Oracle9*i*

アドバンスト・レプリケーション

リリース 2（9.2）

2002 年 7 月

部品番号：J06280-01

ORACLE®

Oracle9*i* アドバンスト・レプリケーション , リリース 2（9.2）

部品番号：J06280-01

原本名：Oracle9*i* Advanced Replication, Release 2 (9.2)

原本部品番号：A96567-01

原本著者：Ted Burroughs

原本協力者：Randy Urbano, Valarie Moore, Nimar Arora, Sukanya Balaraman, Ruth Baylis, Yuen Chan, Al Demers, Alan Downing, Curt Elsbernd, Yong Feng, Jairaj Galagali, Lewis Kaplan, Jonathan Klein, Anand Lakshminath, Jing Liu, Edwina Lu, Pat McElroy, Maria Pratt, Arvind Rajaram, Neeraj Shodhan, Wayne Smith, Jim Stamos, Janet Stern, Mahesh Subramaniam, Lik Wong, David Zhang

Copyright © 1996, 2002 Oracle Corporation. All rights reserved.

Printed in Japan.

制限付権利の説明

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

当プログラムのリバース・エンジニアリング等は禁止されております。

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

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

危険な用途への使用について

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

Restricted Rights Notice

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

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

目次

はじめに vii

Advanced Replication の新機能 xv

1 Advanced Replication の概要

レプリケーションの概要 1-2
レプリケーションを使用するアプリケーション 1-3
レプリケーションのオブジェクト、グループおよびサイト 1-4
レプリケーション・オブジェクト 1-4
レプリケーション・グループ 1-4
レプリケーション・サイト 1-5
レプリケーション環境のタイプ 1-6
マルチマスター・レプリケーション 1-6
マテリアライズド・ビュー・レプリケーション 1-8
マルチマスターとマテリアライズド・ビューのハイブリッド構成 1-14
レプリケーション環境の管理ツール 1-15
Oracle Enterprise Manager のレプリケーション・マネージメント・ツール 1-16
レプリケーション・マネージメント API 1-17
レプリケーション・カタログ 1-17
分散スキーマ管理 1-17
レプリケーションの競合 1-18
マルチマスター・レプリケーションのその他のオプション 1-19
同期レプリケーション 1-19
プロシージャ・レプリケーション 1-19

2 マスター・レプリケーションの概要とアーキテクチャ

マスター・レプリケーションの概念 2-2
マスター・レプリケーションについて 2-2
マルチマスター・レプリケーションの使用目的 2-4
マルチマスター・レプリケーション・プロセス 2-7
競合解消の概念 2-10
ユーザー定義型のレプリケーション 2-12
マスター・レプリケーションのアーキテクチャ 2-18
マスター・サイトのメカニズム 2-18
管理メカニズム 2-30
編成メカニズム 2-35
伝播メカニズム 2-38
パフォーマンス・メカニズム 2-44
レプリケーションの保護メカニズム 2-49
競合解消メカニズム 2-53

3 マテリアライズド・ビューの概要とアーキテクチャ

マテリアライズド・ビューの概要 3-2
マテリアライズド・ビューについて 3-2
マテリアライズド・ビューの使用目的 3-3
読取り専用、更新可能および書込み可能のマテリアライズド・ビュー 3-4
使用可能なマテリアライズド・ビュー 3-7
マテリアライズド・ビューの操作に必要な権限 3-14
マテリアライズド・ビューを使用したデータのサブセット化 3-18
マテリアライズド・ビューの高速リフレッシュ機能に関する判断 3-25
多重化マテリアライズド・ビュー 3-26
ユーザー定義型を含むマテリアライズド・ビュー 3-32
マスター・サイトまたはマスター・マテリアライズド・ビュー・サイトでの
マテリアライズド・ビューの登録 3-43
マテリアライズド・ビューのアーキテクチャ 3-44
マスター・サイトおよびマスター・マテリアライズド・ビュー・サイトのメカニズム 3-46
マテリアライズド・ビュー・サイトのメカニズム 3-50
編成メカニズム 3-53
リフレッシュ処理 3-58

4 配置テンプレートの概要およびアーキテクチャ

大量配布の課題 4-2
配置テンプレートおよび大量配布の目標 4-2
Oracle 配置テンプレートの概念 4-3
配置テンプレートの要素 4-4
配置テンプレートのパッケージ化とインスタンス化 4-9
配置テンプレートのアーキテクチャ 4-13
システム表に格納されたテンプレート定義 4-14
パッケージ化およびインスタンス化のプロセス 4-15
インスタンス化の結果 4-19
配置テンプレートの設計 4-20
配置テンプレートを使用した列のサブセット化 4-21
行のサブセット化 4-22
データ・セット 4-26
設計に関するその他の考慮事項 4-27
マテリアライズド・ビュー作成のローカル制御 4-28

5 競合解消の概念およびアーキテクチャ

競合解消の概念 5-2
データ要件およびアプリケーション要件の理解 5-2
レプリケーションの競合のタイプ 5-3
競合の検出 5-4
競合解消 5-5
競合回避のテクニック 5-15
競合解消のアーキテクチャ 5-19
サポート・メカニズム 5-19
更新の競合の一般的な解消方法 5-21
更新の競合のその他の解消方法 5-24
一意性競合の解消方法 5-32
削除の競合解消方法 5-34
パフォーマンスのメカニズムと競合の解消 5-34

