advanced Encryption Security(AES)[3]暗号 : AES-128 AES-192 AES-256
パスワード暗号化 130 ページ	Oracle GoldenGate コンポーネントでデータベースへのログインに使用されるパスワード。平文のパスワードは、GGSCI で明示的に暗号化する必要があります。	次のいずれかの方法を使用します。 ◆ AES-128 ◆ AES-192 ◆ AES-256 ◆ Blowfish[4]
TCP/IP 暗号化 133 ページ	TCP/IP を通じて送信されるデータ。データは、ターゲットで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)。	次のいずれかの方法を使用します。 ◆ AES-128 ◆ AES-192 ◆ AES-256 ◆ Blowfish
コマンドの認証 135 ページ	GGSCI を通じて発行される Oracle GoldenGate コマンド。	CMDSEC(コマンド・セキュリティ) ファイルを構成します。

表 10　Oracle GoldenGate のセキュリティ・オプション[1](続き)

セキュリティ機能	保護の対象	説明
信頼できる接続 137 ページ	ファイアウォールの先にあるホストへの TCP/IP 接続。	次のいずれかの方法を使用します。 ◆ AES-128 ◆ AES-192 ◆ AES-256 ◆ Blowfish

[1] データ環境の保護に関する追加ガイドラインは、『Oracle セキュリティ・ガイド』を参照してください。

[2] バイト置換では、平文の各バイトが異なる暗号文値に単純に置き換えられます。

[3] Advanced Encryption Standard(AES) は、高度なデータ・セキュリティを必要とする政府機関やその他の組織で使用されている対称鍵暗号化標準です。128 ビット鍵暗号、192 ビット鍵暗号、256 ビット鍵暗号という 3 種類の 128 ビット・ブロック暗号が用意されています。Oracle 以外のデータベースで AES を使用するには、LD_LIBRARY_PATH 変数または SHLIB_PATH 変数 (UNIX)、あるいは PATH 変数 (Windows) を使用して、Oracle GoldenGate インストール・ディレクトリの lib サブディレクトリのパスを設定する必要があります。

[4] Blowfish 暗号化 : 鍵を使用した対称ブロック暗号です。Oracle GoldenGate による Blowfish の実装では、ブロック・サイズは 64 ビット、鍵サイズは 32 ～ 128 ビットの可変長です。

証跡またはファイルの暗号化

この項では、Oracle GoldenGate によって処理されるデータを格納する証跡または抽出ファイルの暗号化設定の指定方法を示します。パラメータ・ファイル内の正しい位置については、127 ページの例を参照してください。

Extract のパラメータ・ファイル

この Extract が書き込む証跡またはファイルを指定した EXTTRAIL、RMTTRAIL、EXTFILE、RMTFILE のいずれかのパラメータの前に ENCRYPTTRAIL パラメータ文を追加します。

```
ENCRYPTTRAIL [{AES128 | AES192 | AES256} KEYNAME <keyname>]
```

条件 :

- AES128 の場合、AES-128 で暗号化されます。
- AES192 の場合、AES-192 で暗号化されます。
- AES256 の場合、AES-256 で暗号化されます。
- KEYNAME <keyname> では、ENCKEYS ファイルにある暗号化鍵の論理参照名を指定します。134 ページの「暗号化鍵の生成」を参照してください。AES オプションには KEYNAME が必須です。
- ENCRYPTTRAIL を単独で(入力引数なしで)指定すると、256鍵バイト置換でファイルが暗号化されます。この方法は安全ではないため、本番環境では使用しないでください。旧バージョンの Oracle GoldenGate との下位互換性を確保する目的でのみ使用してください。

注意　一般的ではありませんが、1 つの Extract が複数の証跡ファイルに書き込むことも可能です。そのような構成では、ENCRYPTTRAIL 文を 1 つ使用して、それらすべてを暗号化することができます。いずれかの証跡またはファイルを暗号化しない場合は、それに対応する EXTTRAIL、RMTTRAIL、EXTFILE または RMTFILE 文を ENCRYPTTRAIL 文の *前* に置いてください。

データ・ポンプの Extract パラメータ・ファイル

1. データ・ポンプが読み取る証跡またはファイルを復号化して、データ・ポンプが処理できるようにするには、DECRYPTTRAIL パラメータ文を追加します。復号化アルゴリズムと鍵は、証跡の暗号化に使用したものと一致している必要があります。

   ```
   DECRYPTTRAIL [{AES128 | AES192 | AES256} KEYNAME <keyname>]
   ```

   条件 :

   - ENCRYPTTRAIL をオプションなしで使用した場合、DECRYPTTRAIL をオプションなしで使用します。
   - ENCRYPTTRAIL で AES 暗号を使用した場合、AES128、AES192、AES256 のうち、一致するものを使用します。
   - AES を使用する場合、KEYNAME <keyname> を使用し、ソースで使用したものと同じ論理参照名を指定します。

2. データ・ポンプが書き込む証跡またはファイルを暗号化するには、EXTTRAIL、RMTTRAIL、EXTFILE、RMTFILE のいずれかのパラメータ文の前に ENCRYPTTRAIL 文を追加します。この文は、DECRYPTTRAIL 文の後に置きます。

   注意 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、ENCRYPTTRAIL は使用できません。証跡またはファイルは、デフォルトの正規形式で書き込む必要があります。また、AES オプションに設定した ENCRYPTTRAIL を ETOLDFORMAT パラメータとともに使用することはできません。ETOLDFORMAT では、256 鍵バイト置換のみがサポートされます。

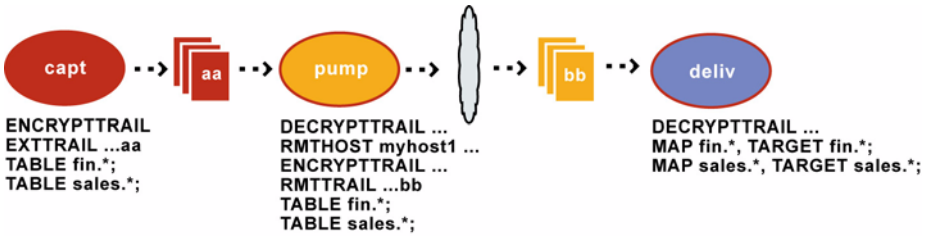
Replicat のパラメータ・ファイル

Replicat の読み取る証跡が暗号化されている場合は、DECRYPTTRAIL パラメータ文を追加します。復号化アルゴリズムと鍵は、証跡の暗号化に使用したものと一致している必要があります。

パラメータ・ファイルの例 : すべての証跡を暗号化

この例は、プライマリ Extract、データ・ポンプ、Replicat を 1 つずつ使用する単純な構成で、様々な証跡に対して暗号化オプションを使用する方法を示しています。ここでは、プライマリ Extract とデータ・ポンプがそれぞれ 1 つの証跡に書き込むと仮定します。次に示す例では、Oracle GoldenGate データベース・ユーザーのパスワードがパスワード暗号化によって暗号化されます (130 ページを参照)。128 ページの「パラメータ・ファイルの例 : 一方の証跡を暗号化し、もう一方は暗号化しない」も参照してください。

注意 次の例は、暗号化オプションに注目した基本的なパラメータ・ファイルであり、特定の Oracle GoldenGate 構成で使用する必要のあるパラメータをすべて示しているとは限りません。



Extract のパラメータ・ファイル

```
EXTRACT capt
USERID ogg, PASSWORD AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC, &
    AES128 KEYNAME mykey1
DISCARDFILE /ogg/capt.dsc, PURGE
-- Encrypt the output trail.Use AES-192 with encryption key mykey1.
ENCRYPTTRAIL AES192 KEYNAME mykey1
EXTTRAIL /ogg/dirdat/aa
TABLE FIN.*;
TABLE SALES.*;
```

データ・ポンプのパラメータ・ファイル

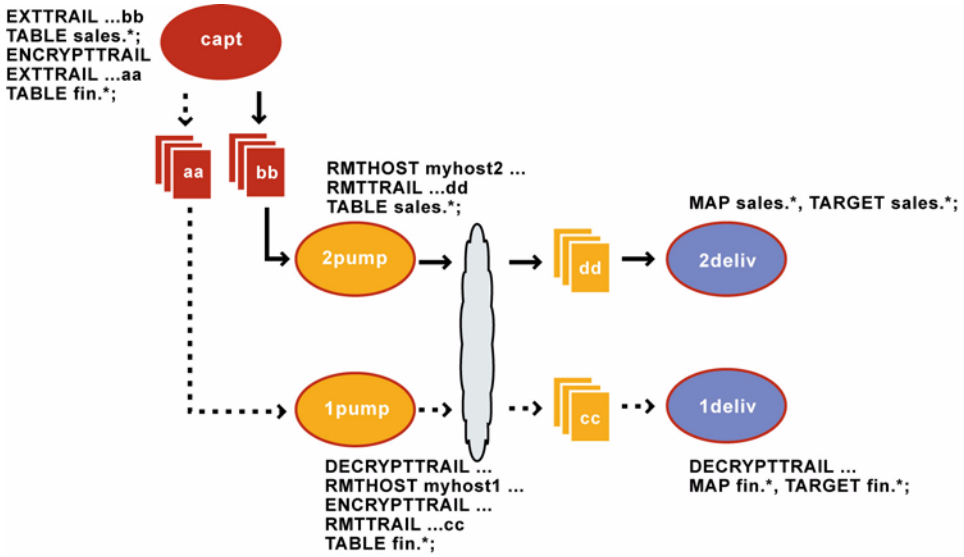
```
EXTRACT pump
USERID ogg, PASSWORD AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC, &
    AES128 KEYNAME mykey1
DISCARDFILE /ogg/pmp.dsc, PURGE
-- Decrypt the input trail.Use encryption key mykey1.
DECRYPTTRAIL AES192 KEYNAME mykey1
-- Encrypt the output trail.Use AES-192 and encryption key mykey2.
RMTHOST myhost1, MGRPORT 7809
ENCRYPTTRAIL AES192 KEYNAME mykey2
RMTTRAIL /ogg/dirdat/bb
TABLE FIN.*;
TABLE SALES.*;
```

Replicat のパラメータ・ファイル

```
REPLICAT deliv
USERID ogg, PASSWORD AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC, &
    AES128 KEYNAME mykey1
ASSUMETARGETDEFS
DISCARDFILE /ogg/deliv.dsc, PURGE
-- Decrypt the input trail.Use encryption key mykey2.
DECRYPTTRAIL AES192 KEYNAME mykey2
MAP FIN.*, TARGET FIN.*;
MAP SALES.*, TARGET SALES.*;
```

パラメータ・ファイルの例 : 一方の証跡を暗号化し、もう一方は暗号化しない

この例は、同じ Extract プロセスまたはデータ・ポンプ・プロセスが書き込む様々な証跡またはファイルに対して、暗号化を有効および無効にする方法を示しています。このタイプの構成では、暗号化*しない*証跡またはファイルを指定した EXTTRAIL、RMTTRAIL、EXTFILE、RMTFILE のいずれかのパラメータを、ENCRYPTTRAIL 文の前に置く必要があります (これにより、その文より後に指定した証跡またはファイルの暗号化が有効になります)。次に示す例では、Oracle GoldenGate データベース・ユーザーのパスワードがパスワード暗号化によって暗号化されます (130 ページを参照)。



Extract のパラメータ・ファイル

この Extract プロセスは 2 つのローカル証跡に書き込みますが、そのうちの 1 つが暗号化されます。

```
EXTRACT capt
USERID ogg, PASSWORD AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC, &
    AES128 KEYNAME mykey1
DISCARDFILE /ogg/capt.dsc, PURGE
-- Do not encrypt this trail.
EXTTRAIL /ogg/dirdat/bb
TABLE SALES.*;
-- Encrypt this trail with AES-192.
ENCRYPTTRAIL AES192 KEYNAME mykey1
EXTTRAIL /ogg/dirdat/aa
TABLE FIN.*;
```

データ・ポンプ 1 のパラメータ・ファイル

このポンプは、fin が所有している表のデータを含む、暗号化されたローカル証跡 /ogg/dirdat/aa を読み取り、ホスト myhost1 上のリモート証跡 /ogg/dirdat/cc にデータを送信します。

```
EXTRACT 1pump
USERID ogg, PASSWORD AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC, &
    AES128 KEYNAME mykey1
DISCARDFILE /ogg/1pmp.dsc, PURGE
-- Decrypt the trail this pump reads.Use encryption key mykey1.
DECRYPTTRAIL AES192 KEYNAME mykey1
-- Encrypt the trail this pump writes to, using AES-192.
RMTHOST myhost1, MGRPORT 7809
ENCRYPTTRAIL AES192 KEYNAME mykey2
RMTTRAIL /ogg/dirdat/cc
TABLE FIN.*;
```

データ・ポンプ 2 のパラメータ・ファイル

このポンプは、sales が所有している表のデータを含む、暗号化されていないローカル証跡 /ogg/dirdat/bb を読み取り、ホスト myhost2 上のリモート証跡 /ogg/dirdat/dd にデータを送信します。

```
EXTRACT 2pump
USERID ogg, PASSWORD AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC, &
    AES128 KEYNAME mykey1
DISCARDFILE /ogg/2pmp.dsc, PURGE
RMTHOST myhost2, MGRPORT 7809
RMTTRAIL /ogg/dirdat/dd
TABLE SALES.*;
```

Replicat1(myhost1 上) のパラメータ・ファイル

この Replicat は、fin が所有している表の暗号化されたデータを含む、暗号化された証跡 /ogg/dirdat/cc を読み取り、myhost1 上のターゲット・データベースに適用します。

```
REPLICAT 1deliv
USERID ogg, PASSWORD AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC, &
    AES128 KEYNAME mykey2
ASSUMETARGETDEFS
DISCARDFILE /ogg/1deliv.dsc, PURGE
-- Decrypt the trail this Replicat reads.Use encryption key mykey2.
DECRYPTTRAIL AES192 KEYNAME mykey2
MAP FIN.*, TARGET FIN.*;
```

Replicat 2(myhost2 上) のパラメータ・ファイル

この Replicat は、sales が所有している表のデータを含む、暗号化されていない証跡 /ogg/dirdat/dd を読み取り、myhost2 上のターゲット・データベースに適用します。

```
REPLICAT 2deliv
USERID ogg, PASSWORD AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC, &
    AES128 KEYNAME mykey2
ASSUMETARGETDEFS
DISCARDFILE /ogg/2deliv.dsc, PURGE
MAP SALES.*, TARGET SALES.*;
```

データベース・ユーザー・パスワードの暗号化

この項では、Oracle GoldenGate プロセスでデータベース管理システムへのログインに使用されるデータベース・パスワードの暗号化の方法を示します。暗号化パスワードを使用するには、次の 2 つの手順を実行します。

- パスワードの暗号化
- パラメータまたはコマンドへの暗号化パスワードの指定

パスワードを暗号化する手順

1. GGSCI を実行します。
2. ENCRYPT PASSWORD コマンドを発行します。

```
ENCRYPT PASSWORD <password> <algorithm> ENCRYPTKEY {<keyname> | DEFAULT}
```

条件 :

- <password> は平文のログイン・パスワードです。パスワードは引用符で囲まないでください。パスワードに大 / 小文字の区別がある場合は、そのように入力してください。
- <algorithm> では、使用する暗号化アルゴリズムを指定します。
  - AES128 の場合、AES-128 暗号が使用されます (鍵サイズは 128 ビットです)。
  - AES192 の場合、AES-192 暗号が使用されます (鍵サイズは 192 ビットです)。
  - AES256 の場合、AES-256 暗号が使用されます (鍵サイズは 256 ビットです)。
  - BLOWFISH の場合、Blowfish 暗号化が使用されます (ブロック・サイズは 64 ビット、鍵サイズは 32 ～ 128 ビットの可変長です)。BLOWFISH は、旧バージョンの Oracle GoldenGate との下位互換性を確保する目的でのみ使用してください。
- ENCRYPTKEY <keyname> では、ENCKEYS 参照ファイルにある、ユーザーが作成した暗号化鍵の論理参照名を指定します。鍵の名前は、ENCKEYS ファイルにある実際の鍵を参照するために使用されます。AES 暗号化を行うには、ユーザー定義の鍵と ENCKEYS ファイルを使用する必要があります。鍵と ENCKEYS ファイルを作成する方法は、134 ページの「暗号化鍵の生成」を参照してください。
- ENCRYPTKEY DEFAULT を指定すると、Oracle GoldenGate でランダムな鍵が生成され、証跡に格納されるので、後続のプロセスで復号化を実行できます。このタイプの鍵は安全ではないため、本番環境では使用しないでください。このオプションは、BLOWFISH を指定した場合にのみ使用します。AES を DEFAULT とともに使用すると、ENCRYPT PASSWORD はエラーを戻します。

アルゴリズムを指定しない場合、AES128 がすべてのデータベース・タイプのデフォルトになります (ただし、z/OS 上の DB2 と NonStop SQL/MX は BLOWFISH がデフォルトです)。

3. 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。

ENCRYPT PASSWORD の例

ENCRYPT PASSWORD mypassword AES256 ENCRYPTKEY mykey1

ENCRYPT PASSWORD mypassword BLOWFISH ENCRYPTKEY mykey1

ENCRYPT PASSWORD mypassword BLOWFISH ENCRYPTKEY DEFAULT

パラメータまたはコマンドに暗号化パスワードを指定する手順

1. ENCRYPT PASSWORD コマンド (130 ページ) で生成した暗号化パスワードをコピーし、表 11 に示すように Oracle GoldenGate の適切なパラメータ文またはコマンドに貼り付けます。オプションについては表の後で説明します。

表 11　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[1]	`USERID <user>, PASSWORD <password>, &` `<algorithm> ENCRYPTKEY {<keyname> | DEFAULT}`

Oracle ASM インスタンスの Oracle GoldenGate ユーザー	`TRANLOGOPTIONS ASMUSER SYS@<ASM_instance_name>, &`
	`ASMPASSWORD <password>, &`
	`<algorithm> ENCRYPTKEY {<keyname> | DEFAULT}`

表 11　Oracle GoldenGate パラメータ・ファイルでの暗号化パスワードの指定 (続き)

パスワード暗号化の対象	使用するパラメータ
{CREATE | ALTER} USER <name> IDENTIFIED BY <password> のパスワード置換	`DDLOPTIONS DEFAULTUSERPASSWORD <password> <algorithm> ENCRYPTKEY {<keyname> | DEFAULT}`
Oracle TDE 共有秘密鍵のパスワード	`DBOPTIONS DECRYPTPASSWORD <password>`[2] `<algorithm> ENCRYPTKEY {<keyname> | DEFAULT}`
GGSCI コマンド・インタフェースからの Oracle GoldenGate データベース・ログイン	`DBLOGIN USERID <user>, PASSWORD <password>, <algorithm> ENCRYPTKEY {<keyname> | DEFAULT}`

[1] USERID に必要な構文要素は、データベースのタイプによって異なります。詳細は、Oracle GoldenGate のリファレンス・ドキュメントを参照してください。
[2] これは共有秘密鍵です。

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。
- <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。
- <algorithm> では、パスワードの暗号化に使用した暗号化アルゴリズム (AES128、AES192、AES256、BLOWFISH のいずれか) を指定します。デフォルトの鍵を使用し、アルゴリズムを指定しない場合は、AES128 がデフォルトになります。
- ENCRYPTKEY <keyname> では、ENCKEYS 参照ファイルにある、ユーザーが作成した暗号化鍵の論理参照名を指定します。ENCRYPT PASSWORD を KEYNAME <keyname> オプションとともに使用した場合に使用します。
- ENCRYPTKEY DEFAULT を指定すると、Oracle GoldenGate でランダムな鍵が使用されます。ENCRYPT PASSWORD を KEYNAME DEFAULT オプションとともに使用した場合に使用します。

パラメータおよびコマンドで暗号化パスワードを使用する方法の例

```
SOURCEDB db1 USERID ogg,&
PASSWORD AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC, &
AES128, ENCRYPTKEY securekey1

USERID ogg, PASSWORD AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC, &
BLOWFISH, ENCRYPTKEY securekey1

USERID ogg, PASSWORD AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC, &
BLOWFISH, ENCRYPTKEY DEFAULT

TRANLOGOPTIONS ASMUSER SYS@asm1, &
ASMPASSWORD AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC, &
AES128, ENCRYPTKEY securekey1

DBLOGIN USERID ogg, PASSWORD &
AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC, &
AES128, ENCRYPTKEY securekey1

DDLOPTIONS DEFAULTUSERPASSWORD &
AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC, &
AES 256 ENCRYPTKEY mykey
```

```
DBOPTIONS DECRYPTPASSWORD AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC, &
AES 256 ENCRYPTKEY mykey

DDLOPTIONS PASSWORD AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC, &
AES 256 ENCRYPTKEY mykey
```

TCP/IP を通じて送信されるデータの暗号化

この項では、Extract またはデータ・ポンプが TCP/IP 接続を通じてリモート・ターゲットに送信するデータの暗号化の方法を示します。データは、ターゲットで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)。

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。134 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 は次のような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 (137 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、暗号化アルゴリズムおよび鍵の論理名を次のように指定します。

   ```
   RMTHOST <host>, MGRPORT <port>,
   ENCRYPT <algorithm> KEYNAME <keyname>

   RMTHOSTOPTIONS ENCRYPT <algorithm> KEYNAME <keyname>
   ```

   条件 :

   - <algorithm> では、使用する暗号化アルゴリズムを指定します。
     - AES128 の場合、AES-128 暗号が使用されます (鍵サイズは 128 ビットです)。
     - AES192 の場合、AES-192 暗号が使用されます (鍵サイズは 192 ビットです)。
     - AES256 の場合、AES-256 暗号が使用されます (鍵サイズは 256 ビットです)。
     - BLOWFISH の場合、Blowfish 暗号化が使用されます (ブロック・サイズは 64 ビット、鍵サイズは 32 ～ 128 ビットの可変長です)。BLOWFISH は、旧バージョンの Oracle GoldenGate との下位互換性を確保する目的でのみ使用してください。
   - KEYNAME <keyname> では、ENCKEYS 参照ファイルにある暗号化鍵の論理名を指定します。鍵の名前は、ENCKEYS ファイルにある実際の鍵を参照するために使用されます。

TCP/IP 暗号化を構成する方法の例

```
RMTHOST myhost, MGRPORT 7840, ENCRYPT BLOWFISH, KEYNAME mykey

RMTHOST myhost, MGRPORT 7840, ENCRYPT AES256, KEYNAME mykey

RMTHOSTOPTIONS ENCRYPT AES192, KEYNAME mykey
```

4. 静的 Collector 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 <keyname>
-ENCRYPT <algorithm>
```

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

次のものを使用する場合、暗号化鍵を生成して格納する必要があります。

- KEYNAME <keyname> を指定した ENCRYPTTRAIL(126 ページを参照)
- ENCRYPTKEY <keyname> を指定した ENCRYPT PASSWORD(130 ページを参照)
- ENCRYPT を指定した RMTHOST または RMTHOSTOPTIONS(133 ページを参照)

この手順は、次の暗号化オプションを使用する場合には*不要*です。

- ENCRYPTKEY DEFAULT を指定した ENCRYPT PASSWORD(BLOWFISH を使用する場合にのみ有効)
- オプションなしの ENCRYPTTRAIL(256 鍵バイト置換の場合)

この手順では、次の操作を実行します。

- 1 つ以上の暗号化鍵を作成します。
- ソース・システムの ENCKEYS 参照ファイルに鍵を格納します。
- 各ターゲット・システムに ENCKEYS ファイルをコピーします。

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な鍵を作成できます。

独自の鍵を定義する手順

任意のツールを使用します。鍵の値は、次のいずれかの形式で最大 128 ビット (16 バイト) です。

- 引用符で囲んだ英数字の文字列 ("Dailykey" など)
- 接頭辞 0x を付けた 16 進文字列 (0x420E61BE7002D63560929CCA17A4E1FB など)

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。これらをコピーして ENCKEYS ファイルに貼り付けることができます。

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

条件 :

- <key length> は、最大 128 ビット (16 バイト) 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
```

ENCKEYS 参照ファイル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. 生成した鍵の値ごとに、任意の論理名に続けて鍵の値を入力します。
   - 鍵の名前には、空白または引用符のない 1 ～ 24 個の英数字文字列を指定できます。
   - 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. Oracle GoldenGate のインストール・ディレクトリに、このファイルを*拡張子なし*で、すべて大文字の ENCKEYS という名前で保存します。
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・セキュリティの構成

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 136 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 12　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 13　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 13 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

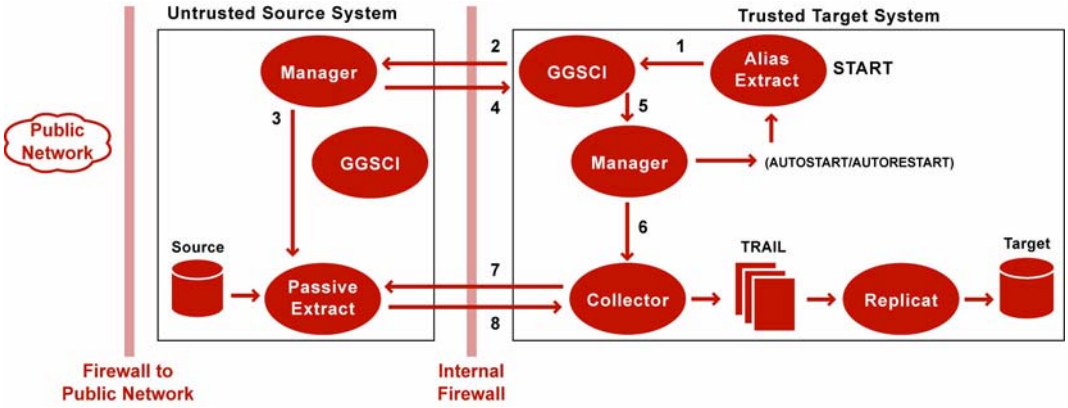
CMDSEC ファイルは、セキュリティの根源であるため、保護する必要があります。読取りアクセス権は必要に応じて付与できますが、書込みおよび削除アクセス権は、Oracle GoldenGate 管理者以外のすべてのユーザーに対して Oracle GoldenGate で拒否することをお薦めします。

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 がパッシブ Extract を起動すると、パッシブ Extract 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、そのポートでリスニングを行います。
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。

次に例を示します。

```
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 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 14 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、171 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。
- 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 11 章

データのマッピングおよび操作

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。

- ソース・オブジェクトからターゲット・オブジェクトへのマッピング
- レコードおよび列の選択
- SQL 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

サポートの制限

データのマッピングおよび操作のサポートには次のような制限があります。

- 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。
- 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。

マッピングおよびデータ統合を制御するパラメータ

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 1 つ以上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。

異なるデータベース間のマッピング

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。このファイルを使用する場合は、SOURCEDEFS パラメータまたは TARGETDEFS パラメータで指定する必要があります。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、173 ページの第 13 章を参照してください。

データのマッピングおよび変換の実行場所の決定

計画中の構成が大量の列マッピングまたはデータ変換を伴う場合は、次のガイドラインに従って、それらの機能をどのプロセスで実行するかを判断してください。

Windows システムと UNIX システムでのマッピングおよび変換

Oracle GoldenGate が Windows ベースのシステムと UNIX ベースのシステムでのみ稼働している場合、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。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、ターゲット・システムまたは中間システムでのマッピングおよび変換を構成できます。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。使用する定義ファイルのタイプと場所の詳細は、173 ページの「レプリケートされたデータとメタデータとの関連付け」を参照してください。

NonStop システムでのマッピングおよび変換

Windows システムまたは UNIX システムから NonStop ターゲットへのデータのマッピングまたは変換を行う場合、マッピングまたは変換はソースの Windows システムまたは UNIX システムで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できません。Extract は、証跡データを NonStop の名前およびターゲット・データ型でフォーマットできます。

データ・マッピング時のグローバリゼーションに関する考慮事項

データベース間でのデータのマッピングおよび変換を計画する際には、グローバリゼーションに関して Oracle GoldenGate でサポートされている処理とサポートされていない処理を考慮してください。キャラクタ・セットおよびロケールがどのように適用され、変換されるかを理解しておくと、正確な結果を得るのに役立ちます。

キャラクタ・セット間の変換

Oracle GoldenGate では、ソースとターゲットのキャラクタ・セットが異なると、オブジェクト名および列データがデータベース間で正しく比較、マップ、操作されるようにするため、キャラクタ・セットの変換が行われます。サポートされるキャラクタ・セットのリストは、286 ページの「サポートされるキャラクタ・セット」を参照してください。

異なるデータベースの間で文字が正確に表現されるためには、次の条件を満たす必要があります。

- ターゲット・データベースのキャラクタ・セットは、ソース・データベースのキャラクタ・セットのスーパーセットであるか、ソース・データベースのキャラクタ・セットと同等である必要があります。*同等*とは、同一であることではなく、同じ文字のセットが存在することを意味します。たとえば、Shift-JIS と EUC-JP は、厳密に言うと完全に同一ではありませんが、ほとんどの場合に同じ文字が存在します。
- クライアント・アプリケーションで異なるキャラクタ・セットが使用されている場合、データベースのキャラクタ・セットがクライアント・アプリケーションのキャラクタ・セットのスーパーセットであるか、クライアント・アプリケーションのキャラクタ・セットと同等であることも必要です。

この構成では、クライアントまたはソースのキャラクタ・セットからローカル・データベースのキャラクタ・セットへの変換時に、すべての文字が表現されます。

データベース・オブジェクト名

Oracle GoldenGate では、カタログ、スキーマ、表および列の名前は、ソース・データベースおよびターゲット・データベースのキャラクタ・セット・エンコーディングに従って、ネイティブ言語で処理されます。このサポートにより、データベース階層のあらゆるレベルで、シングルバイトおよびマルチバイトの名前、記号、アクセント文字、大 / 小文字の区別が維持され、ロケールがあればそれも考慮されます。

列データ

Oracle GoldenGate では、データが次のタイプの列に格納されている場合、キャラクタ・セット間での列データの変換がサポートされます。

- 文字型の列 : CHAR/VARCHAR/CLOB から別のキャラクタ・セットの CHAR/VARCHAR/CLOB への変換、および CHAR/VARCHAR/CLOB と NCHAR/NVARCHAR/NCLOB との間の変換。
- 文字列ベースの数値および日時データを含む列。z/OS の EBCDIC データと z/OS 以外の ASCII データの間では、これらの列の変換が実行されます。このデータの ASCII 版と ASCII 版の間、および EBCDIC 版と EBCDIC 版の間では、データに互換性があるため、変換は実行されません。

列データのキャラクタ・セット変換は、Replicat の MAP パラメータの COLMAP 句または USEDEFAULTS 句のソース列とターゲット列との直接マッピングに限定されています。直接マッピングは、ストアド・プロシージャや列変換関数を使用しない、名前から名前へのマッピングです。Extract やデータ・ポンプでは変換は実行されません。

Replicat は、Oracle 以外のすべてのターゲット・データベース・タイプに対して変換を実行します。Oracle では常に独自の変換が実行されますが、ソースが z/OS 上の DB2 で、かつ証跡が EBCDIC で記述されている場合は例外です。その場合、Oracle では EBCDIC エンコーディングからの変換は行われないため、Replicat が変換を実行します。

> **注意** Oracle ターゲットで変換を実行するには、Replicat パラメータ・ファイルに SETENV パラメータを含めて、NLS_LANG 環境変数を *ソース・データベース*のキャラクタ・セットに設定する必要があります。詳細は、『*Oracle GoldenGate Oracle インストレーション・ガイド*』を参照してください。

Replicat パラメータ CHARSETCONVERSION および NOCHARSETCONVERSION では、Replicat でキャラクタ・セットの変換を行うかどうかを制御します。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。

証跡が古い形式で (通常は旧バージョンの Extract によって) 記述されている場合、文字型の列のキャラクタ・セットを Replicat に送信できない可能性があります。この場合、デフォルトでは Replicat は変換を実行しません (EBCDIC と ASCII の間の変換を除く)。CHAR、VARCHAR2、CLOB の各列にはホスト・オペレーティング・システムのキャラクタ・セットが使用され、NCHAR、NVARCHAR2、NCLOB の各列には UTF-16 ビッグ・エンディアンが使用されます。正確な変換を可能にするため、Oracle サポートに相談したうえで、内部パラメータ _TRAILCHARSET を使用してください。

ロケールの維持

Oracle GoldenGate では、大 / 小文字が区別されないオブジェクト名の比較時に、データベースのロケールが考慮されます。サポートされるロケールのリストは、294 ページの「サポートされるロケール」を参照してください。

特定の文字に対するエスケープ・シーケンスの使用

Oracle GoldenGate では、パラメータ・ファイルで文字の列、リテラル・テキストまたはオブジェクト名を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、オペレーティング・システムで必要な文字 (制御文字など) がサポートされていない場合に、またはパラメータ・ファイルで使用できない文字が必要になるその他の目的のために使用できます。

エスケープ・シーケンスは、パラメータ・ファイルの任意の場所で使用できますが、TABLE 文または MAP 文の次の要素で特に便利です。

- オブジェクト名
- 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。サロゲート・ペアを除き、任意の UNICODE コード・ポイントを使用できます。
- \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

次のルールが適用されます。

- TABLE または MAP でオブジェクト名のマッピングに使用する場合、制限は何もありません。たとえば、次の TABLE 指定は有効です。

  ```
  TABLE schema."\u3000ABC";
  ```

- 列マッピング関数とともに使用する場合、任意のコード・ポイントを使用できますが、対象となるのは NCHAR/NVARCHAR 列のみです。CHAR/VARCHAR 列の場合、コード・ポイントは 7 ビット ASCII と同等のものに限定されます。

- 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。

注意 **パラメータ・ファイルで実際のバックスラッシュを指定するには、二重のバックスラッシュを指定します。たとえば、@STRFIND(COL1, "\\") と指定すると、COL1 内でバックスラッシュが検索されます。**

\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 ～ 9(U+0030 ～ U+0039)
  - A ～ F(U+0041 ～ U+0046)
  - a ～ f(U+0061 ～ U+0066)

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 ～ 3(U+0030 ～ U+0033)
  - 2 桁目と 3 桁目の範囲は 0 ～ 7(U+0030 ～ 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 ～ 9(U+0030 ～ U+0039)
  - A ～ F(U+0041 ～ U+0046)
  - a ～ f(U+0061 ～ U+0066)

例 \x80 は、Microsoft Windows 1252 Latin1 コード・ページのユーロ通貨記号を表す 16 進エスケープ・シーケンスです。

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。

列名での大 / 小文字の区別と特殊文字のサポート

Oracle GoldenGate ではデフォルトで、二重引用符で囲んだ文字列がリテラルとして扱われます。大 / 小文字が区別される列名や特殊文字を含む列名をサポートするには、USEANSISQLQUOTES パラメータを使用します。USEANSISQLQUOTES を使用すると、Oracle GoldenGate で引用符を使用して識別子およびリテラル文字列を区切る際に、SQL-92 ルールに従うことができます。USEANSISQLQUOTES が有効になっている場合、Oracle GoldenGate では、二重引用符で囲んだ文字列は列名として扱われ、一重引用符で囲んだ文字列はリテラルとして扱われます。このサポートは、Oracle GoldenGate インスタンスのすべてのプロセスに対してグローバルに適用されます。使用方法と制限の詳細は、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』の USEANSISQLQUOTES の説明を参照してください。

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。

ソースとターゲット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。

- データベース・タイプが同じであること (すべて Oracle であるなど)。
- キャラクタ・セットおよびロケールが同じであること。
- 同じ数の列が含まれていること。
- 列名が同一であること (該当する場合は大 / 小文字、空白、引用符も含む)。
- 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。
- すべての列が同じ順序で定義されていること。

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ターゲット・システム上の Replicat の定義ファイルを指定するか、TARGETDEFS パラメータを使用してソース・システムまたは中間システム上の Extract またはデータ・ポンプの定義ファイルを指定します。

173 ページの「レプリケートされたデータとメタデータとの関連付け」を参照してください。

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 14　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。列名での大 / 小文字の区別または特殊文字のサポートの詳細は、145 ページの「列名での大 / 小文字の区別と特殊文字のサポート」を参照してください。
<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) ◆ 二重引用符で囲まれた文字列定数 ("ABCD" など)。GLOBALS ファイルで USEANSISQLQUOTES パラメータが使用されている場合、一重引用符 ('ABCD' など) を使用できます。 ◆ ソース列の名前 (ORD_DATE など)。大 / 小文字が区別される列名や特殊文字を含む列名に二重引用符を使用するには ("Ord Date" など)、GLOBALS ファイルで USEANSISQLQUOTES パラメータを使用します。 ◆ 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) ソースとターゲットのキャラクタ・セットが異なる場合、いかなる変換も行わない、名前から名前の直接マッピングのみが、サポートされる列マッピングです。列名での大 / 小文字の区別または特殊文字のサポートの詳細は、145 ページの「列名での大 / 小文字の区別と特殊文字のサポート」を参照してください。
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 ユーティリティ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、151 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 15　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。

大 / 小文字の区別は次のようにサポートされます。

- Sybase、MySQL、SQL Server、PostgreSQL および Teradata の場合、データベースの大 / 小文字が区別されるときは、名前が引用符で囲まれているかどうかにかかわらず、大 / 小文字と名前が完全に一致するものが検索されます。

- 同じデータベース内で名前の大 / 小文字を区別することも区別しないことも可能な Oracle、DB2 および SQL/MX の場合、名前が引用符で囲まれているときは、大 / 小文字と名前が完全に一致することが求められます。

Oracle、DB2 および SQL/MX でのサポートについて、次に詳しく説明します。

COLMATCH と NAMES の組合せ

NAMES オプションを使用する場合、引用符で囲んだ列名は大 / 小文字を区別するものとして扱われ、引用符で囲んでいない列名は大 / 小文字を区別しないものとして扱われます。次の例では、大 / 小文字が区別されないソース列名 abc と、大 / 小文字が区別されるターゲット列名 "abc" を指定しています。

```
COLMATCH NAMES "abc" = abc
```

次の例では、大 / 小文字が区別されないソース列名 abc と、大 / 小文字が区別されないターゲット列名 xyz を指定しています。

```
COLMATCH NAMES xyz = abc
```

COLMATCH と PREFIX または SUFFIX の組合せ

二重引用符で囲んだ、または囲まない PREFIX および SUFFIX を指定して、COLMATCH に無視させることができます。引用符で囲んだ接頭辞または接尾辞は大 / 小文字を区別するものとして扱われ、引用符で囲んでいない接頭辞または接尾辞は大 / 小文字を区別しないものとして扱われます。接頭辞または接尾辞の大 / 小文字の区別が影響するのは接頭辞または接尾辞のみで、列指定の他の部分は影響を受けません。

次の例では、大 / 小文字が区別されない接頭辞 p_ を無視するよう指定しています。ターゲット列名 P_ABC はソース列名 ABC にマップされ、ターゲット列名 "P_abc" はソース列名 "abc" にマップされます。

```
COLMATCH PREFIX p_
```

次の例では、大 / 小文字が区別される接尾辞 _k を無視するよう指定しています。ターゲット列名 ABC_k はソース列名 ABC にマップされ、ターゲット列名 "abc_k" はソース列名 "abc" にマップされます。

```
SUFFIX "_k"
```

構文

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

表 16　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。
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。

表 16　COLMATCH のオプション (続き)

引数	説明
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。データベースが大 / 小文字が区別されません。

表 17　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 SALE.ORD, TARGET SALES.ORDER, COLMAP (USEDEFAULTS);
COLMATCH RESET
MAP SALES.REG, TARGET SALE.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。

- 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、名前のフォールバック・マッピングが使用されます。フォールバック・マッピングでは、大 / 小文字を区別せずにターゲット表のマッピングを実行して、一致する名前を探します。大文字の名前を使用して、厳密でない列名マッピングが適用されます。この動作は、GLOBALS のパラメータ NAMEMATCHIGNORECASE によって制御します。名前のフォールバック照合は、NAMEMATCHEXACT パラメータを使用して無効にすることも、NAMEMATCHNOWARNING パラメータを使用して、有効にしたまま警告メッセージを表示することもできます。
- 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 18 に示されているいずれかの値になります。

表 18　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

列間での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。

文字型の列

文字型の列では、VARCHAR などの文字ベースのデータ型、文字列形式の数値、文字列形式の日時、および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。

デフォルトでは、リテラルは二重引用符で囲む必要があります。SQL-92 ルールを使用して列の識別子およびリテラルを区切る方法の詳細は、145 ページの「列名での大 / 小文字の区別と特殊文字のサポート」を参照してください。

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。

デフォルトでは、リテラルは二重引用符で囲む必要があります。SQL-92 ルールを使用して列の識別子およびリテラルを区切る方法の詳細は、145 ページの「列名での大 / 小文字の区別と特殊文字のサポート」を参照してください。

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。

行の選択

抽出またはレプリケートする行をフィルタで除外または選択するには、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、159 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。

デフォルトでは、FILTER 内のリテラル文字列は二重引用符で囲む必要があります。大 / 小文字が区別される列名や特殊文字を含む列名に二重引用符を使用する場合、およびリテラル文字列に一重引用符を使用する場合は、GLOBALS ファイルで USEANSISQLQUOTES 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。

Oracle GoldenGate では、マルチバイトのキャラクタ・セット、またはローカル・オペレーティング・システムのキャラクタ・セットと互換性のないキャラクタ・セットが使用されている列に対する FILTER はサポートされません。

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、@STREQ 関数を使用して、文字列が 'JOE' と同一であるレコードを抽出します。この例では、GLOBALS パラメータ・ファイルで USEANSISQLQUOTES パラメータを使用して、一重引用符および二重引用符に関する SQL-92 ルールを適用すると仮定しています。

```
TABLE ACCT.TCUSTORD, FILTER (@STREQ ("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。この場合、オブジェクト名は大 / 小文字が区別されます。

(Replicat グループ 1 のパラメータ・ファイル)

```
MAP "sales"."acct", TARGET "sales"."acct", FILTER (@RANGE (1, 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 19 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 19 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. 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 19　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 20 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 20　使用可能な WHERE 演算子

要素	例
列名	`PRODUCT_AMT`
数値	`-123, 5500.123`
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。156 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。

デフォルトでは、WHERE 内のリテラル文字列は二重引用符で囲む必要があります。大 / 小文字が区別される列名や特殊文字を含む列名に二重引用符を使用する場合、およびリテラル文字列に一重引用符を使用する場合は、GLOBALS ファイルで USEANSISQLQUOTES 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。

Oracle GoldenGate では、マルチバイトのキャラクタ・セット、またはローカル・オペレーティング・システムのキャラクタ・セットと互換性のないキャラクタ・セットが使用されている列に対する FILTER はサポートされません。

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 (238 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。

注意 この項の例では大 / 小文字が区別されないデータベースを想定しているため、パラメータで使用されている文字が大文字か小文字かは問題ではありません。

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。
- それらの列に対するサプリメンタル・ロギングを有効にして、必要な列値を使用できるようにします。あるいは、TABLE パラメータの FETCHCOLS オプションまたは FETCHCOLSEXCEPT オプションを使用することもできます。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.PEOPLE, FETCHBEFOREFILTER, FETCHCOLS (AGE), FILTER (AGE > 50);
  ```

- 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、該当する列タイプを比較します。

- 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)。

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)
```

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) を取得すると役立つことがあります。これらの値は証跡に格納されており、フィルタおよび列マッピングで使用できます。たとえば、次のことが可能です。

- 例外 MAP 文における列マッピング指定の一部として、行の変更前イメージを取得し、競合解決ルーチンのテストまたはトラブルシューティング用に、それらの値を例外表にマップします。
- デルタ計算を実行します。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);
```

注意 前の例は、Oracle などの大 / 小文字が区別されるデータベースを示しているため、表名を引用符で囲みます。大 / 小文字が区別される列名に引用符を使用するには、USEANSISQLQUOTES パラメータを使用する必要があります。

変更前の値を参照する手順

1. 次のように、BEFORE キーワード、ドット (.)、変更前の値を必要とする列の名前という順で使用します。

   ```
   BEFORE.<column_name>
   ```

2. GETUPDATEBEFORES パラメータを Extract のパラメータ・ファイルで使用して、トランザクション・レコードから変更前イメージを取得するか、Replicat のパラメータ・ファイルで使用して、列マッピングまたはフィルタで変更前イメージを使用します。競合の検出および解決 (CDR) 機能を使用する場合は、TABLE の GETBEFORECOLS オプションを使用できます。これら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、156 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。

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

デフォルトでは、Oracle GoldenGate によって挿入操作、更新操作、削除操作が取得および適用されます。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

Replicat パラメータ・ファイルで次のパラメータを使用すると、あるタイプの SQL 操作を別のタイプに変換できます。

- 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。

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 17 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

158 ページの図 17 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。これらの例が、大 / 小文字が区別されないデータベースを表していることにも注意してください。

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、238 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)。関数名と入力引数の前にある開き丸カッコの間の空白は、オプションです。
`<argument>`	関数の引数として、次の要素を指定できます。 **引数の要素** / **例** 数値定数 / `123` 文字列リテラル / `"ABCD"` または `'ABCD'` (GLOBALS で USEANSISQLQUOTES が使用されているかどうかによります。) ソース列の名前 / `PHONE_NO` または `"Phone_No"` (GLOBALS で USEANSISQLQUOTES が使用されているかどうかによります。) 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)`

関数での列名およびリテラルの処理

デフォルトでは、列変換関数内のリテラル文字列は二重引用符で囲む必要があります。大 / 小文字が区別される列名や特殊文字を含む列名に二重引用符を使用する場合、およびリテラル文字列に一重引用符を使用する場合は、GLOBALS ファイルで USEANSISQLQUOTES パラメータを使用します。このパラメータにより、リテラルおよび識別子に SQL-92 ルールが適用されます。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。

適切な関数の使用

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。LOB 列は、変換関数では使用できません。

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")
```

162 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)
```

注意 **デフォルトでは、この例に示したハードコードされた日付は二重引用符で囲まれているため、リテラルとして扱われます。USEANSISQLQUOTES パラメータを使用する場合は、リテラル日付を一重引用符で囲む必要があります。**

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 21　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

表 21　数値および文字のための変換関数 (続き)

用途	変換関数
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)

> **注意** デフォルトでは、この例に示した文字列は二重引用符で囲まれているため、リテラルとして扱われます。USEANSISQLQUOTES パラメータを使用する場合は、文字列を一重引用符で囲む必要があります。

@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、Oracle GoldenGate 列変換関数から取得された任意の有効な文字データまたは値です。

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 (USEANSISQLQUOTES を使用する場合は一重引用符) で囲んだ文字列か、Oracle GoldenGate 列変換関数の結果です (次の例を参照)。トークン・データのキャラクタ・セットは変換されません。トークンは、Extract の場合はソース・データベースのキャラクタ・セット、Replicat の場合はターゲット・データベースのキャラクタ・セットである必要があります。

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。引数は二重引用符で囲まれていますが、USEANSISQLQUOTES パラメータを使用する場合は一重引用符で囲む必要があります。USEANSISQLQUOTES および @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。引数は二重引用符で囲まれていますが、USEANSISQLQUOTES パラメータを使用する場合は一重引用符で囲む必要があります。USEANSISQLQUOTES 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。

```
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
```

第 12 章

Oracle GoldenGate の処理エラーへの対処

この章には、処理エラーへの対処方法に関する情報が記載されています。Oracle GoldenGate では、処理エラーは監視ツールおよびレポート・ツールによって複数の方法でレポートされます。これらのツールの詳細は、249 ページの第 17 章を参照してください。

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

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、『Oracle GoldenGate Oracle インストレーションおよびセットアップ・ガイド』の「DDL 処理エラーの処理」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。

REPERROR による各種レスポンスは次のオプションで構成されます。

- 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。

ほとんどのオプションは、エラーを生成した個々のレコードに対して機能します。エラーを生成していない、トランザクションのその他の操作は、Replicat によって処理されます。ただし、TRANSDISCARD と TRANSEXCEPTION は例外で、これらのオプション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。)

例外としてのエラーの処理

REPERROR のアクションが EXCEPTION または TRANSEXCEPTION である場合、エラーを生成する操作の値を例外表にマップしたり、オプションでエラーの解決に利用可能なその他のエラー情報をマップしたりすることができます。「例外表について」を参照してください。

例外を例外表にマップするには、MAP パラメータの次のオプションを使用します。

- MAP での EXCEPTIONSONLY
- MAP での MAPEXCEPTION

EXCEPTIONSONLY の使用

EXCEPTIONSONLY は、ソース表とターゲット表の 1 つのペアが MAP 文で明示的に指定され、1 対 1 でマップされている場合 (つまり、ワイルドカードが使用されていない場合) に有効です。EXCEPTIONSONLY を使用するには、ターゲットで EXCEPTIONSONLY を使用するソース表ごとに、MAP 文を 2 つ作成します。

- 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するとただちに実行され、行の値を例外表に送信します。MAP 文を例外 MAP 文として指定するには、INSERTALLRECORDS オプションと EXCEPTIONSONLY オプションを使用します。例外 MAP 文は、同じ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。ソース表と例外表の列が同一でない場合、または例外表の追加の列に追加情報 (列変換関数または SQLEXEC で取得した情報など) をマップする場合は、例外 MAP 文で COLMAP 句を使用します。

MAPEXCEPTION の使用

MAPEXCEPTION は、MAP 文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ている場合に有効です。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) に配置します。Replicat は、エラーを生成するすべての操作を、ワイルドカード指定されたすべての表から同じ例外表にマップします。したがって、例外表には、ワイルドカード指定されたすべての表におけるすべての列のスーパーセットが含まれている必要があります。また、ワイルドカード構成では列を個別にマップできないため、COLMAP 句を USEDEFAULTS オプションとともに使用してワイルドカード指定された表の列マッピングを処理し (または COLMATCH パラメータを適宜使用し)、明示的な列マッピングを使用して追加情報 (列変換関数または SQLEXEC で取得した情報など) をマップします。

例外表について

エラーに関する情報を取得して、アプリケーションのトラブルシューティングやエラー処理のためのアプリケーション構成などに役立てるには、例外表を使用します。例外表には少なくとも、失敗した操作から行イメージ全体を受信できるだけの列が含まれている必要があります。追加の列を定義して、列変換関数、SQLEXEC、その他の外部手段によって取得したその他の情報を含めることができます。

例外表にマップしたすべての列の値を証跡レコードに確実に含めるには、Extract パラメータ・ファイルで次のパラメータを使用します。

- NOCOMPRESSDELETES パラメータを使用することで、行のすべての列が DELETE 操作の証跡に書き込まれるようにします。
- GETUPDATEBEFORES パラメータを使用することで、Extract によって行の変更前イメージが取得され、証跡に書き込まれるようにします。

例 1 EXCEPTIONSONLY

この例では、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。この例は、REPERROR に関連するパラメータのみを示していますが、Replicat には、エラー処理に関連しない他のパラメータも必要です。

```
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。このシナリオでは一意性違反が発生する可能性があり、発生した場合はエラーが生成されます。

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、『Oracle GoldenGate Oracle インストレーションおよびセットアップ・ガイド』の「DDL 処理エラーの処理」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 18 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 22 に示されている列の値を変更します。

表 22 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 *エラー・メッセージ・ガイド*』を参照してください。

第 13 章

レプリケートされたデータとメタデータとの関連付け

ある表から別の表にデータをレプリケートする際に考慮すべき重要な点の 1 つは、ソース表とターゲット表の列構造 (メタデータ) が同一であるかどうかです。Oracle GoldenGate は、次の目的でメタデータを参照します。

- ソースでは、取得された操作に関する完全な情報を Replicat プロセスに提供するため。
- ターゲットでは、レプリケートされたデータが Replicat によって正しくマップされ、必要に応じて変換されるように、ターゲット表の構造を判別するため。

次に示すシナリオでは、それぞれ異なるパラメータまたはパラメータ・セットを使用して、メタデータを処理する Oracle GoldenGate プロセスに対してメタデータを正しく表現する必要があります。

- ソース表を、同一のメタデータ定義を持つターゲット表にレプリケートする場合 (同機種レプリケーション)。
- ソース表を、異なるメタデータ定義を持つターゲット表にレプリケートする場合。
- ソース表を 2 つのターゲット表 (同一の定義を持つターゲット表と異なる定義を持つターゲット表) にレプリケートする場合。

同一のメタデータを想定するための Oracle GoldenGate の構成

ソース表とターゲット表が同一のメタデータ定義を持つ場合、Replicat のパラメータ・ファイルで ASSUMETARGETDEFS パラメータを使用します。このパラメータは、Replicat に対して、ターゲットの定義がソースの定義と同じであると想定し、SQL 文の作成時にその定義をレプリケートされたデータに適用するよう指示します。ソース表とターゲット表はあらゆる点で同一である必要があるため、変換処理は不要です (ただし、表の所有者または名前、あるいはその両方が異なっていてもかまいません)。

表が同一とみなされるためのルール

ソースとターゲット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。

- データベース・タイプが同じであること (すべて Oracle であるなど)。
- キャラクタ・セットおよびロケールが同じであること (american_AMERICA など)。
- 同じ数の列が含まれていること。
- 列名が同一であること (該当する場合は大 / 小文字、空白、引用符も含む)。
- 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。
- すべての列が同じ順序で定義されていること。

次の単純な Replicat パラメータ・ファイルは、ASSUMETARGETDEFS の使用方法を示しています。詳細は、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』の ASSUMETARGETDEFS の説明を参照してください。

```
-- Specifies the group name.
REPLICAT acctrep
-- Specifies database login with encrypted password.
USERID ogg, PASSWORD AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC, &
    AES128 KEYNAME mykey1
-- Specifies a file for discard output.
DISCARDFILE ./dirrpt/backup/r_prod.dsc, APPEND
-- States that source and target definitions are identical.
ASSUMETARGETDEFS
-- Maps source tables to target tables.
MAP hq.product, TARGET region1.product;
MAP hq.price, TARGET region1.price;
```

異なるメタデータを想定するための Oracle GoldenGate の構成

「表が同一とみなされるためのルール」のルールを満たさない場合、ソースとターゲットのメタデータ定義は同一であるとみなされません。ソース表とターゲット表の定義が異なる場合、Oracle GoldenGate では、ある形式から別の形式への変換を実行する必要があります。変換を実行するには、Oracle GoldenGate が両方の定義セットを認識していることが必要です。Oracle GoldenGate は、ローカル・データベースに問い合せて一方の定義セットを取得できますが、リモート・データベースから定義を取得するには*データ定義ファイル*が必要になります。データ定義ファイルには、レプリケート対象データのメタデータに関する情報が含まれています。次の 2 種類の定義ファイルがあります。

- *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。
- *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。

パラメータ・ファイルでは複数のデータ定義ファイルを使用できます。たとえば、それぞれに異なるアプリケーションの定義を含めることができます。

定義ファイルの内容

データ定義ファイルの形式は内部使用を目的としているため、手順に記載されている場合、またはサポート担当者から指示された場合を除き、Oracle GoldenGate ユーザーが編集しないでください。ファイルの先頭に、DEFGEN のバージョン、キャラクタ・セットの情報、データベース・タイプ、ロケール、およびその他のデータ・プロパティを示す内部メタデータからなるファイル・ヘッダーがあります。ヘッダーの次にあるのが表定義セクションです。各表定義セクションには、表名、レコード長、列数、および 1 つ以上の列定義が含まれています。

使用する定義ファイルのタイプと場所

使用する定義ファイルのタイプは、列のマッピングおよび変換がどこで実行されるかによって異なります。

- WindowsまたはUNIXベースの任意のタイプのデータベース・システムからWindowsまたはUNIXベースの他のシステムにレプリケートする場合、マッピングと変換は Extract、データ・ポンプ Extract、Replicat のいずれでも実行できますが、通常はターゲット・システムの Replicat によって実行されます。ただし、Oracle GoldenGate で異なるキャラクタ・セット間の変換を行う必要がある場合、ターゲットの Replicat によってマッピングと変換が行われる必要があります。「定義ファイルのキャラクタ・セットの影響の理解」を参照してください。
- Windows、UNIX または Linux ベースの任意のデータベース・システムから NonStop Server ターゲットにレプリケートする場合、マッピングと変換は Windows、UNIX、Linux のいずれかのシステムで実行する必要があります。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るのは Extract のみです。このシナリオでは、Oracle GoldenGate はソースとターゲットの間でキャラクタ・セットを変換できません。「定義ファイルのキャラクタ・セットの影響の理解」を参照してください。

したがって、次のようになります。

- 列のマッピングおよび変換をターゲットで実行するには、ソースで生成された*ソース定義ファイル*を使用して、Replicat にソース定義を提供します。
- 列のマッピングおよび変換をソースで実行するには、ターゲットで生成された*ターゲット定義ファイル*を使用し、プライマリ Extract とデータ・ポンプ Extract のどちらが変換を実行するかに応じて、該当するプロセスにターゲット定義を提供します。
- 中間システムで列マッピングとトランスフォーメーションを行うには、複数の定義ファイル・タイプを使用する必要がある場合があります。51 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」および 56 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。中間システムに Replicat がない場合、キャラクタ・セット間の変換は行われないことに注意してください。

定義ファイルのキャラクタ・セットの影響の理解

Oracle GoldenGate では、データの変換を行う場合は、データベースのキャラクタ・セット・エンコーディングを考慮に入れ、定義ファイルを作成する場合は、ローカル・オペレーティング・システムのキャラクタ・セットを考慮に入れます。ソース・データとターゲット・データのキャラクタ・セットが異なる場合、次のガイドラインを考慮に入れます。

データのマッピングと変換を Replicat プロセスに限定

Replicat は、異なるキャラクタ・セット間でレプリケートされたデータの変換を行う唯一のプロセスです。ソース・データベースのキャラクタ・セットからターゲット・データベースのキャラクタ・セット (カスケード構成では中間システムのデータベースのキャラクタ・セット) へデータを変換します。そのため、ソースとターゲットのキャラクタ・セットが異なる場合、データ・マッピングと変換は Replicat によって行われる必要があります。データ・ポンプのみが含まれるソース・システムや中間システムで行うことはできません。このような場合、ターゲット定義ファイルは無効です。

オペレーティング・システムのキャラクタ・セットが原因のファイルの破損の回避

デフォルトでは、定義ファイルは DEFGEN によってローカル・オペレーティング・システムのキャラクタ・セットで記述されます。次の条件を満たす場合、エンコーディング関連の問題が生じることなく、定義ファイルをローカル・システムで作成し、リモート・システムに転送できます。

- 定義ファイルの転送先のリモート・システムに、ローカル・システムと同じか同等のキャラクタ・セットがあります。

- リモート・システムのオペレーティング・システム・キャラクタ・セットは、ローカル・システムのオペレーティング・システム・キャラクタ・セットのサブセットです。たとえば、ソースとターゲットのキャラクタ・セットがいずれも ASCII 互換または EBCDIC 互換で、表名と列名がすべて 7 ビットの US-ASCII またはそれと同等の文字を使用している場合、ソースとターゲットのシステム間で定義ファイルを移動できます。

オペレーティング・システム・キャラクタ・セット同士の多くは、あまり互換性がないか、まったく互換性がありません。リモート・システムで使用されるものと同じか、互換性のあるキャラクタ・セットで定義ファイルを記述するには、DEFGEN を構成する際、DEFSFILE パラメータの CHARSET オプションを使用します。

既存の定義ファイルのキャラクタ・セットの変更

既存の定義ファイルを、キャラクタ・セットに互換性のないオペレーティング・システムへ転送する場合、そのシステムで DEFGEN ユーティリティを実行して、ファイルのキャラクタ・セットを目的のものに変換できます。この手順では、2 つの入力引数 (定義ファイルの名前と UPDATECS <character set> パラメータ) を指定します。次に例を示します。

```
defgen paramfile ./dirdef/source.def UPDATECS UTF-8
```

UPDATECS が役立つのは、日本語の Windows における日本語の表名が Windows CP932 でデータ定義ファイルに記述されており、その定義ファイルを日本語の UNIX に転送するような場合です。UNIX が PCK ロケールで構成されていないかぎり、このファイルは使用できません。そのため、UPDATECS を使用して定義ファイルのエンコーディングを正しい形式に変換する必要があります。

定義テンプレートの使用

定義ファイルを作成する際に定義テンプレートを指定すると、初期起動の後、Oracle GoldenGate 構成に表が追加された場合に、新しい定義ファイルを作成する必要性が軽減されます。テンプレートを使用するには、たとえば顧客データベースに顧客ごとの別々の表が存在するが、すべての表が同一であるというように、新しい表がすべて同一の構造を持っている必要があります (「表が同一とみなされるためのルール」を参照)。

テンプレートを使用しない場合に、起動後、新しい表が追加されたときは、Oracle GoldenGate 構成に追加された新しい表ごとに定義ファイルを生成し、その内容を既存のマスター定義ファイルにコピーしてから、プロセスを再起動する必要があります。

データ定義を取得するための Oracle GoldenGate の構成

データ定義ファイルとテンプレート (必要な場合) を使用するように Oracle GoldenGate を構成するには、次の操作を実行します。

- DEFGEN の構成
- DEFGEN の実行
- リモート・システムへの定義ファイルの転送
- 定義ファイルの指定

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。

DEFGEN の構成

メタデータ定義を取得するシステムで次の手順を実行します。

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 23 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 23　DEFGEN のパラメータ

パラメータ	説明
`CHARSET <character set>`	DEFGEN でパラメータ・ファイルの読取りに使用されるキャラクタ・セットを指定するには、このパラメータを使用します。デフォルトでは、パラメータ・ファイルのキャラクタ・セットはローカル・オペレーティング・システムのキャラクタ・セットです。CHARSET を使用する場合は、パラメータ・ファイルの 1 行目に指定する必要があります。
`DEFSFILE <full_pathname> [APPEND | PURGE] [CHARSET <character set>]` ◆ APPEND を指定した場合、DEFGEN は、指定のファイルがすでに存在すれば、既存の内容の最後に (現在の実行からの) 新しい内容を書き込みます。 ◆ PURGE を指定した場合、DEFGEN は、現在の実行からの新しい内容を書き込む前に、指定のファイルを消去します。この設定がデフォルトです。 ◆ CHARSET を指定した場合、オペレーティング・システムのデフォルト・キャラクタ・セットではなく、指定のキャラクタ・セットで定義ファイルが生成されます。	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。 各パラメータ・オプションとそれらがキャラクタ・セットに与える影響についての重要な情報は、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』の DEFSFILE の説明を参照してください。 キャラクタ・セットと Oracle GoldenGate 全般については、175 ページの「定義ファイルのキャラクタ・セットの影響の理解」を参照してください。

`[{SOURCEDB | TARGETDB} <dsn>]`
`USERID <user>[, PASSWORD <password> [<encryption options>]]`
◆ 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。
◆ USERID とオプションの PASSWORD では、必要に応じてデータベース認証を行います。
	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 23　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、アスタリスク・ワイルドカードで定義された表または表のグループの名前です (この表または表のグループの定義が定義ファイルに出力されます)。DEFGEN では、疑問符ワイルドカードはサポートされません。 ◆ [, {DEF | TARGETDEF} <template name>] を指定すると、この表のメタデータに基づいた定義テンプレートが追加で作成されます。このオプションは、初期ロードではサポートされません。	定義を定義する 1 つ以上の表を指定し、オプションで表のメタデータを定義テンプレートの基礎として使用します。大 / 小文字が区別されるデータベースに対しては、大 / 小文字の区別が維持されます。ワイルドカードと大 / 小文字の区別の詳細は、38 ページの「Oracle GoldenGate の入力におけるオブジェクト名の指定」を参照してください。 ソース定義ファイルを生成する場合はソース表を指定し、ターゲット定義ファイルを生成する場合は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 の実行

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]
[NOEXTATTR] [UPDATECS UTF-8]
```

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。
- NOEXTATTR は、リリース 11.2.1 より古い Oracle GoldenGate との下位互換性を維持するために使用できますが、ASCII 以外のキャラクタ・セット、大 / 小文字の区別、および空白とともに引用符で囲んだオブジェクト名をサポートしません。NOEXTATTR を指定すると、DEFGEN は、Oracle GoldenGate リリース 11.2.1 で導入されたグローバリゼーション機能をサポートするデータベース・ロケールおよびキャラクタ・セットを組み込みません。表名または列名にマルチバイト文字や空白などの特殊文字が含まれている場合、NOEXTATTR を指定すると、DEFGEN は表定義を組み込みません。パラメータ・ファイルで APPEND モードが使用されている場合、NOEXTATTR は無視され、その他の属性が指定されているかどうかに関係なく、新しい表定義が既存のファイル形式で追加されます。
- UPDATECS は、定義ファイルのキャラクタ・セットをリモート・システムと互換性のあるものに変換します。176 ページの「既存の定義ファイルのキャラクタ・セットの変更」を参照してください。

2. 追加で作成する定義ファイルについて、前述の手順を繰り返します。

3. ASCII モードを使用して、ローカルの Oracle GoldenGate dirdef サブディレクトリからリモートの dirdef サブディレクトリに定義ファイルを FTP 送信します。

リモート・システムへの定義ファイルの転送

ローカルとリモートのオペレーティング・システムが異なり、リモート・オペレーティング・システムのキャラクタ・セット用に定義ファイルが作成されている場合に、データ定義ファイルをリモート・システムに FTP 送信するには、BINARY モードを使用します。こうすることで、FTP プログラムによって予期しない文字 (復帰改行文字や改行文字など) がファイルに挿入されるのを回避できます。

定義ファイルの指定

次の方法でデータ定義ファイルを適切な Oracle GoldenGate プロセスに関連付けます。

- 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・ポンプに関連付けます。
- 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 グループに関連付けます。
- 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 に関連付けます。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。

複数の定義ファイルを指定する正しい方法は、179 ページの「定義ファイルの使用例」を参照してください。

定義テンプレートに一致する表の追加

Oracle GoldenGate 構成内の新しい表を定義テンプレートにマップするには、TABLE パラメータと MAP パラメータの次のオプションを適宜使用します。

- ソース定義テンプレートの名前を指定するには、DEF を使用します。
- ターゲット定義テンプレートの名前を指定するには、TARGETDEF を使用します。

これらのオプションを指定すると、Extract プロセスまたは Replicat プロセスで指定のテンプレートと同じ定義が使用されるため、新しい表の定義ファイルを新たに作成したり、プロセスを再起動したりする必要はありません。

定義ファイルの使用例

例 1　ターゲットで使用するソース定義ファイルの作成

次の例では、ソース定義ファイルを出力として作成する DEFGEN パラメータ・ファイルを使用します。この例は、Oracle データベースの表を対象としています。

```
DEFSFILE C:\ggs\dirdef\record.def
USERID ogg, PASSWORD AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC, &
    AES128 KEYNAME mykey1
TABLE acct.cust100, DEF custdef;
TABLE ord.*;
TABLE hr.*;
```

この DEFGEN 構成の結果は次のとおりです。

- 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。
- 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。

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。

```
-- This is a simplified parameter file. Your requirements may vary.
REPLICAT acctrep
USERID ogg, PASSWORD AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC, &
    AES128 KEYNAME mykey1
SOURCEDEFS c:\ggs\dirdef\record.def
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;
```

最初の MAP 文の DEF オプションで指定したとおり、ワイルドカード指定 acct.cust* に一致する表の定義は custdef テンプレートから取得されます。

例 2　ソースで使用するターゲット定義ファイルの作成

同じ表のターゲット定義が必要な場合、その表をプライマリ Extract またはデータ・ポンプに対してマップできます。

- ターゲットが Enscribe データベースの場合、ソース定義のかわりにターゲット定義が必要です。
- マッピングと変換を中間システムで実行する場合、ソース定義に加えてターゲット定義も必要です。

ターゲット定義ファイルを作成するための DEFGEN 構成は次のようになります。

```
DEFSFILE C:\ggs\dirdef\trecord.def
USERID ogg, PASSWORD AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC, &
    AES128 KEYNAME mykey1
TABLE acct.cust100, DEF tcustdef;
TABLE ord.*;
TABLE hr.*;
```

注意　ソース定義ファイルを作成する DEFGEN 構成については、前の例を参照してください。

Extract 構成は次のようになります。

```
-- This is a simplified parameter file. Your requirements may vary.
EXTRACT acctex
USERID ogg, PASSWORD AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC, &
    AES128 KEYNAME mykey1
RMTHOST sysb, MGRPORT 7890, ENCRYPT AES192 KEYNAME mykey1
ENCRYPTTRAIL AES192 KEYNAME mykey2
RMTTRAIL $data.ggsdat.rt
SOURCEDEFS c:\ggs\dirdef\record.def
TARGETDEFS c:\ggs\dirdef\trecord.def
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;
```

例 3 この例では、acc.cust* 表に対して、ソース・テンプレート custdef(record.def ファイルのもの) とターゲット・テンプレート tcustdef(trecord.def ファイルのもの) を使用します。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 (そのかわりに、ここでワイルドカード指定を使用することも可能です)。

例 4 **ターゲットで使用する複数のソース定義ファイルの作成**

これは、複数の定義ファイルの使用方法を示す単純な例です。パラメータの要件は、Oracle GoldenGate トポロジとデータベース・タイプによって異なる場合があります。

次に示すのは、1 つ目のデータ定義ファイルを作成する DEFGEN パラメータ・ファイルです。

```
DEFSFILE C:\ggs\dirdef\sales.def
USERID ogg, PASSWORD AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC, &
    AES128 KEYNAME mykey1
TABLE ord.*;
```

次に示すのは、2 つ目のデータ定義ファイルを作成する DEFGEN パラメータ・ファイルです。ファイル名および表の指定が 1 つ目のものとは異なります。

```
DEFSFILE C:\ggs\dirdef\admin.def
USERID ogg, PASSWORD AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC, &
    AES128 KEYNAME mykey1
TABLE hr.*;
```

1 つ目の定義ファイルと 2 つ目の定義ファイルの表は、同じ Replicat パラメータ・ファイルで次のようにマップされます。

```
REPLICAT acctrep
USERID ogg, PASSWORD AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC, &
    AES128 KEYNAME mykey1
SOURCEDEFS c:\ggs\dirdef\sales.def
MAP ord.*, TARGET ord.*;
SOURCEDEFS c:\ggs\dirdef\admin.def
MAP hr.*, TARGET hr.*;
```

同じ定義と異なる定義の組合せを使用するための Oracle GoldenGate の構成

ASSUMETARGETDEFS と SOURCEDEFS を同じパラメータ・ファイルで使用できます。これが可能なのは、一部のソース表とターゲット表のペアの間では列のマッピングまたは変換を実行する必要があるが、その他の表のペアは同一である場合です。

例 5 これは、同じパラメータ・ファイルで SOURCEDEFS と ASSUMETARGETDEFS を使用する方法を示す例です。この例は、acct、ord、hr の各スキーマの表が SOURCEDEFS を必要とする前の例に基づいていますが、動的に作成され stock という名前がランダムな数値とともに付加される表を持つ rpt スキーマが追加されて

います。Oracle GoldenGate で DML だけでなく DDL もレプリケートするには、ターゲット表が同一であることが必要です。その場合、ASSUMETARGETDEFS を指定する必要があります。

```
REPLICAT acctrep
USERID ogg, PASSWORD AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC, &
    AES128 KEYNAME mykey1
SOURCEDEFS c:\ggs\dirdef\record.def
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;
ASSUMETARGETDEFS
MAP rpt.stock, TARGET rpt.stock;
```

第 14 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、199 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 では、Oracle GoldenGate Manager のインスタンスごとに Extract および Replicat の同時グループが最大 5,000 個サポートされます。サポートされているレベルで、INFO、STATUS などの GGSCI コマンドですべてのグループを完全に制御および表示できます。Oracle GoldenGate では、環境を効果的に管理するために、Extract および Replicat グループの数 (合計) をデフォルト・レベルの 300 以下に保つことが推奨されます。

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。

特定のトポロジに適合させるための変更同期の構成

必要なプロセスおよび証跡の数は、デプロイするレプリケーション・トポロジによって異なります。特定のレプリケーション・トポロジのデプロイの詳細は、次のものを参照してください。

44 ページの「ライブ・レポートのための Oracle GoldenGate の使用」

64 ページの「リアルタイム・データ分散のための Oracle GoldenGate の使用」

70 ページの「リアルタイム・データ・ウェアハウスのための Oracle GoldenGate の構成」

77 ページの「ライブ・スタンバイ・データベース管理のための Oracle GoldenGate の構成」

93 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 の構成」

グループのネーミング規則

グループ名を指定する場合、次のルールに従います。

- 最大 8 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も含めることができます。ローカル・オペレーティング・システムのキャラクタ・セットでサポートされ、オペレーティング・システムでファイル名での使用が許可されているかぎり、任意の文字を使用できます。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。
- { \ / : * ? " < > | } は、グループ名に使用できません。
- 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。
- グループ名に数値を含めることができます。ただし、レポート生成時に書込みプロセスによってグループ名の末尾に数値が追加されるため、グループ名の末尾に数値を使用すると (fin1 など)、レポート・ファイル名の重複やエラーの原因になります。グループ名の先頭であれば、数値を使用しても心配はありません (1_fin や 1fin など)。

チェックポイント表の作成

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、15 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>[<encryption options]]
```

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> および <encryption options> では、必要に応じてデータベース資格証明と暗号化情報を指定します。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。グループのネーミング規則については、184 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。
  - INTEGRATED TRANLOG では、この Extract が統合取得モードで動作して、データベース・ログマイニング・サーバーから論理変更レコード (LCR) を受信することを指定します。このパラメータは、Oracle データベースにのみ適用されます。統合取得の使用の詳細は、『Oracle GoldenGate *Oracle インストレーションおよびセットアップ・ガイド*』を参照してください。
  - VAM 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、137 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。

複数の証跡を作成し、異なる表またはアプリケーションのデータを分けたり、カスケード・トポロジなどの特定のレプリケーション・トポロジの要件に適合させることができます。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 証跡の概念については、14 ページを参照してください。

Oracle GoldenGate 証跡の記憶域の割当て

通常の構成では、少なくともソース・システムに 1 つ、ターゲット・システムに 1 つの証跡があります。次の点を考慮に入れ、十分なディスク領域を割り当てます。

- プライマリ Extract プロセスは、ソース・データベースからトランザクション・データを取得し、ローカル証跡に書き込みます。データ・ポンプ Extract は、その証跡を読み取り、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転送します。ネットワークに障害があると、データ・ポンプは失敗しますが、プライマリ Extract によるローカル証跡へのデータの処理は続けられます。累積していくデータを格納するのに十分なディスク領域がある必要があります。そうでないと、プライマリ Extract が異常終了します。
- ターゲットの場所にある証跡の場合、PURGEOLDEXTRACTS パラメータで設定された消去ルールに従ってデータの累積を処理するのに十分なディスク領域を用意します。PURGEOLDEXTRACTS を使用しても、ターゲット・データベースへの適用よりもネットワーク経由の転送の方が速いため、データはターゲットに常に累積します。

証跡アクティビティがビジネス・アプリケーションによって干渉されないようにするには、別個のディスクまたはファイル・システムを割り当てて証跡ファイルを含めます。証跡ファイルは、Oracle GoldenGate インストールのローカル・ドライブに配置することも、NAS または SAN デバイスに配置することもできます。Oracle クラスタでは、ASM または DBFS 記憶域に配置できます。1GB から始め、必要に応じて調整します。

必要な証跡の領域を見積る手順

1. ネットワークが使用できない可能性のある最長の時間を見積もります。考えられる最長の停止時間に対応するのに十分なデータを格納するよう計画します。そうしないと、停止時間がディスク容量より長くなった場合にソースとターゲットのデータを再同期化する必要があります。
2. ビジネス・アプリケーションで 1 時間に生成されるトランザクション・ログのボリュームを見積もります。
3. 次の式を使用して、必要なディスク領域を計算します。

   [1 時間当たりのログ・ボリューム] x [停止時間] x .4 = 証跡のディスク領域

   Oracle GoldenGate で必要なトランザクション・ログのデータは 40 パーセントのみのため、この式では 40 パーセントという乗数を使用します。

   注意 この式は控えめな見積りです。Oracle GoldenGate の構成後、テストを実行し、必要な領域を正確に割り出します。

証跡の追加

証跡を作成または*追加*する際、ディスクにファイルを物理的に作成しません。ファイルは Extract プロセスによって自動的に作成されます。証跡の名前を指定し、その証跡に書き込む Extract グループに割り当てます。

証跡を追加する手順

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 100 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 200
```

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、137 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 24 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 24 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>[<encryption options>]]`
◆ 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と暗号化オプションを指定します。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 ...`
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。パスワードと暗号化オプションの詳細は、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』の表示パラメータの説明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。

表 24　オンライン変更抽出のパラメータ (続き)

パラメータ	説明
RMTHOST <hostname>, MGRPORT <portnumber>, [, ENCRYPT <algorithm> KEYNAME <keyname>]	ターゲット・システム、Manager が稼働しているポート、および TCP/IP 経由でのデータ暗号化 (オプション) 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。
ENCRYPTTRAIL <encryption options>	このエントリの後に指定した証跡がすべて暗号化されます。暗号化オプションについては、『*Windows and UNIX リファレンス・ガイド*』を参照してください。
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。

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。

表 24　　オンライン変更抽出のパラメータ (続き)

パラメータ	説明
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")
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。グループのネーミング規則については、184 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 (183 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 25 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 25　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。
`{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 13 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。

表 25　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
[TARGETDB <dsn>,] [USERID <user id>][, PASSWORD <pw>[<encryption options>]] ◆ 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と暗号化情報を指定します。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 ...	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。パスワードと暗号化オプションの詳細は、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』の表示パラメータの説明を参照してください。
DECRYPTTRAIL <encryption options>	この Replicat の読み取る入力証跡が暗号化されます。<encryption options> は、この入力証跡の ENCRYPTTRAIL 文で指定したものと一致している必要があります。
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 13 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、199 ページの第 15 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、137 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、137 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>[<encryption options>]
   ```

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>[ <encryption
   options>]]
   ```

   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> および <encryption options> では、必要に応じてデータベース資格証明と暗号化情報を指定します。

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 15 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。

- 202 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。
- 203 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。
- 209 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。
- 214 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。
- 219 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。
- 223 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、『Oracle GoldenGate Oracle インストレーションおよびセットアップ・ガイド』の「Oracle データベース用 DDL 同期の構成」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、21 ページの「Manager およびネットワーク通信の構成」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 13 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。

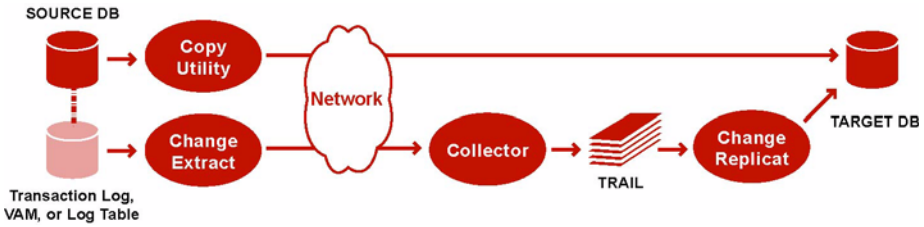
プロセス・グループ間でのパラメータの共有

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、34 ページを参照してください。

マクロの詳細は、232 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. 200 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 [<encryption options>]
   ```

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。リストされていない場合は、このパラメータをファイルに追加します。

警告　ローカル・オペレーティング・システムのものとは異なるキャラクタ・セットを使用した既存のパラメータ・ファイル (CHARSET オプションを使用して別のキャラクタ・セットを指定したファイルなど) は、VIEW PARAMS コマンドまたは EDIT PARAMS コマンドを使用して表示や編集を行わないでください。そのようなパラメータ・ファイルは GGSCI の外部から表示してください。そうしないと、内容が破損する可能性があります。

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
```

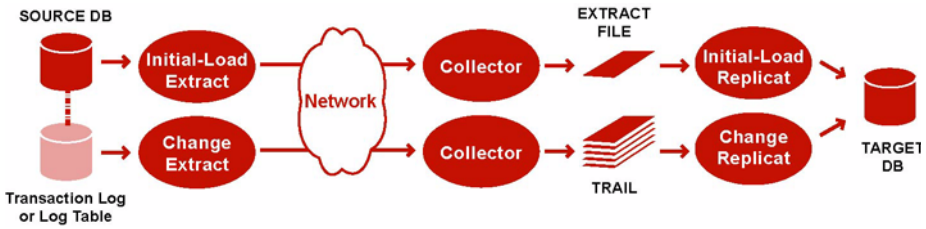
13. ターゲット・システムで、Replicat のパラメータ・ファイルを編集して HANDLECOLLISIONS パラメータを削除します。これによって、次回 Replicat が起動したときに HANDLECOLLISIONS が再度有効化されることを防止します。

警告　ローカル・オペレーティング・システムのものとは異なるキャラクタ・セットを使用した既存のパラメータ・ファイル (CHARSET オプションを使用して別のキャラクタ・セットを指定したファイルなど) は、VIEW PARAMS コマンドまたは EDIT PARAMS コマンドを使用して表示や編集を行わないでください。そのようなパラメータ・ファイルは GGSCI の外部から表示してください。そうしないと、内容が破損する可能性があります。

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. 200 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. 次の表 26 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。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 パラメータ・ファイルの例を次に示します。

   ```
   SOURCEISTABLE
   SOURCEDB mydb, USERID ogg, PASSWORD
       AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC &
       AES128, ENCRYPTKEY securekey1
   RMTHOST ny4387, MGRPORT 7888, ENCRYPT AES 192 KEYNAME mykey
   ENCRYPTTRAIL AES192, KEYNAME mykey1
   RMTFILE /ggs/dirdat/initld, MEGABYTES 2, PURGE
   TABLE hr.*;
   TABLE sales.*;
   ```

表 26　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。

表 26　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
[SOURCEDB <dsn>,] [USERID <user id>[, PASSWORD <pw> [<encryption options>]]] ◆ 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と暗号化情報を指定します。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 ... 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> [, ENCRYPT <algorithm> KEYNAME <keyname>]	ターゲット・システム、Manager が稼働しているポート、および TCP/IP 経由でのデータ暗号化 (オプション) を指定します。
ENCRYPTTRAIL <encryption options>	リモート・ファイルのデータを暗号化します。暗号化オプションについては、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』を参照してください。
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** : 抽出ファイルのサイズは 2GB 以下にする必要があります。
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。

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. 次の表 27 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。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 パラメータ・ファイルの例を次に示します。

```
SPECIALRUN
END RUNTIME
TARGETDB mydb, USERID ogg, PASSWORD
    AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC &
    AES128, ENCRYPTKEY securekey1
DECRYPTTRAIL AES192, KEYNAME mykey1
EXTFILE /ggs/dirdat/initld
SOURCEDEFS /ggs/dirdef/source_defs
MAP hr.*, TARGET hr.*;
MAP sales.*, TARGET hr.*;
```

表 27　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> [<encryption options>]]]`
◆ 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 ...`
	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。

`DECRYPTTRAIL <encryption options>`	入力ファイルのデータを復号化します。暗号化オプションについては、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』を参照してください。

`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 で指定されている入力抽出ファイルを指定します。

表 27　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 13 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 [<encryption options>]
    ```

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。

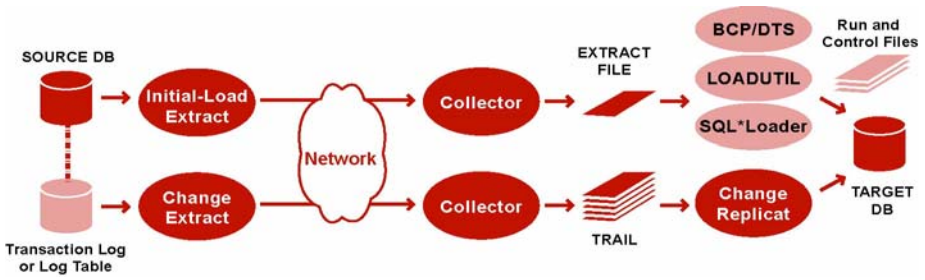
    ```
    SEND REPLICAT <Replicat group name>, NOHANDLECOLLISIONS
    ```

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。

    警告 **ローカル・オペレーティング・システムのものとは異なるキャラクタ・セットを使用した既存のパラメータ・ファイル (CHARSET オプションを使用して別のキャラクタ・セットを指定したファイルなど) は、VIEW PARAMS コマンドまたは EDIT PARAMS コマンドを使用して表示や編集を行わないでください。そのようなパラメータ・ファイルは GGSCI の外部から表示してください。そうしないと、内容が破損する可能性があります。**
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. 200 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. 次の表 28 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。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ための初期ロード Extract パラメータ・ファイルの例を次に示します。

   ```
   SOURCEISTABLE
   SOURCEDB mydb, USERID ogg, PASSWORD
       AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC &
       AES128, ENCRYPTKEY securekey1
   RMTHOST ny4387, MGRPORT 7888, ENCRYPT AES 192 KEYNAME mykey
   ENCRYPTTRAIL AES192, KEYNAME mykey1
   RMTFILE /ggs/dirdat/initld, MEGABYTES 2, PURGE
   FORMATASCII, SQLLOADER
   TABLE hr.*;
   TABLE sales.*;
   ```

表 28　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。
[SOURCEDB <dsn>,] [USERID <user id>[, PASSWORD <pw> [<encryption options>]]] ◆ 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 ... 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> [, ENCRYPT <algorithm> KEYNAME <keyname>] [, 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。	ターゲット・システム、Manager が稼働しているポート、および TCP/IP 経由でのデータ暗号化 (オプション) 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)。
ENCRYPTTRAIL <encryption options>	リモート・ファイルのデータを暗号化します。暗号化オプションについては、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』を参照してください。
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 (最大 2MB) 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。

表 28　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
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. 次の表 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。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ための初期ロード Replicat パラメータ・ファイルの例を次に示します。

```
GENLOADFILES sqlldr.tpl
USERID ogg, PASSWORD
    AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC &
    AES128, ENCRYPTKEY securekey1
DECRYPTTRAIL AES192, KEYNAME mykey1
EXTFILE /ggs/dirdat/initld
SOURCEDEFS /ggs/dirdef/source_defs
MAP hr.*, TARGET hr.*;
MAP sales.*, TARGET hr.*;
```

表 29　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。

表 29　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

`[TARGETDB <dsn>,]`
`[USERID <user id>[, PASSWORD <pw> [encryption options>]]]`
◆ 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 ...`
	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。

`DECRYPTTRAIL <encryption options>`	入力抽出ファイルのデータを復号化します。暗号化オプションについては、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』を参照してください。

`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 13 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. 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 [<encryption options>]
```

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
```

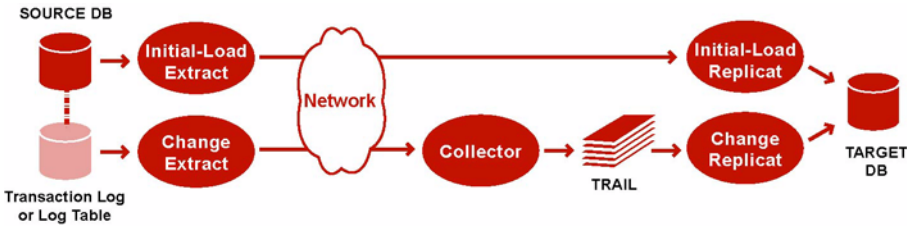
25. ターゲット・システムで、Replicat のパラメータ・ファイルを編集して HANDLECOLLISIONS パラメータを削除します。これによって、次回 Replicat が起動したときに HANDLECOLLISIONS が再度有効化されることを防止します。

警告 **ローカル・オペレーティング・システムのものとは異なるキャラクタ・セットを使用した既存のパラメータ・ファイル (CHARSET オプションを使用して別のキャラクタ・セットを指定したファイルなど) は、VIEW PARAMS コマンドまたは EDIT PARAMS コマンドを使用して表示や編集を行わないでください。そのようなパラメータ・ファイルは GGSCI の外部から表示してください。そうしないと、内容が破損する可能性があります。**

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、203 ページの「ファイルから Replicat へのデータのロード」または 209 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し、検索およびバインドのプロセスを高速化するには、Manager のパラメータ・ファイルで DYNAMICPORTLIST パラメータを指定します。Manager は、このパラメータで指定されたポート番号のリストを Replicat タスク・プロセスに渡します。Replicat はまずこのリストのポートを検索し、リストのポートが使用できない場合にのみ、デフォルトの Manager ポート番号からスキャンを開始して、使用可能なポートが見つかるまで昇順でスキャンを続けます。

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. 200 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. 次の表 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。Oracle GoldenGate ダイレクト・ロード用の初期ロード Extract パラメータ・ファイルの例を次に示します。

   ```
   EXTRACT initext
   SOURCEDB mydb, USERID ogg, PASSWORD
       AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC &
       AES128, ENCRYPTKEY securekey1
   RMTHOST ny4387, MGRPORT 7888, ENCRYPT AES 192 KEYNAME mykey
   RMTTASK REPLICAT, GROUP initrep
   TABLE hr.*;
   TABLE sales.*;
   ```

表 30　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>`	手順 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> [<encryption options>]]]`
◆ 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 ...`
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>` `[, ENCRYPT <algorithm> KEYNAME <keyname>]`	ターゲット・システム、Manager が稼働しているポート、および TCP/IP 経由でのデータ暗号化 (オプション) 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. 次の表 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。Oracle GoldenGate ダイレクト・ロード用の初期ロード Replicat パラメータ・ファイルの例を次に示します。

```
REPLICAT initrep
TARGETDB mydb, USERID ogg, PASSWORD
    AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC &
    AES128, ENCRYPTKEY securekey1
SOURCEDEFS /ggs/dirdef/source_defs
MAP hr.*, TARGET hr.*;
MAP sales.*, TARGET hr.*;
```

表 31　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。手順 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> [<encryption options>]]]`
◆ 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と暗号化情報を指定します。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 ...`
	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 13 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 [<encryption options>]
```

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。

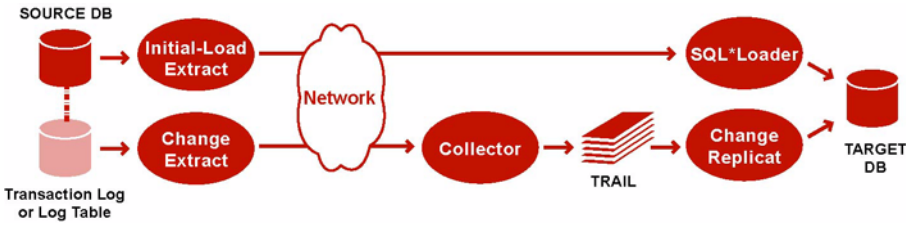
```
SEND REPLICAT <Replicat group name>, NOHANDLECOLLISIONS
```

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。

警告 ローカル・オペレーティング・システムのものとは異なるキャラクタ・セットを使用した既存のパラメータ・ファイル (CHARSET オプションを使用して別のキャラクタ・セットを指定したファイルなど) は、VIEW PARAMS コマンドまたは EDIT PARAMS コマンドを使用して表示や編集を行わないでください。そのようなパラメータ・ファイルは GGSCI の外部から表示してください。そうしないと、内容が破損する可能性があります。

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し、検索およびバインドのプロセスを高速化するには、Manager のパラメータ・ファイルで DYNAMICPORTLIST パラメータを指定します。Manager は、このパラメータで指定されたポート番号のリストを Replicat タスク・プロセスに渡します。Replicat はまずこのリストのポートを検索し、リストのポートが使用できない場合にのみ、デフォルトの Manager ポート番号からスキャンを開始して、使用可能なポートが見つかるまで昇順でスキャンを続けます。

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、203 ページの「ファイルから Replicat へのデータのロード」または 209 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. 200 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. 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 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。SQL*Loader へのダイレクト・バルク・ロード用の初期ロード Extract パラメータ・ファイルの例を次に示します。

```
EXTRACT initext
SOURCEDB mydb, USERID ogg, PASSWORD
    AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC &
    AES128, ENCRYPTKEY securekey1
RMTHOST ny4387, MGRPORT 7888, ENCRYPT AES 192 KEYNAME mykey
RMTTASK REPLICAT, GROUP initrep
TABLE hr.*;
TABLE sales.*;
```

表 32　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>`	手順 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> [<encryption options>]]]`
◆ 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と暗号化情報を指定します。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 ...`
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>`
`[, ENCRYPT <algorithm> KEYNAME <keyname>]`
	ターゲット・システム、Manager が稼働しているポート、および TCP/IP 経由でのデータ暗号化 (オプション) を指定します。

表 32　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 (続き)

パラメータ	説明

`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 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。SQL*Loader へのダイレクト・ロード用の初期ロード Replicat パラメータ・ファイルの例を次に示します。

    ```
    REPLICAT initrep
    USERID ogg, PASSWORD
        AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC &
        AES128, ENCRYPTKEY securekey1
    BULKLOAD
    SOURCEDEFS /ggs/dirdef/source_defs
    MAP hr.*, TARGET hr.*;
    MAP sales.*, TARGET hr.*;
    ```

表 33　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。手順 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> [<encryption options>]`	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、パスワードおよび暗号化オプションを指定します。次のようなホスト文字列を含めることができます。 `USERID ogg@ora1.ora, PASSWORD & AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC, & AES128 KEYNAME mykey1` 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 13 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。

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 [<encryption options>]
    ```

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。

    警告 **ローカル・オペレーティング・システムのものとは異なるキャラクタ・セットを使用した既存のパラメータ・ファイル (CHARSET オプションを使用して別のキャラクタ・セットを指定したファイルなど) は、VIEW PARAMS コマンドまたは EDIT PARAMS コマンドを使用して表示や編集を行わないでください。そのようなパラメータ・ファイルは GGSCI の外部から表示してください。そうしないと、内容が破損する可能性があります。**

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 14 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・ファイルを編集のために開きます。

   警告 **ローカル・オペレーティング・システムのものとは異なるキャラクタ・セットを使用した既存のパラメータ・ファイル (CHARSET オプションを使用して別のキャラクタ・セットを指定したファイルなど) は、VIEW PARAMS コマンドまたは EDIT PARAMS コマンドを使用して表示や編集を行わないでください。そのようなパラメータ・ファイルは GGSCI の外部から表示してください。そうしないと、内容が破損する可能性があります。**

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 16 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。

- 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 マクロを使用した作業の簡略化および自動化
- ユーザー・イグジットを使用した Oracle GoldenGate 機能の拡張
- 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

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 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 (227 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。 引用符で囲んだオブジェクト名を SQLEXEC 問合せで使用するには、SQL 問合せを二重引用符ではなく一重引用符で囲み、USEANSISQLQUOTES パラメータを GLOBALS ファイルで使用して、オブジェクトおよびリテラルの識別子に SQL-92 ルールを適用する必要があります。次に示すのは、引用符で囲んだオブジェクト名を問合せで使用する例です。 `SQLEXEC 'SELECT "col1" from "schema"."table"'`
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 (227 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。

パラメータ構文	用途
`SQLEXEC "call <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
`"call <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 "call 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。 引用符で囲んだオブジェクト名を SQLEXEC 問合せで使用するには、SQL 問合せを二重引用符ではなく一重引用符で囲み、USEANSISQLQUOTES パラメータを GLOBALS ファイルで使用して、オブジェクトおよびリテラルの識別子に SQL-92 ルールを適用する必要があります。次に示すのは、引用符で囲んだオブジェクト名を問合せで使用する例です。 `SQLEXEC 'SELECT "col1" from "schema"."table"'` 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、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、該当する列のサプリメンタル・ロギングを有効化できます。

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 34 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。

表 34　ERROR オプション (続き)

アクション	説明
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 35　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。233 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。233 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)。234 ページの「入力パラメータを使用したマクロの使用」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。

表 35　マクロ定義の引数 (続き)

引数	説明

<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 36　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)`

入力パラメータを使用したマクロの使用

マクロでの入力パラメータの使用は、オプションです。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)
);
```

入力パラメータを使用しないマクロの使用

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;
```

パラメータ置換の使用

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。(236 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. 232 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 ogg@ora1.ora, PASSWORD &
   AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC, &
   AES128 KEYNAME mykey1
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 Extract または Replicat の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。

Oracle GoldenGate レコード情報のルーチンへの使用可能化

ほとんどのユーザー・イグジット処理の基本となるのが EXIT_CALL_PROCESS_RECORD 関数です。Extract では、この関数はレコード・バッファが証跡に出力される直前にコールされます。Replicat では、これはレコードがターゲットに適用される直前にコールされます。パラメータ・ファイルにソースとターゲットのマッピングが指定されている場合、EXIT_CALL_PROCESS_RECORD イベントはマッピングの実行後に発生します。

EXIT_CALL_PROCESS_RECORD がコールされると、コールバック・ルーチンを通じて、レコード・バッファとその他のレコード情報が利用可能になります。ユーザー・イグジットは、マッピング、変換、クリーンアップ、その他すべての操作をデータ・レコードに対して実行できます。操作が完了したら、ユーザー・イグジットは、Extract または Replicat によってレコードを処理するのか、無視するのかを示すステータスを戻すことができます。

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。

注意　ユーザー・イグジットは、データベース・オブジェクト名の大 / 小文字を区別します。名前は、ホスト・データベースで定義されたとおりに戻されます。

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、usrdecs.h ファイルをインクルードします。usrdecs.h ファイルは、ユーザー・イグジット API 用のインクルード・ファイルです。型定義、戻りステータス値、コールバック関数コード、および多数の他の定義が含まれます。usrdecs.h ファイルは、Oracle GoldenGate ディレクトリ内にインストールされます。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
```

キャラクタ・セットの変換のサポート

データ整合性を保つには、ユーザー・イグジットで、Oracle GoldenGate プロセスと交換する文字型データのキャラクタ・セットを認識する必要があります。Oracle GoldenGate ユーザー・イグジットのロジックでは、次のものに対するグローバリゼーション・サポートが提供されます。

- 文字ベースのデータベース・メタデータ (カタログ、スキーマ、表および列の名前など)
- 文字型の列 (CHAR、VARCHAR2、CLOB、NCHAR、NVARCHAR2、NCLOB など) の値や文字列ベースの数値、日時および期間

キャラクタ・セット間での適切な変換によって、列データの比較、操作、変換およびあるタイプのデータベースとキャラクタ・セットから別のものへの適切なマップが可能になります。この処理の大半は、EXIT_CALL_PROCESS_RECORD コール・タイプがコールされると実行され、レコード・バッファと他のレコード情報がコールバック・ルーチンを介して使用可能になります。

ユーザー・イグジットは独自のセッション・キャラクタ・セットを持ちます。これは、GET_SESSION_CHARSET および SET_SESSION_CHARSET コールバック関数によって定義されます。ユーザー・イグジットのキャラクタ・セットがプロセスのホスト・コンテキストと異なる場合、コール元のプロセスでキャラクタ・セット間の変換が行われます。

ユーザー・イグジットでこのサポートを有効にするために、GET_DATABASE_METADATA コールバック関数コードがあります。この関数によって、ユーザー・イグジットはデータベース・メタデータ (ロケールやコール元プロセス (Extract、データ・ポンプ、Replicat) と交換する文字型データのキャラクタ・セットなど) を取得できます。データベースでのオブジェクト名の大 / 小文字の区別、引用符付きおよび引用符なしの名前の処理およびオブジェクト名の格納方法も返されます。

これらのコンポーネントの詳細は、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』を参照してください。

名前のメタデータをチェックするマクロの使用

ユーザー・イグジット API によって渡されるオブジェクト名は、ユーザー・イグジット・セッション・キャラクタ・セットでエンコードされた正確な名前で、データベースから取得されたものと完全に同じ名前です。ユーザー・イグジットでオブジェクト名をリテラル文字列と比較する場合、ユーザー・イグジットでデータベースのロケールを取得し、文字列を正規化して同じエンコーディングでオブジェクト名と比較できるようにする必要があります。

Oracle GoldenGate には、ユーザー・イグジットでコールし、データベース・オブジェクト名のメタデータをチェックできる次のマクロが用意されています。たとえば、マクロを使用して、引用符で囲まれた名前で大 / 小文字が区別されるかどうかや、大 / 小文字混在でデータベース・サーバーに格納されているかどうかをチェックできます。これらのマクロは、usrdecs.h ファイルで定義されています。

表 37　メタデータ・チェック用のマクロ

マクロ	確認の対象
supportsMixedCaseIdentifiers(nameMeta, DbObjType)	大 / 小文字混在で引用符で囲まれていない、指定されたデータ型の名前を、データベースが大 / 小文字を区別して処理し、大 / 小文字混在で格納するかどうか。
supportsMixedCaseQuotedIdentifiers(nameMeta, DBObjType)	大 / 小文字混在で引用符で囲まれた、指定されたデータ型の名前を、データベースが大 / 小文字を区別して処理し、大 / 小文字混在で格納するかどうか。
storesLowerCaseIdentifiers(nameMeta, DbObjType)	大 / 小文字混在で引用符で囲まれていない、指定されたデータ型の名前を、データベースが大 / 小文字を区別せずに処理し、小文字で格納するかどうか。
storesLowerCaseQuotedIdentifiers(nameMeta, DbObjType)	大 / 小文字混在で引用符で囲まれた、指定されたデータ型の名前を、データベースが大 / 小文字を区別せずに処理し、小文字で格納するかどうか。

表 37　メタデータ・チェック用のマクロ (続き)

マクロ	確認の対象
storesMixedCaseIdentifiers(nameMeta, DbObjType)	大 / 小文字混在で引用符で囲まれていない、指定されたデータ型の名前を、データベースが大 / 小文字を区別せずに処理し、大 / 小文字混在で格納するかどうか。
storesMixedCaseQuotedIdentifiers(nameMeta, DbObjType)	大 / 小文字混在で引用符で囲まれた、指定されたデータ型の名前を、データベースが大 / 小文字を区別せずに処理し、大 / 小文字混在で格納するかどうか。
storesUpperCaseIdentifiers(nameMeta, DbObjType)	大 / 小文字混在で引用符で囲まれていない、指定されたデータ型の名前を、データベースが大 / 小文字を区別せずに処理し、大文字で格納するかどうか。
storesUpperCaseQuotedIdentifiers(nameMeta, DbObjType)	大 / 小文字混在で引用符で囲まれた、指定されたデータ型の名前を、データベースが大 / 小文字を区別せずに処理し、大文字で格納するかどうか。

文字形式の説明

入力パラメータ column_value_mode は、処理されるデータの文字形式を表し、いくつかの関数コードで使用されます。次の表では、EXIT_FN_RAW_FORMAT、EXIT_FN_CHAR_FORMAT および EXIT_FN_CNVTED_SESS_FORMAT 形式コードの意味を、データ型ごとに説明します。

表 38　column_value_mode_matrix の意味

データ型	EXIT_FN_RAW_FORMAT	EXIT_FN_CHAR_FORMAT	EXIT_FN_CNVTED_SESS_FORMAT
CHAR "abc	"2 バイト NULL インジケータ + 2 バイトの長さ情報 + 列値 0000 0004 61 62 63 20	ASCII または EBCDIC でエンコードされた "abc"。 NULL 終端。 末尾の空白は切り捨てられます。	ユーザー・イグジット・セッションのキャラクタ・セットでエンコードされた "abc"。 NOT NULL 終端。 GLOBALS パラメータの NOTRIMSPACES が指定されていない場合、デフォルトで末尾の空白は切り捨てられます。
NCHAR 0061 0062 0063 0020	2 バイト NULL インジケータ + 2 バイトの長さ情報 + 列値。 0000 0008 00 61 0062 0063 0020	NCHAR が UTF-8 として処理されるかどうかに応じて、"abc" (UTF8 でエンコード) または最初のバイトで切捨て。 NULL 終端。 末尾の空白は切り捨てられます。	ユーザー・イグジット・セッションのキャラクタ・セットでエンコードされた "abc"。 NOT NULL 終端。 GLOBALS パラメータの NOTRIMSPACES が指定されていない場合、デフォルトで末尾の空白は切り捨てられます。

表 38　column_value_mode_matrix の意味 (続き)

データ型	EXIT_FN_RAW_FORMAT	EXIT_FN_CHAR_FORMAT	EXIT_FN_CNVTED_SESS_FORMAT
VARCHAR2 "abc"	2 バイト NULL インジケータ + 2 バイトの長さ情報 + 列値	ASCII または EBCDIC でエンコードされた "abc"。 NULL 終端。 切捨てなし。	ユーザー・イグジット・セッションのキャラクタ・セットでエンコードされた "abc"。 NOT NULL 終端。 切捨てなし。
NVARCHAR2 0061 0062 0063 0020	2 バイト NULL インジケータ + 2 バイトの長さ情報 + 列値	NVARCHAR2 が UTF-8 として処理されるかどうかに応じて、"abc" (UTF8 でエンコード) または最初のバイトで切捨て。 NULL 終端。 切捨てなし。	ユーザー・イグジット・セッションのキャラクタ・セットでエンコードされた "abc"。 NOT NULL 終端。 切捨てなし。
CLOB	2 バイト NULL インジケータ + 2 バイトの長さ情報 + 列値	VARCHAR2 と同様ですが、最大 4K バイトのみの出力。 NULL 終端。 切捨てなし。	VARCHAR2 と同様ですが、ユーザー・イグジット・セッションのキャラクタ・セットでリクエストされたデータのみ出力。 NOT NULL 終端。 切捨てなし。
NCLOB	2 バイト NULL インジケータ + 2 バイトの長さ情報 + 列値	NVARCHAR2 と同様ですが、最大 4K バイトのみの出力。 NULL 終端。 切捨てなし。	NVARCHAR2 と同様ですが、ユーザー・イグジット・セッションのキャラクタ・セットでリクエストされたデータのみ出力。 NOT NULL 終端。 切捨てなし。
NUMBER 123.89	2 バイト NULL インジケータ + 2 バイトの長さ情報 + 列値	ASCII または EBCDIC でエンコードされた "123.89"。 NULL 終端。	ユーザー・イグジット・セッションのキャラクタ・セットでエンコードされた "123.89"。 NOT NULL 終端。
DATE 31-May-11	2 バイト NULL インジケータ + 2 バイトの長さ情報 + 列値	ASCII または EBCDIC でエンコードされた "2011-05-31"。 NULL 終端。	ユーザー・イグジット・セッションのキャラクタ・セットでエンコードされた "2011-05-31"。 NOT NULL 終端。
TIMESTAMP 31-May-11 12.00.00 AM	2 バイト NULL インジケータ + 2 バイトの長さ情報 + 列値	ASCII または EBCDIC でエンコードされた "2011-05-31 12.00.00 AM"。 NULL 終端。	ユーザー・イグジット・セッションのキャラクタ・セットでエンコードされた "2011-05-31 12.00.00 AM"。 NOT NULL 終端。
Interval Year to Month または Interval Day to Second	2 バイト NULL インジケータ + 2 バイトの長さ情報 + 列値	NA	NA
RAW	2 バイト NULL インジケータ + 2 バイトの長さ情報 + 列値	2 バイト NULL インジケータ + 2 バイトの長さ情報 + 列値	2 バイト NULL インジケータ + 2 バイトの長さ情報 + 列値

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。

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_utf16.c は、ユーザー・イグジットとコール元プロセスの間で交換される情報のコールバック構造で、UTF16 エンコードのデータ (メタデータと列データの両方) を使用する方法を示しています。
- 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。

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。

たとえば、イベント・マーカー・システムを使用して、プロセスの起動、一時停止または停止、変換の実行、統計のレポートを行うことができます。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か電子メールを送信するシェル・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
- 1 日の終わりにレポートまたはバッチ処理を実行するプロセスの停止または一時停止

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。

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、TABLE 文または MAP 文の FILTER、WHERE または SQLEXEC で指定できます。あるいは、Replicat パラメータ・ファイルに特別な TABLE 文を指定すると、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ます。
2. イベント・レコードを指定する 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。

   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。

EVENTACTIONS のオプションは、次の例に示すように組み合せることができます。

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・マーカー・システムの使用方法のケース・スタディ

次の各例では、イベント・マーカー・システムの使用方法に重点を置いて説明します。構文の詳細は、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』を参照してください。

日次処理の起動

```
TABLE source.event_table, EVENTACTIONS (IGNORE, LOG INFO, STOP);
```

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。

初期ロードから変更同期への移行の簡略化

```
TABLE source.event_table, EVENTACTIONS (IGNORE, LOG INFO, STOP);
```

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。

データ異常値が検出された場合の処理の停止

```
MAP source.account, TARGET target.account;
TABLE source.account, FILTER (withdrawal > balance), EVENTACTIONS (ABORT);
```

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。

ハートビート表およびロギングを使用したラグの分析

```
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);
```

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。

特定の注文番号のトレース

```
MAP sales.order, TARGET rpt.order;
TABLE source.order,
FILTER (@GETENV ("GGHEADER", "OPTYPE") = "INSERT" AND order_no = 1), &
EVENTACTIONS (TRACE order_1.trc TRANSACTION);
```

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。

バッチ・プロセスの実行

```
TABLE source.job, FILTER (@streq(job_type = "HOUSEKEEPING")=1), &
EVENTACTIONS (IGNORE TRANSACTION);
```

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。

結果となる操作を除く SQL 文のみの伝播

Extract:

```
TABLE source.statement, EVENTACTIONS (IGNORE TRANS INCLUDEEVENT);
```

Replicat:

```
TABLE source.statement, SQLEXEC (<execute SQL statement>), &
EVENTACTIONS (INFO, IGNORE);
```

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。

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

```
TABLE source.batch_table, EVENTACTIONS (CHECKPOINT BEFORE);
```

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。

データ検証のためのシェル・スクリプトの実行

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);
```

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。

第 17 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 39　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 39　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 19 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、ラグしきい値を指定します。ラグが指定した値を超えると、Oracle GoldenGate により、ラグ情報がエラー・ログに記録されます。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 20 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 21 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 22 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 23 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

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。

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 24 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。

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。

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 18 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。Oracle GoldenGate 環境でアプリケーションのパッチまたはアップグレードを適用するには、次のいずれかを実行します。

- 使用中のデータベースのタイプに対するDDLレプリケーションがOracle GoldenGateでサポートされている場合は、DDL レプリケーションを使用すると、レプリケーション・プロセスを停止せずに DDL をレプリケートできます。この方法を使用するには、ソース表とターゲット表の構造が同一である必要があります。
- レプリケーションを継続させるための適切な手順を踏んだ後、ソースとターゲットの両方でパッチまたはアップグレードを手動で適用できます。

Oracle GoldenGate を使用してパッチの DDL をレプリケートする手順

1. ある程度時間をかけて Oracle GoldenGate の DDL サポートについて学習し、DDL サポートのインストールと構成を行います (まだ行っていない場合)。このドキュメントで、使用中のデータベースに関する指示を参照してください。DDL 環境を整えておけば、将来パッチやアップグレードを適用するのが容易になります。
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、必ずそれらのオブジェクトを DDL パラメータ文に追加します。新規オブジェクトを TABLE 文および MAP 文に追加する場合は、270 ページの手順を参照してください。
3. 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操作か

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. パッチによって表の定義が変更された場合、ソース表に対し DEFGEN を実行して更新後のソース定義を生成し、ターゲット・システムにある既存のソース定義ファイルの古い定義を新しい定義に置き換えます。
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。次に例を示します。

```
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。

   警告 **ローカル・オペレーティング・システムのものとは異なるキャラクタ・セットを使用した既存のパラメータ・ファイル (CHARSET オプションを使用して別のキャラクタ・セットを指定したファイルなど) は、VIEW PARAMS コマンドまたは EDIT PARAMS コマンドを使用して表示や編集を行わないでください。そのようなパラメータ・ファイルは GGSCI の外部から表示してください。そうしないと、内容が破損する可能性があります。**

   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手順 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> [<encryption
options>]]]
```

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

この手順では、Oracle GoldenGate の DDL サポート機能が使用されていない、または使用中のデータベースでサポートされていないと仮定しています。

注意 Oracle および Teradata のデータベースの場合、この手順を使用するかわりに、Oracle GoldenGate の DDL サポート機能を有効化し、新しい表を追加する DDL を自動的に取得して適用することができます。このドキュメントで、使用中のデータベースに該当する指示を参照してください。

始める前に、次の手順に目を通してください。このプロセスは、新しい表が TABLE パラメータのワイルドカードに一致するかどうかと、名前またはデータ定義をターゲットでマップする必要があるかどうかによって、若干変わります。

前提条件

- この手順では、ソース表とターゲット表が空であるか、同一の (すでに同期されている) データを含んでいると仮定しています。
- DBLOGIN コマンドと ADD TRANDATA コマンドを使用する可能性があります。この手順を開始する前に、『Oracle GoldenGate *Windows and UNIX リファレンス・ガイド*』を参照し、使用中のデータベースでのこれらのコマンドの正しい使用方法を確認してください。

Oracle GoldenGate 構成に表を追加する手順

1. 新しい表へのユーザー・アクセスを停止します。
2. (*新しい表がワイルドカードに一致しない場合*) ワイルドカードに一致しない表を多数追加する場合、Extract および Replicat のパラメータ・ファイルのコピーを作成し、TABLE 文および MAP 文で新しい表を追加します。コピーを作成しない場合は、各プロセスを停止するよう要求された後で、元のパラメータ・ファイルを編集します。
3. (*新しい表がワイルドカードに一致する場合*)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。
4. (*新しい表がワイルドカードに一致しない場合*) 新しい表がワイルドカード定義に一致*しない*場合、Extract を停止します。

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

5. 新しい表をソース・データベースおよびターゲット・データベースに追加します。
6. ソース・データベースでの必要に応じて、GGSCI で新しい表に対して ADD TRANDATA コマンドを発行します。ADD TRANDATA を使用する前に、DBLOGIN コマンドを発行します。
7. ソース定義とターゲット定義が同一か、異なるかに応じて、Replicat のパラメータ・ファイルで ASSUMETARGETDEFS または SOURCEDEFS を使用します。SOURCEDEFS が必要な場合は、次のいずれかの操作を実行できます。
   - DEFGEN を実行し、ターゲットのソース定義ファイルに新しい定義をコピーします。
   - 新しい表が定義テンプレートに一致する場合は、MAP パラメータの DEF オプションでテンプレートを指定します。(DEFGEN は不要です。)
8. 新しいソース定義または新しい MAP 文を登録するには、Replicat を停止して起動します。

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

9. Extract を起動します (該当する場合)。

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

10. 新しい表へのユーザー・アクセスを許可します。

ソースとターゲットの間の表属性の調整

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) など、Oracle GoldenGate 構成内のソース表の属性を変更する場合は、次の手順に従ってください。ここでは、レプリケーション待機時間を発生させずにターゲット表に同じ変更を加える方法を示します。

注意　この手順では、Oracle GoldenGate の DDL サポート機能が使用されていない、または使用中のデータベースでサポートされていないと仮定しています。Oracle および Teradata のデータベースの場合、この手順を使用するかわりに、Oracle GoldenGate の DDL サポート機能を有効化して DDL の変更をターゲットに伝播することができます。

1. ソース・システムとターゲット・システムで、トランザクション・ログ内の停止ポイントを示すマーカーを生成するための表 (マーカー表) を作成します。単純な列を 2 つ作成します (一方は主キー、もう一方は通常の列です)。次に例を示します。

```
CREATE TABLE marker
(
id int NOT NULL,
column varchar(25) NOT NULL,
PRIMARY KEY (id)
);
```

2. ソース・システムとターゲット・システムの両方で、マーカー表に行を挿入します。

```
INSERT INTO marker VALUES (1, 1);
COMMIT;
```

3. ソース・システムで、GGSCI を実行します。
4. Extract のパラメータ・ファイルを編集のために開きます。

警告　ローカル・オペレーティング・システムのものとは異なるキャラクタ・セットを使用した既存のパラメータ・ファイル (CHARSET オプションを使用して別のキャラクタ・セットを指定したファイルなど) は、VIEW PARAMS コマンドまたは EDIT PARAMS コマンドを使用して表示や編集を行わないでください。そのようなパラメータ・ファイルは GGSCI の外部から表示してください。そうしないと、内容が破損する可能性があります。

5. Extract のパラメータ・ファイルの TABLE 文にマーカー表を追加します。

```
TABLE marker;
```

6. パラメータ・ファイルを保存して閉じます。
7. データ・ポンプを使用している場合、データ・ポンプの TABLE 文にマーカー表を追加します。
8. Extract およびデータ・ポンプのプロセスを停止し、取得のラグが発生しないよう、ただちに再起動します。

```
STOP EXTRACT <group>
START EXTRACT <group>

STOP EXTRACT <data pump>
START EXTRACT <data pump>
```

9. ターゲット・システムで、GGSCI を実行します。
10. Replicat のパラメータ・ファイルを編集のために開きます。

    警告 ローカル・オペレーティング・システムのものとは異なるキャラクタ・セットを使用した既存のパラメータ・ファイル (CHARSET オプションを使用して別のキャラクタ・セットを指定したファイルなど) は、VIEW PARAMS コマンドまたは EDIT PARAMS コマンドを使用して表示や編集を行わないでください。そのようなパラメータ・ファイルは GGSCI の外部から表示してください。そうしないと、内容が破損する可能性があります。

11. Replicat のパラメータ・ファイルの MAP 文にマーカー表を追加し、次に示すように、EVENTACTIONS パラメータを使用して Replicat を停止し、マーカー表に対する操作を無視します。

    ```
    MAP marker, TARGET marker, EVENTACTIONS (STOP, IGNORE);
    ```

12. パラメータ・ファイルを保存して閉じます。
13. Replicat プロセスを停止し、ただちに再起動します。

    ```
    STOP REPLICAT <group>
    START REPLICAT <group>
    ```

14. ソース表とターゲット表両方の表属性を変更する準備ができたら、それらの表に対するユーザー・アクティビティをすべて停止します。
15. ソース・システムで、マーカー表に対する UPDATE 操作をトランザクションの唯一の操作として実行します。

    ```
    UPDATE marker
    SET column=2,
    WHERE id=1;
    COMMIT;
    ```

16. ターゲット・システムで、EVENTACTIONS ルールの結果として Replicat が停止したことが示されるまで、次のコマンドを発行します。

    ```
    STATUS REPLICAT <group>
    ```

17. ソース表とターゲット表に対して DDL を実行しますが、まだユーザー・アクティビティは許可しません。
18. Replicat を起動します。

    ```
    START REPLICAT <group>
    ```

19. ソース表とターゲット表に対するユーザー・アクティビティを許可します。

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プライマリ Extract グループとデータ・ポンプを削除します。

```
DELETE EXTRACT <group name>
DELETE EXTRACT <pump name>
```

4. ターゲット・システムで、次のコマンドを発行して Replicat グループを削除します。

```
DELETE REPLICAT <group name>
```

5. 標準のオペレーティング・システム・コマンドを使用して、ローカルおよびリモートの証跡ファイルを削除します。
6. 新しい RAC スレッド数を指定し、プライマリ Extract グループを以前と同じ名前で再度追加します。

```
ADD EXTRACT <group name> TRANLOG, THREADS <n>, BEGIN NOW
```

7. ローカル証跡を以前と同じ名前で再度追加します。

```
ADD EXTTRAIL <trail name>, EXTRACT <group name>
```

8. データ・ポンプ Extract を以前と同じ名前で再度追加します。

```
ADD EXTRACT <group name> EXTTRAILSOURCE <trail name>, BEGIN NOW
```

9. リモート証跡を以前と同じ名前で再度追加します。

```
ADD RMTTRAIL <trail name>, EXTRACT <group name>
```

10. Replicat グループを以前と同じ名前で追加します。BEGIN オプションは外し、証跡の最初から処理が開始されるようにします。

```
ADD REPLICAT <group name> EXTTRAIL <trail name>
```

11. ワイルドカードを適宜使用して、すべてのプロセスを起動します。再作成したプロセスがソースおよびターゲットの Oracle GoldenGate インスタンスにおける唯一のプロセスである場合は、次のコマンドのかわりに START ER * を使用することもできます。

```
START EXTRACT <group name>
START REPLICAT <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 19 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 25 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 40　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

反転処理は、GGSCI を通じて作成および開始したオンラインの Extract プロセスと Replicat プロセスを使用して実行します。これらのプロセスは、標準のローカル証跡またはリモート証跡に書き込みます。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 41 および表 42 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 41　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>`	EXTRACT <group> では、Extract プロセスを指定します。GGSCI でこのプロセスを作成します。

表 41　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
`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>`
`[<encryption options>]]]`
◆ 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と暗号化情報を指定します。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 ...`
	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。
`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。
`{EXTTRAIL <input trail> |` `RMTTRAIL <input trail>}`	◆ 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`

表 41　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
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、リモート証跡を使用します。

```
EXTRACT ext_1
END 2011-01-09 14:12:20
USERID ogg@ora1.ora, PASSWORD &
    AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC, &
    AES128 KEYNAME mykey1
GETUPDATEBEFORES
NOCOMPRESSDELETES
RMTHOST sysb, MGRPORT 8040
RMTTRAIL /home/ggs/dirdat/in
TABLE tcustmer;
TABLE tcustord;
```

表 42　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>	REPLICAT <group> では、GGSCI で作成する Replicat プロセスを指定します。
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 42　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
[TARGETDB <dsn>,] [USERID <user id> [, PASSWORD <password> [<encryption options>]]] ◆ 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と暗号化オプションを指定します。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 ...	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、173 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、Replicat パラメータ・ファイルの例です。

```
REPLICAT rep_1
END RUNTIME
USERID ogg@ora1.ora, PASSWORD &
    AACAAAAAAAAAAAJAUEUGODSCVGJEEIUGKJDJTFNDKEJFFFTC, &
    AES128 KEYNAME mykey1
ASSUMETARGETDEFS
MAP tcustmer, TARGET tcustmer;
```

反転処理のためのプロセス・グループおよび証跡の作成

プロセス・グループを作成して取消し手順を実行するには、次の要素を作成します。

- 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 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 100 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 (184 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 100 です)。

リバース・ユーティリティの実行

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
サポートされるキャラクタ・セット

Oracle GoldenGate では、次のキャラクタ・セットがサポートされます。キャラクタ・セットを指定する必要がある場合、実際のキャラクタ・セット名ではなく、記載されている識別子を Oracle GoldenGate のパラメータまたはコマンドに使用してください。現在 Oracle GoldenGate には、データベース固有のキャラクタ・セットを指定する機能は用意されていません。

パラメータ・ファイルで使用する識別子およびコマンド	キャラクタ・セット
UTF-8	ISO-10646 UTF-8、サロゲート・ペアは 1 文字当たり 4 バイト
UTF-16	ISO-10646 UTF-16
UTF-16BE	UTF-16 ビッグ・エンディアン
UTF-16LE	UTF-16 リトル・エンディアン
UTF-32	ISO-10646 UTF-32
UTF-32BE	UTF-32 ビッグ・エンディアン
UTF-32LE	UTF-32 リトル・エンディアン
CESU-8	UTF-8 に類似。UCS-2 に対応しており、サロゲート・ペアは 1 文字当たり 6 バイト
US-ASCII	US-ASCII、ANSI X34-1986
windows-1250	Windows 中央ヨーロッパ
windows-1251	Windows キリル文字
windows-1252	Windows Latin-1
windows-1253	Windows ギリシャ語
windows-1254	Windows トルコ語
windows-1255	Windows ヘブライ語
windows-1256	Windows アラビア語
windows-1257	Windows バルト語

パラメータ・ファイルで使用する識別子およびコマンド	キャラクタ・セット
windows-1258	Windows ベトナム語
windows-874	Windows タイ語
cp437	DOS Latin-1
ibm-720	DOS アラビア語
cp737	DOS ギリシャ語
cp775	DOS バルト語
cp850	DOS マルチリンガル
cp851	DOS Greek-1
cp852	DOS Latin-2
cp855	DOS キリル文字
cp856	DOS キリル文字 /IBM
cp857	DOS トルコ語
cp858	DOS マルチリンガル (ユーロ記号を含む)
cp860	DOS ポルトガル語
cp861	DOS アイスランド語
cp862	DOS ヘブライ語
cp863	DOS フランス語
cp864	DOS アラビア語
cp865	DOS 北欧
cp866	DOS キリル文字 /GOST 19768-87
ibm-867	DOS ヘブライ語 /IBM
cp868	DOS ウルドゥー語
cp869	DOS Greek-2

パラメータ・ファイルで使用する識別子およびコマンド	キャラクタ・セット
ISO-8859-1	ISO-8859-1 Latin-1/ 西ヨーロッパ
ISO-8859-2	ISO-8859-2 Latin-2/ 東ヨーロッパ
ISO-8859-3	ISO-8859-3 Latin-3/ 南ヨーロッパ
ISO-8859-4	ISO-8859-4 Latin-4/ 北ヨーロッパ
ISO-8859-5	ISO-8859-5 Latin/ キリル文字
ISO-8859-6	ISO-8859-6 Latin/ アラビア語
ISO-8859-7	ISO-8859-7 Latin/ ギリシャ語
ISO-8859-8	ISO-8859-8 Latin/ ヘブライ語
ISO-8859-9	ISO-8859-9 Latin-5/ トルコ語
ISO-8859-10	ISO-8859-10 Latin-6/ 北欧
ISO-8859-11	ISO-8859-11 Latin/ タイ語
ISO-8859-13	ISO-8859-13 Latin-7/ バルト語
ISO-8859-14	ISO-8859-14 Latin-8/ ケルト語
ISO-8859-15	ISO-8859-15 Latin-9/ 西ヨーロッパ
IBM037	IBM 037-1/697-1 EBCDIC、ブラジル、カナダ、オランダ、ポルトガル、米国および 037/1175 繁体字中国語
IBM01140	IBM 1140-1/695-1 EBCDIC、ブラジル、カナダ、オランダ、ポルトガル、米国および 1140/1175 繁体字中国語
IBM273	IBM 273-1/697-1 EBCDIC、オーストリア、ドイツ
IBM01141	IBM 1141-1/695-1 EBCDIC、オーストリア、ドイツ
IBM277	IBM 277-1/697-1 EBCDIC、デンマーク、ノルウェー
IBM01142	IBM 1142-1/695-1 EBCIDC、デンマーク、ノルウェー
IBM278	IBM 278-1/697-1 EBCDIC、フィンランド、スウェーデン
IBM01143	IBM 1143-1/695-1 EBCDIC、フィンランド、スウェーデン
IBM280	IBM 280-1/697-1 EBCDIC、イタリア
IBM01144	IBM 1144-1/695-1 EBCDIC、イタリア

パラメータ・ファイルで使用する識別子およびコマンド	キャラクタ・セット
IBM284	IBM 284-1/697-1 EBCDIC、ラテン・アメリカ、スペイン
IBM01145	IBM 1145-1/695-1 EBCDIC、ラテン・アメリカ、スペイン
IBM285	IBM 285-1/697-1 EBCDIC、英国
IBM01146	IBM 1146-1/695-1 EBCDIC、英国
IBM290	IBM 290 EBCDIC、日本 (英数カナ) 拡張
IBM297	IBM 297-1/697-1 EBCDIC、フランス
IBM01147	IBM 1147-1/695-1 EBCDIC、フランス
IBM420	IBM 420 EBCDIC、アラビア語圏 (2 か国語併用)
IBM424	IBM 424/941 EBCDIC、イスラエル (ヘブライ語 - Bulletin コード)
IBM500	IBM 500-1/697-1 EBCDIC、各国間共通
IBM01148	IBM 1148-1/695-1 EBCDIC 各国間共通
IBM870	IBM 870/959 EBCDIC、Latin-2 マルチリンガル
IBM871	IBM 871-1/697-1 EBCDIC アイスランド
IBM918	IBM EBCDIC コード・ページ 918、アラビア語 2
IBM1149	IBM 1149-1/695-1、EBCDIC アイスランド
IBM1047	IBM 1047/103 EBCDIC、Latin-1(オープン・システム)
ibm-803	IBM 803 EBCDIC、イスラエル (ヘブライ語 - 旧コード)
IBM875	IBM 875 EBCDIC、ギリシャ
ibm-924	IBM 924-1/1353-1 EBCDIC 各国間共通
ibm-1153	IBM 1153/1375 EBCDIC、Latin-2 マルチリンガル
ibm-1122	IBM 1122/1037 EBCDIC、エストニア
ibm-1157	IBM 1157/1391 EBCDIC、エストニア
ibm-1112	IBM 1112/1035 EBCDIC、ラトビア、リトアニア
ibm-1156	IBM 1156/1393 EBCDIC、ラトビア、リトアニア

パラメータ・ファイルで使用する識別子およびコマンド	キャラクタ・セット
ibm-4899	IBM EBCDIC コード・ページ 4899、ヘブライ語 (ユーロ記号を含む)
ibm-12712	IBM 12712 EBCDIC、ヘブライ語 (ユーロ記号を含む最大セット)
ibm-1097	IBM 1097 EBCDIC、ペルシア語
ibm-1018	IBM 1018 EBCDIC、フィンランド、スウェーデン (ISO-7)
ibm-1132	IBM 1132 EBCDIC、ラオス
ibm-1137	IBM EBCDIC コード・ページ 1137、デバナーガリ
ibm-1025	IBM 1025/1150 EBCDIC、キリル文字
ibm-1154	IBM EBCDIC コード・ページ 1154、キリル文字 (ユーロ記号を含む)
IBM1026	IBM 1026/1152 EBCDIC、Latin-5 トルコ
ibm-1155	IBM EBCDIC コード・ページ 1155、トルコ語 (ユーロ記号を含む)
ibm-1123	IBM 1123 EBCDIC、ウクライナ
ibm-1158	IBM EBCDIC コード・ページ 1158、ウクライナ語 (ユーロ記号を含む)
IBM838	IBM 838/1173 EBCDIC、タイ
ibm-1160	IBM EBCDIC コード・ページ 1160、タイ語 (ユーロ記号を含む)
ibm-1130	IBM 1130 EBCDIC、ベトナム
ibm-1164	IBM EBCDIC コード・ページ 1164、ベトナム語 (ユーロ記号を含む)
ibm-4517	IBM EBCDIC コード・ページ 4517、アラブ・フランス語
ibm-4971	IBM EBCDIC コード・ページ 4971、ギリシャ
ibm-9067	IBM EBCDIC コード・ページ 9067、ギリシャ 2005
ibm-16804	IBM EBCDIC コード・ページ 16804、アラビア語
KOI8-R	ロシア語およびキリル文字 (KOI8-R)
KOI8-U	ウクライナ語 (KOI8-U)
eucTH	EUC タイ語
ibm-1162	Windows タイ語 (ユーロ記号を含む)

パラメータ・ファイルで使用する識別子およびコマンド	キャラクタ・セット
DEC-MCS	DEC マルチリンガル
hp-roman8	HP Latin-1 Roman8
ibm-901	IBM バルト語 ISO-8 CCSID 901
ibm-902	IBM エストニア ISO-8(ユーロ記号を含む)CCSID 902
ibm-916	IBM ISO8859-8 CCSID
ibm-922	IBM エストニア ISO-8 CCSID 922
ibm-1006	IBM ウルドゥー語 ISO-8 CCSID 1006
ibm-1098	IBM ペルシア語 PC CCSID 1098
ibm-1124	ウクライナ語 ISO-8 CCSID 1124
ibm-1125	ウクライナ語 (ユーロ記号を含まない)CCSID 1125
ibm-1129	IBM ベトナム語 (ユーロ記号を含まない)CCSID 1129
ibm-1131	IBM ベラルーシ語 CCSID 1131
ibm-1133	IBM ラオ語 CCSID 1133
ibm-4909	IBM ギリシャ・ラテン語 ASCII CCSID 4909
JIS_X201	JIS X201 日本語
windows-932	Windows 日本語
windows-936	Windows 簡体字中国語
ibm-942	IBM Windows 日本語
windows-949	Windows 韓国語
windows-950	Windows 繁体字中国語
eucjis	EUC 日本語
EUC-JP	IBM/MS EUC 日本語
EUC-CN	EUC 簡体字中国語、GBK
EUC-KR	EUC 韓国語

パラメータ・ファイルで使用する識別子およびコマンド	キャラクタ・セット
EUC-TW	EUC 繁体字中国語
ibm-930	IBM 930/5026 日本語
ibm-933	IBM 933 韓国語
ibm-935	IBM 935 簡体字中国語
ibm-937	IBM 937 繁体字中国語
ibm-939	IBM 939/5035 日本語
ibm-1364	IBM 1364 韓国語
ibm-1371	IBM 1371 繁体字中国語
ibm-1388	IBM 1388 簡体字中国語
ibm-1390	IBM 1390 日本語
ibm-1399	IBM 1399 日本語
ibm-5123	IBM CCSID 5123 日本語
ibm-8482	IBM CCSID 8482 日本語
ibm-13218	IBM CCSID 13218 日本語
ibm-16684	IBM CCSID 16684 日本語
shiftjis	日本語 Shift JIS、チルド 0x8160 を U+301C にマップ
gb18030	GB-18030
GB2312	GB-2312-1980
GBK	GBK
HZ	HZ GB2312
Ibm-1381	IBM CCSID 1381 簡体字中国語
Big5	Big5、繁体字中国語
Big5-HKSCS	Big5、香港拡張
Big5-HKSCS2001	Big5、香港拡張 HKSCS-2001

パラメータ・ファイルで使用する識別子およびコマンド	キャラクタ・セット
ibm-950	IBM Big5、CCSID 950
ibm-949	CCSID 949 韓国語
ibm-949C	IBM CCSID 949 韓国語、バックスラッシュを含む
ibm-971	IBM CCSID 971 韓国語 EUC、KSC5601 1989
x-IBM1363	IBM CCSID 1363、韓国語

付録 2
サポートされるロケール

Oracle GoldenGate では、次のロケールがサポートされます。ロケールは、大 / 小文字が区別されないオブジェクト名を Oracle GoldenGate で比較する際に使用されます。

af	af_NA	af_ZA	am
am_ET	ar	ar_AE	ar_BH
ar_DZ	ar_EG	ar_IQ	ar_JO
ar_KW	ar_LB	ar_LY	ar_MA
ar_OM	ar_QA	ar_SA	ar_SD
ar_SY	ar_TN	ar_YE	as
as_IN	az	az_Cyrl	az_Cyrl_AZ
az_Latn	az_Latn_AZ	be	be_BY
bg	bg_BG	bn	bn_BD
bn_IN	ca	ca_ES	cs
cs_CZ	cy	cy_GB	da
da_DK	de	de_AT	de_BE
de_CH	de_DE	de_LI	de_LU
el	el_CY	el_GR	en
en_AU	en_BE	en_BW	en_BZ
en_CA	en_GB	en_HK	en_IE
en_IN	en_JM	en_MH	en_MT
en_NA	en_NZ	en_PH	en_PK
en_SG	en_TT	en_US	en_US_POSIX
en_VI	en_ZA	en_ZW	eo

es	es_AR	es_BO	es_CL
es_CO	es_CR	es_DO	es_EC
es_ES	es_GT	es_HN	es_MX
es_NI	es_PA	es_PE	es_PR
es_PY	es_SV	es_US	es_UY
es_VE	et	et_EE	eu
eu_ES	fa	fa_AF	fa_IR
fi	fi_FI	fo	fo_FO
fr	fr_BE	fr_CA	fr_CH
fr_FR	fr_LU	fr_MC	ga
ga_IE	gl	gl_ES	gu
gu_IN	gv	gv_GB	haw
haw_US	he	he_IL	hi
hi_IN	hr	hr_HR	hu
hu_HU	hy	hy_AM	hy_AM_REVISED
id	id_ID	is	is_IS
it	it_CH	it_IT	ja
ja_JP	ka	ka_GE	kk
kk_KZ	kl	kl_GL	km
km_KH	kn	kn_IN	ko
ko_KR	kok	kok_IN	kw
kw_GB	lt	lt_LT	lv
lv_LV	mk	mk_MK	ml
ml_IN	mr	mr_IN	ms
ms_BN	ms_MY	mt	mt_MT

nb	nb_NO	nl	nl_BE
nl_NL	nn	nn_NO	om
om_ET	om_KE	or	or_IN
pa	pa_Guru	pa_Guru_IN	pl
pl_PL	ps	ps_AF	pt
pt_BR	pt_PT	ro	ro_RO
ru	ru_RU	ru_UA	sk
sk_SK	sl	sl_SI	so
so_DJ	so_ET	so_KE	so_SO
sq	sq_AL	sr	sr_Cyrl
sr_Cyrl_BA	sr_Cyrl_ME	sr_Cyrl_RS	sr_Latn
sr_Latn_BA	sr_Latn_ME	sr_Latn_RS	sv
sv_FI	sv_SE	sw	sw_KE
sw_TZ	ta	ta_IN	te
te_IN	th	th_TH	ti
ti_ER	ti_ET	tr	tr_TR
uk	uk_UA	ur	ur_IN
ur_PK	uz	uz_Arab	uz_Arab_AF
uz_Cyrl	uz_Cyrl_UZ	uz_Latn	uz_Latn_UZ
vi	vi_VN	zh	zh_Hans
zh_Hans_CN	zh_Hans_SG	zh_Hant	zh_Hant_HK
zh_Hant_MO	zh_Hant_TW		

付録 3

Oracle GoldenGate 証跡について

この付録には、トラブルシューティング、サポートの利用、その他の目的で知っておく必要のある、Oracle GoldenGate 証跡に関する情報が記載されています。Oracle GoldenGate 証跡レコードを表示するには、ログダンプ・ユーティリティを使用してください。

証跡リカバリ・モード

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。

証跡ファイルのヘッダー・レコード

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。

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、各証跡ファイルのファイル・ヘッダーにはバージョン・インジケータが含まれています。デフォルトでは、証跡ファイルのバージョンは、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必要がある場合は、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)。

証跡のレコード形式

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。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。トークン領域には、Oracle GoldenGate ユーザーが列のマッピングおよび変換用に指定した情報が含まれます。

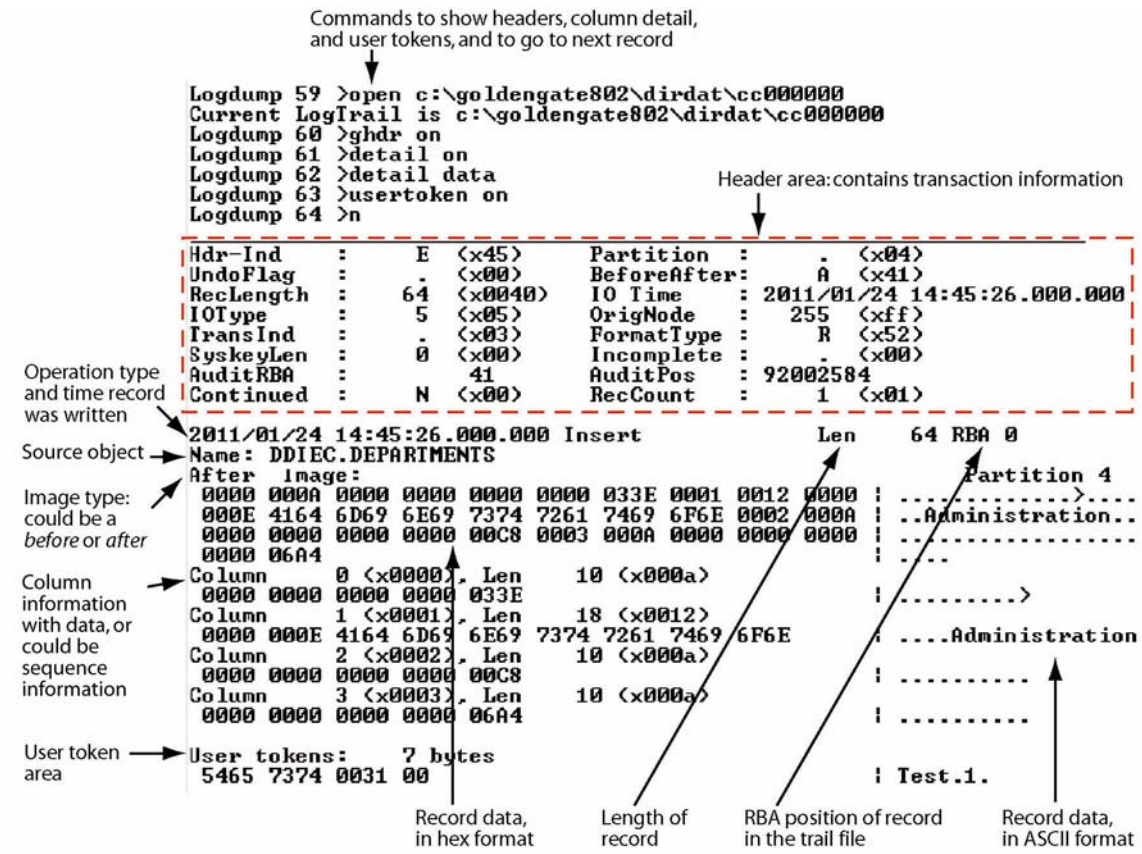
Oracle GoldenGate の証跡ファイルは構造化されていません。Oracle GoldenGate レコードは、Oracle GoldenGate ソフトウェアに付属するログダンプ・ユーティリティを使用して表示できます。詳細は、『*Windows and UNIX トラブルシューティングおよびチューニング・ガイド*』のログダンプの説明を参照してください。

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 26 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 43　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、304 ページの表 44 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	Oracle REDO ログ順序番号などのトランザクション・ログ識別子を識別します。

表 43　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 ～ 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。
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 ～ 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。
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

表 43　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
Incomplete	このフィールドは現在使用されていません。
AuditPos	データ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

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 44　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
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

表 44　Oracle GoldenGate の操作タイプ (続き)

タイプ	説明	プラットフォーム
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
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

表 44　Oracle GoldenGate の操作タイプ (続き)

タイプ	説明	プラットフォーム
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 ～ 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 44　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 ～ 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ステータス・レコード (SENDERSTATS が有効化されており、ステータス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、ステータス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。

付録 4
コミット順序番号について

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。これにより、トランザクションがデータベースにコミットされた時点を一意に識別します。

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。

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)。

MySQL ではトランザクション ID がイベント・データの一部として作成されないため、Oracle GoldenGate は次の項目の*組合せ*を一意のトランザクション識別子とみなします。

- 識別するトランザクションの START TRANSACTION レコードが含まれるログ・ファイルのログ・ファイル番号
- そのレコードのレコード・オフセット

表 45　データベースごとの Oracle GoldenGate の CSN 値

データベース	CSN 値
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

表 45　データベースごとの Oracle GoldenGate の CSN 値 (続き)

データベース	CSN 値
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
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

表 45　データベースごとの Oracle GoldenGate の CSN 値 (続き)

データベース	CSN 値

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`

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 ～ 65535 です。timestamp-low の 4 バイト整数の場合、0 ～ 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。

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。
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。

用語	定義
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。
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。

用語	定義
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。
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、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。

用語	定義
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。
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。

用語	定義
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。
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。

用語	定義
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。
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。

用語	定義
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。
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。

用語	定義
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。
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。

用語	定義
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。
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。
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。

用語	定義
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、特殊なタイプの初期ロード。
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。
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。

用語	定義
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

記号

*** ワイルドカード文字** 35, 40
@ABSENT 156
@CASE 関数 164
@COLSTAT 関数 163
@COLTEST 関数 163
@COMPUTE 関数 153, 161
@DATENOW 関数 170
@EVAL 関数 164
@GETENV 関数 159, 165, 170
@IF 関数 164
@NULL 関数 156
@NUMBIN 関数 162
@PRESENT 156
@STR* 関数 162
@TOKEN 関数 166
@VALONEOF 関数 164

数字

256 鍵バイト置換 125

A

ABEND オプション
REPERROR 168
TCP エラー 139, 171
ADD EXTRACT コマンド 186, 188, 283
ADD EXTTRAIL コマンド 190, 283, 284
ADD REPLICAT コマンド 193, 283
ADD RMTTRAIL コマンド 190, 283, 284
Advanced Encryption Security(AES) 125
ALLOWNESTED コマンド , GGSCI 27
ALTER EXTTRAIL コマンド 276
ALTER RMTTRAIL コマンド 276
ASSUMETARGETDEFS パラメータ 146
AUTORESTART パラメータ 23
AUTOSTART パラメータ 23, 197

B

BEGIN 引数 , ADD EXTRACT 187, 194, 283
BEGIN マクロ・キーワード 232
Blowfish 暗号化 125, 126
BULKLOAD パラメータ 222

C

CHARSET パラメータ 28
CHECKPARAMS パラメータ 33
CHECKPOINTTABLE オプション , ADD REPLICAT 194, 284
CMDSEC ファイル 125, 135
CMDTRACE パラメータ 238
Collector, 概要 16
COLMAP オプション , TABLE または MAP 145, 148, 225
COLMATCH パラメータ 148
COLSTAT 関数 163
COLS および COLSEXCEPT オプション , TABLE 157
COLTEST 関数 163
COMPUTE 関数 153, 161
CSN, サポートされるデータベース 18
CSN, 「*変更順序番号*」を参照
CUSEREXIT パラメータ 240
C コード・マクロ , 使用 238

D

DB2
サポートされる処理方法 11
双方向同期 97
ブートストラップ・データセット , 指定 187
DB2 for i, サポートされる処理方法 11
DBOP オプション , SQLEXEC 231

DECRYPTTRAIL パラメータ 127
DEFAULTUSERPASSWORD オプション , DDLOPTIONS 132
DEFAULT オプション
ENCRYPTKEY 131, 132
REPERROR 167
DEFERAPPLYINTERVAL パラメータ 195
DEFGEN 174
DEFGEN の NOEXTATTR オプション 178
DEFGEN の UPDATECS オプション 176
DEFSFILE パラメータ 177
DELETE EXTRACT コマンド 269, 275
DESC オプション , ADD EXTRACT 188
DISCARDFILE パラメータ 258
DISCARDROLLOVER パラメータ 258
DISCARD オプション
EVENTACTIONS 245
REPERROR 168
DSOPTIONS パラメータ 192
DYNAMICPORTLIST パラメータ 22, 215

E

EDIT PARAMS コマンド 31
EMSCLNT 259
ENCKEYS ファイル 135
ENCRYPTKEY オプション
DDLOPTIONS 132
ENCRYPT オプション , RMTHOST 133
END マクロ・キーワード 233
ERCALLBACK 関数 240
ERROR オプション , SQLEXEC 230
ER コマンド 197
EXCEPTION オプション , REPERROR 168
EXCLUDETRANS オプション , TRANLOGOPTIONS 98
EXCLUDEUSERID オプション , TRANLOGOPTIONS 98, 100, 102
EXCLUDEUSER オプション , TRANLOGOPTIONS 98
Extract
エイリアス 137, 139
エラー , 処理 167
概要 12
グループ , 追加
アクティブな構成 263
新規構成 186
実行
GGSCI から 196
データ・ポンプ , 使用 13
パッシブ 137
EXTRACT パラメータ 191
EXTRACT 引数 , ADD RMTTRAIL, ADD EXTTRAIL 190, 283
EXTRBA オプション , ADD REPLICAT 194
EXTSEQNO オプション , ADD REPLICAT 194, 284
EXTTRAILSOURCE オプション
ADD EXTRACT 187
EXTTRAIL オプション , ADD REPLICAT 194, 283

F

FastLoad, Teradata 223
FETCHBEFOREFILTER オプション , TABLE 156
FETCHCOLS オプション , TABLE 156, 230
FieldComp レコード 305
FILTERTABLE オプション , TRANLOGOPTIONS 97
FILTER 句 , TABLE または MAP 152, 225

G

GENLOADSFILE パラメータ 211
GETAPPLOPS パラメータ 97
GETDELETES パラメータ 158
GETINSERTS パラメータ 157
GETREPLICATES パラメータ 97
GETUPDATEBEFORES パラメータ 157, 158
GETUPDATES パラメータ 158
GGFILEHEADER オプション , @GETENV 298
GGHEADER オプション , GETENV 159
ggmessage.dat ファイル 172
GGS_BEFORE_AFTER_IND 列 302
GGS_OP_TYPE 列 302
GGS_TRANS_RBA 列 302

GGSCI
使用 26
セキュリティ 135
GLOBALS ファイル
作成 29
チェックポイント表と組み合せた使用 185

H

HANDLECOLLISIONS パラメータ 200
HELP コマンド 9

I

ID オプション , SQLEXEC 226
IF 関数 164
IGNORE DELETE オプション , FILTER 句 153
IGNORE INSERT オプション , FILTER 句 153
IGNORE UPDATE オプション , FILTER 句 153
IGNOREAPPLOPS パラメータ 97
IGNOREDELETES パラメータ 158
IGNOREINSERTS パラメータ 157
IGNOREREPLICATES パラメータ 97
IGNOREUPDATES パラメータ 158
IGNORE オプション
ERROR 付きの SQLEXEC 230
EVENTACTIONS 245
REPERROR 168
INCLUDE パラメータ 237
INFO コマンド 249
INSERTALLRECORDS パラメータ 158, 170
INSERTDELETES パラメータ 158
INSERTUPDATES パラメータ 158
IPv6 プロトコル 22

K

KEYCOLS オプション , TABLE または MAP 200
KeyFieldComp レコード 306
KEYGEN 134

L

LAGCRITICAL パラメータ 251
LAGINFO パラメータ 251
LAGREPORT パラメータ 251
LAG コマンド 249
LOGEND オプション , SEND EXTRACT 270

M

MACROCHAR パラメータ 233
MACRO パラメータ 232
Manager
インスタンス , 数 21
概要 16
構成および実行 21
自動起動オプション 197
統計 , 表示 250
ラグ・パラメータ 251
MAP パラメータ 141, 152
MAP 文の EXCEPTIONSONLY 168
MAP 文の MAPEXCEPTION 169
MAXVARCHARLEN オプション , SQLEXEC 231
MEGABYTES オプション , ADD RMTTRAIL, ADD EXTTRAIL 190, 283, 284
MGRPORT オプション , ADD EXTRACT 188
Microsoft SQL Server, 「SQL Server」を参照
MultiLoad, Teradata 223
MySQL, サポートされる処理方法 11

N

NODBCHECKPOINT オプション , ADD REPLICAT 194, 284
NOLIST パラメータ 237
NonStop, メッセージの送信 259
NOPARAMS オプション , SQLEXEC 226
NOPASSTHRU パラメータ 193
NULL 関数 156
NULL 値 , テスト 156, 163
NUMBIN 関数 162
NUMSTR 関数 162

O

OBEY
コマンド 27
パラメータ 29, 35
ON DELETE オプション , FILTER 句 153

ON INSERT オプション , FILTER 句 153
ON UPDATE オプション , FILTER 句 153
Oracle
SQL*Loader 初期ロード 209, 219
サポートされる処理方法 11
パスワード , 暗号化 130
Oracle GoldenGate
概要およびサポートされるデータベース 10
制御 197
変換関数 152, 164
メッセージ・ファイル 172
ユーザー・インタフェース 26
レコード形式 297, 298
Oracle GoldenGate の制御 26, 28, 238
ORACLE_SID, 変更 275

P

PARAMS オプション
ADD EXTRACT 33, 188
ADD REPLICAT 33, 194
MACRO 232
SQLEXEC 226
VAM 193
PASSIVE オプション , ADD EXTRACT 139, 187
PASSTHRU パラメータ 193
peer-to-peer 構成 , 作成 93
PORT オプション , ADD EXTRACT 188
PURGEOLDEXTRACTS パラメータ 23

Q

QUERY 句 , SQLEXEC 226

R

RAISEERROR オプション , FILTER 153
REDO スレッド
指定 187
変更 274
REPERROR パラメータ 167
Replicat
エラー , 処理 167
概要 14
グループ , 追加 193, 268
実行
GGSCI から 196
トランザクション
無視 100, 102
トランザクション , 遅延 195
トランザクション名 98
REPLICAT パラメータ 195
REPORTFILE オプション , ADD または SEND コマンド 256
REPORTROLLOVER パラメータ 257
REPORT オプション
ADD EXTRACT 188
ADD REPLICAT 194
SEND コマンド 256
REPORT パラメータ 256
RESET オプション , REPERROR 168
RMTHOSTOPTIONS パラメータ 139
RMTHOST オプション , ADD EXTRACT 139, 188
RMTTRAIL パラメータ 192

S

SEND コマンド 250
SERVLOG 260
SET EDITOR コマンド 32
SOURCEDB パラメータ 231
SOURCEDEFS パラメータ 179
SOURCEISTABLE パラメータ 204, 210, 215, 220
SPECIALRUN オプション , ADD REPLICAT 216, 221
SPECIALRUN パラメータ
Replicat ロード 206
SQL Server
アクティブ / アクティブ・サポート 93
サポートされる処理方法 11
双方向同期 98
バルク初期ロード 209
SQL*Loader へのダイレクト・バルク・ロード 219
SQL/MX, サポートされる処理方法 11
SQLEXEC パラメータ 225
STARTUPVALIDATIONDELAY パラメータ 23
STATS コマンド 249
STATUS コマンド 249
STR* 関数 162

Sybase, サポートされる処理方法 11
syslog, Oracle GoldenGate メッセージ 258
SYSLOG パラメータ 259

T

TABLE パラメータ 141, 146, 152
TARGETDB パラメータ 231
TCP/IP
エラー処理 171
使用 16
データ暗号化 133
不安定なネットワーク用の計画 13
TCPSOURCETIMER パラメータ 259
Teradata
サポートされる処理方法 11
ロード・ユーティリティ , 使用 223
THREADS オプション , ADD EXTRACT 187
TimesTen, サポートされる処理方法 11
TOKENS オプション , TABLE 165
TRANLOG オプション
ADD EXTRACT 187
TRANSABORT オプション , REPERROR 168

U

UpdateComp レコード 305
UPDATEDELETES パラメータ 158
USEDEFAULTS オプション , TABLE または MAP 147
USEIPV6 パラメータ 23
USERID パラメータ 23
usrdecs.h ファイル 239

V

VALONEOF 関数 164
VAMTRAILSOURCE オプション , ADD EXTRACT 187
VAM オプション , ADD EXTRACT 187
VAM パラメータ 193
VIEW GGSEVT コマンド 255
VIEW PARAMS コマンド 33

W

WARNRATE パラメータ 257
WHERE 句
レコード選択 155
WILDCARDRESOLVE パラメータ 271
Windows CP932 176

ア

アーカイブ・ログ , 消去 276
アーキテクチャ , Oracle GoldenGate 11
アクション , 処理中の起動 244
アクティブ / アクティブ構成 , 作成 93
アスタリスク・ワイルドカード文字 35, 40
値
NULL, テスト 156
フィルタでの使用可能性の確保 156
変更前と変更後の比較 157
無効 , NULL, 欠落 163
列での変換 162
暗号化
IDENTIFIED BY のパスワード 132
データ 125
パスワード 130

イ

イグジット・ルーチン , 使用 238
イベント
監視 249
処理 167
処理中の起動 244
イベントおよびエラーの監視 249
イベント ビューア , Oracle GoldenGate メッセージ 258
イベント・マーカー・システム 244
イベント・レコード 244

ウ

上書きリカバリ・モード 15

エ

エイリアス Extract 137
エディタ , 変更 32

エラー
SQL 257
TCP/IP 171
処理 167
処理中
ストアド・プロシージャ 230
双方向同期 104
プロセス 255
レスポンス・オプション 167

オ

大 / 小文字の区別
CMDSEC の名前 136
トークン名 165
パラメータ宣言 36
マクロ文 233
オンライン処理
概要 17
構成 183
変更 261
オンライン・ヘルプ , 取得 8

カ

鍵
暗号化 134
カスケード同期 , 構成 56
カスタム・プログラミング , 使用 225
環境
情報 , 取得 165
パラメータ・ファイルの変数 35
関数
ユーザー・イグジット 239
列変換 141

キ

キー
主 , 競合解決 94
データベース生成による値 79, 95
機密データ , 除外 157
キャラクタ・セットの維持 19
キャラクタ・セットのサポート 19
競合解決 104
行
初期ロードのプロセス間での分割 200
すべて挿入 158
選択および除外 152

ク

クラスタ , Manager の実行 24
グループ
概要 17
削除 198
追加 186, 262
グローバリゼーション・サポート 19
グローバル・パラメータ 30
グローバル列マッピング 148

ケ

警告
処理中のイベント・アクションとして 246
表示 254
フィルタ時の欠落列 156
計算 , 算術 157
継続的な変更同期 183
言語サポート 19

コ

高可用性 , 計画 77, 93
更新
圧縮 156
欠落値 , フェッチ 156, 230
削除からの作成 158
挿入への変換 158
同時 104
前の状態への変更 278

構成
Manager 23
アクティブ / アクティブ (双方向) 93
初期データ・ロード 199
セキュリティ
GGSCI コマンド 135
データ 126, 133
パスワード 130
ソース定義 177
データ・ウェアハウス (多対 1) 70
データ分散 (1 対多) 64
変更データの同期 183
ライブ・スタンバイ 77
レポート , カスケード 56
レポート , ソースでのデータ・ポンプの使用 48
レポート , 中間システムでのデータ・ポンプの使用 51
レポート , 標準 46
構文 , パラメータ・ファイルでの検証 33
コピー・ユーティリティ , 初期ロード 202
コマンド
GGSCI 26
自動化 27
データベース 225
認可 135
コミット順序番号 (CSN), 概要 18
コメント
パラメータ・ファイル 32

サ

削除 , 変換
挿入または更新 158
反転処理中の挿入 278
作成
暗号化鍵 134
証跡 189
初期チェックポイント 187, 194, 283
ソース定義ファイル 174
パラメータ・ファイル 31
ユーザー・イグジット 239
「追加」も参照
サプリメンタル・ロギング , 属性の変更 272
算術演算
WHERE 句 155
変換中 161
変更前の値の使用 157
ユーザー・イグジット 238
残高 , 計算 157

シ

シェル・スクリプト , 起動 28
システム・メンテナンス , 実行 270
集中型のレポート 70
証跡
暗号化 126
概要 14
形式 297
形式 , 指定 297
形式およびプロパティ , 返却 298
作成 189
トークン , ユーザー 165
バージョン 297
バージョン , 指定 297
ファイル・サイズ , 変更 276
レコード形式 297, 298
証跡のトークン領域 303
初期データ・ロード
Oracle GoldenGate ダイレクト・ロードの使用 214
SQL*Loader へのダイレクト・バルク・ロードの使用 219
Teradata ロード・ユーティリティの使用 223
概要 199
データベース・ユーティリティの使用 202
ファイルから Replicat へ 203
ファイルからデータベース・ユーティリティへ 209
循環レプリケーション 96
除外
Replicat トランザクション 96
行 152
取得からのトランザクション 100, 102
列 157

ス

数値
比較 156
マッピングおよび変換 151, 162
スキーマ , 変更 270
スクリプト , バッチおよびシェル 28
ストアド・プロシージャ , 使用 225
スループット , ターゲットへのデータ 252
スレッド , 数の変更 274

セ

正規形式 , 証跡データ 15
静的 Collector 17
セカンダリ Extract プロセス 13
セキュリティ
GGSCI コマンド 135
機密データ , 除外 157
データおよびパスワード 125
接続 , ネットワーク,「**ネットワーク**」を参照
選択
行 152
ストアド・プロシージャおよび問合せの使用 225
操作 157
列 157

ソ

操作 , SQL
選択および変換 157
統計 , 表示 249, 252
履歴 158
「トランザクション」も参照
挿入
削除への変更 278
削除または更新からの作成 158
例外表 170
双方向構成 , 作成 93
ソース・システム
証跡 13
ソース定義ファイル , 作成 174
ソース・データベース
属性 , 変更 270
トランザクション履歴 158
同期
中央ターゲットの使用 13
複数のターゲットの使用 14
別のソース・データベースの使用 93
ソース表
初期ロード中にアクティブ 199
データ定義 , 作成 174
速度 , 処理 253

タ

ターゲット・システム
数 64
接続 , 開始 137
ターゲット定義ファイル , 作成 174, 175
ターゲット表
移入 199
すべてのレコードの挿入 158
変更の取消し 278
「表」も参照
待機時間
監視 251
表示 249
タイムスタンプ
調整による他のシステムとの一致 259
マッピング 152
タスク , 概要 17
ダイレクト・ロード , Oracle GoldenGate 214

チ

チェックポイント
概要 15
初期 , 作成 187, 194, 283
チェックポイント表
Extract への指定 97
使用 184
遅延
Replicat トランザクション 195
抽出証跡,「**証跡**」を参照
抽出ファイル , 概要 15

ツ

追加
Extract グループ 186, 188, 263, 283
Replicat グループ 193, 268, 283
証跡 190
チェックポイント表 184
抽出するオブジェクト 270
パラメータ 34
「*作成*」も参照
追加リカバリ・モード 15

テ

定義 , 生成 174
定義テンプレート , 使用 176
テキスト・エディタ , 変更 32
テスト
NULL 値 156
データ 155, 164
列のステータス 163
列の存在 156
テンプレート , 定義 176
データ
暗号化 125
格納 13
抽出,「データの抽出」を参照
フィルタリング 152
マッピングおよび操作 141, 225, 238
ループ , 防止 100, 102
レプリケート,「データのレプリケート」を参照
ロード 199
データ・ウェアハウス構成 , 作成 70
データ型
変換 159
マッピング 151
データソース , 説明 12
データ定義ファイル , 作成 173
データの抽出
概要 12
証跡から 13
初期ロード 199
データの変換 141
Oracle GoldenGate 変換関数の使用 159
複数のステージ 13
ユーザー・イグジットの使用 238
データのレプリケート
概要 14
初期ロード 199
双方向 96
変更同期 183
データのロード
Oracle GoldenGate ダイレクト・ロードの使用 214
SQL*Loader へのダイレクト・バルク・ロードの使用 219
データベース・ユーティリティの使用 202
ファイルから Replicat へ 203
ファイルからデータベース・ユーティリティへ 209
データ分散構成 , 作成 64
データベース
コマンド , Oracle GoldenGate からの実行 225
サポートされるタイプ 11
属性 , 変更 270
パスワード , 暗号化 130
プロシージャおよび問合せ , 使用 225
データ・ポンプ
概要 13
追加 187, 265
パススルー・モード 13
複数ターゲット構成 64
データ・ループ , 防止 96
デュアルスタックの IPv6 22

ト

問合せ , Oracle GoldenGate を通じた実行 225
統計
処理された操作 249
実行時 254
プロセス用の表示 167, 249
統合同期 , 計画 70
トークン , ユーザー 165
特別実行 17
トラブルシューティング,**「問題解決」を参照**

トランザクション
Replicat, 識別および無視 96
証跡でのスキップ 197
ソースからの識別 18
抽出の防止 96
無視 100, 102
履歴 158
トランザクション・ログ
初期化 269
データソースとして 187
トランザクション・ログの初期化 269
同期
初期ロード 199

ナ

名前
ワイルドカードの使用 40

ニ

日本語のローカライズ 176

ネ

ネットワーク
信頼できるゾーン構成 137
通信 , 構成 21
データ暗号化 133
不安定 13

ハ

廃棄ファイル 257
バージョン , 証跡または抽出ファイル 297
バージョン , 表示 297
バイト置換による暗号化 125
バッチ・スクリプト 28
バルク・データ・ロード 219
パススルー・データ・ポンプ 13
パスワード
DEFGEN 177
Extract 191
Replicat 196
暗号化 130
パッシブ Extract 137
パッチ , アプリケーション 261
パラメータ
構文の検証 33
使用 28
使用場所
SQL プロシージャおよび問合せ 226
マクロ 234
実行時の置換 35
別のファイルからの取得 35
パラメータ値の置換 35
パラメータ・ファイル
DEFGEN 177
GLOBALS 29
Manager 23
作成および管理 28
初期ロード
Replicat ロード 204, 206
ダイレクト・バルク・ロード 220, 222
ダイレクト・ロード 216, 217
バルク・ロード 210, 211
変更同期
オンライン抽出 191
オンライン・レプリケーション 195
リバース・ユーティリティ 279
パラメータ・ファイルの検証 33

ヒ

比較
変更前の値と変更後の値 157
列値 156
日付
変換 161
マッピング 152

表
DB2, 再編成 276
異なる表のマッピング 141
削除および再作成 274
ソース・データベースへの追加 270
チェックポイント
Extract への指定 97
作成 184
変更の同期 183
例外 170
「ソース表」および「ターゲット表」も参照
表示
暗号化ファイル 135
エラーおよび統計 167, 249
コマンド権限 135
パラメータ 33
マクロ展開 238

フ

ファイル
CMDSEC 125, 135
ENCKEYS 135
ggserr.log 254
GLOBALS 29
usrdecs.h 239
証跡 , 概要 14
抽出 15
データ定義 174
廃棄 257
パラメータ 28
ヘッダーおよびリリース 297
フィールド , 比較 156
フィールド変換関数 141
フィルタのための列値のフェッチ 156, 230
フィルタリング
DML 操作タイプ 157
データ 152
トランザクション 100, 102
フェイルオーバー構成 , 作成 77
プロシージャ,「ストアド・プロシージャ」を参照
プロセス , Oracle GoldenGate
監視および統計 167, 249
構成 17
パラレル 14, 262

ヘ

ヘッダー , ファイル 297
ヘッダー , レコード
概要 298
説明 300
ユーザー・トークン領域 165
変換関数 , Oracle GoldenGate 152, 164
変更
Oracle GoldenGate プロセスの構成 262
TCP/IP エラーの処理 171
証跡ファイル , サイズ 276
テキスト・エディタ 32
データ構造 159
データベース・オブジェクト 270
パラメータ 34
ファイル形式 297
マクロ文字 233
変更順序番号
概要 309
変更前の値 , 使用 157, 278
編集
CMDSEC ファイル 125, 135
ENCKEYS ファイル 135
パラメータ・ファイル 34
「変更」も参照

ホ

ホット・バックアップ , 初期ロード 202
ポート番号
構成 21
動的リスト 22

マ

マクロ
作成 232
実行 233
他のマクロからの起動 236
展開のトレース 238
ネーミング 233
パラメータの使用 234
ライブラリ 236
レポート・ファイルからの除外 237
マッピング
行 152
データ型 151
マクロの使用 234
ユーザー・トークン 165
列 145

メ

メッセージ , 表示 254

モ

文字
操作 162
比較 156
ワイルドカードとの照合 40
文字 , マクロ 233
文字列
比較および変換 162

ユ

ユーザー
Oracle GoldenGate のインタフェース 26
コマンドへのアクセス , 制御 135
除外 100, 102
トランザクション , 無視 100, 102
ユーザー・イグジット , 使用 238
ユーザー・イグジットのコールバック・ルーチン 240
ユーティリティ
KEYGEN 134

ラ

ラージ・オブジェクト , 制限 141
ライブ・スタンバイ構成 , 作成 77
ライブラリ , マクロ 236
ライブ・レポート , 構成 44
ラグ
監視 251
ハートビート表を使用した分析 246
パラレル・グループの数を決定するための見積り 183

リ

リカバリ・モード , 概要 15
リバース・ユーティリティ 278
リモート証跡,「証跡」を参照
量統計 , 取得 252

ル

ループ , 防止 96

レ

例外処理 , 構成 168
レコード , 証跡
概要 298
形式 297, 298
列
NULL または欠落 163
使用可能性 , 確保 156
選択および除外 157
テストおよび変換 156, 159
表への追加 270
フィルタのためのフェッチ 156, 230
マッピング 145, 234
列変換関数 145, 152
レポート
Extract 処理 188
Replicat 処理 194
SQLEXEC 処理中のエラー 230
パラメータ・ファイルのテスト 33
プロセス・イベントおよびエラー 254, 255

レポート , プロセス
使用 255
マクロの除外 237
レポート構成 , 作成 44

ロ

ローカル証跡, 「証跡」を参照
ログ
エラー 254
プロセス 255
ログイン
Extract, 指定 191
Replicat, 指定 196
セキュリティ 130, 135

ワ

ワイルドカード
コマンド 26
コマンド・セキュリティ・ファイル 135
使用 40
表の追加時 271