6 レプリケーション環境の計画

レプリケート表に関する考慮事項 6-2
主キー 6-2
外部キー 6-2
データ型に関する考慮事項 6-2
行レベルでの依存性の追跡 6-4
初期化パラメータ 6-4
マスター・サイトとマテリアライズド・ビュー・サイトの比較 6-8
マスター・サイトの利点 6-9
マテリアライズド・ビュー・サイトの利点 6-9
マテリアライズド・ビューの準備 6-10
マテリアライズド・ビュー・ログの作成 6-16
マテリアライズド・ビュー環境の作成 6-18
新規マテリアライズド・ビュー・サイトを追加するときの問題の回避 6-20
スケジュール・リンクのガイドライン 6-21
定期的なプッシュのスケジューリング 6-22
連続的なプッシュのスケジューリング 6-23
遅延トランザクション・キューのスケジュール・パージのガイドライン 6-24
定期的なパージのスケジューリング 6-25
連続的なパージのスケジューリング 6-26
シリアル伝播およびパラレル伝播 6-27
配置テンプレート 6-28
配置テンプレートをインスタンス化するマテリアライズド・ビュー・サイトの準備 6-28
競合解消 6-31
セキュリティ 6-31
耐障害性を実現するための設計 6-32
Oracle Real Application Clusters とレプリケーションの比較 6-33
耐障害性を実現するレプリケーション環境の設計 6-34
耐障害性の高いシステムの実装 6-35

7 レプリケーション・マネージメント・ツールの概要

レプリケーション・マネージメント・ツールの使用例 7-2
レプリケーション・マネージメント・ツールへのログイン 7-3
レプリケーション・マネージメント・ツールのインタフェース 7-4
ナビゲータ・ペイン 7-5
右ペイン 7-9
レプリケーション・マネージメント・ツールのウィザード 7-14
Setup Wizard 7-15
Materialized View Group Wizard 7-18
Deployment Template Wizard 7-19
Template Script Generation Wizard 7-21
Copy Template Wizard 7-23
レプリケーション環境を作成するためのフローチャート 7-24

A レプリケーションに関する問題のトラブルシューティング

データベース・リンクに関する問題の診断方法 A-2
マスター・サイトに関する問題の診断方法 A-2
新しいマスター・サイトでレプリケート・オブジェクトが作成されない場合 A-3
DDL による変更がマスター・サイトに伝播されない場合 A-3
DML による変更が他のサイトに非同期で伝播されない場合 A-4
DML をレプリケート表に適用できない場合 A-4
バルク更新および制約違反 A-4
レプリケート・オブジェクトの再作成 A-5
表のレプリケーション・サポートを生成できない場合 A-5
レプリケート・プロシージャまたはレプリケート・トリガーに関する問題 A-5
ON DELETE CASCADE および整合性制約に関する問題 A-6
遅延トランザクション・キューに関する問題の診断方法 A-6
スケジュール・リンクのジョブのチェック A-6
同期レプリケーション使用時の分散トランザクションの問題 A-6
データベース・リンク指定の不備 A-7
不適切なレプリケーション・カタログ・ビュー A-7
マテリアライズド・ビューに関連する問題の診断方法 A-7
マテリアライズド・ビュー・サイトでのレプリケート・オブジェクトの作成に関する問題 A-7
配置テンプレートのオフライン・インスタンシエーションの実行に関する問題 A-8

リフレッシュに関する問題 A-8
リフレッシュに関する問題の高度なトラブルシューティング A-10

B 列長セマンティクスと Unicode

レプリケーション・サイトおよび表の列に関する列長セマンティクス B-2
マルチマスター環境での列長セマンティクス・サポート B-3
Advanced Replication によって生成された表に対する列長セマンティクス・サポート B-3
事前作成された表に対する列長セマンティクス・サポート B-4
マテリアライズド・ビューでの列長セマンティクス・サポート B-5
ビルトインのコンテナ表を含むマテリアライズド・ビュー B-5
更新可能マテリアライズド・ビューに対する列長セマンティクス・サポート B-5
DDL 伝播と列長セマンティクス B-7
Unicode に対するレプリケーション・サポート B-7
NCLOB データ型列のレプリケーション B-8

索引

はじめに

このマニュアルでは、Oracle9*i* のレプリケーション機能について説明します。主に、Oracle9*i* Replication の概要と、レプリケーション環境の計画およびレプリケーションの問題のトラブルシューティングに関する情報を提供します。また、Oracle Enterprise Manager のレプリケーション・マネージメント・ツールの概要についても説明します。

この章の内容は、次のとおりです。

- 対象読者
- このマニュアルの構成
- 関連文書
- 表記規則

対象読者

『Oracle9*i* アドバンスト・レプリケーション』は、データベース管理者と、レプリケーション環境を開発およびメンテナンスするアプリケーション開発者を対象としています。これらの管理者およびアプリケーション開発者は、次のような作業を 1 つ以上実行します。

- レプリケーション環境の計画
- 次のタイプのレプリケーション環境の構成
  - 読取り専用マテリアライズド・ビュー
  - 更新可能マテリアライズド・ビュー
  - シングル・マスター・レプリケーション
  - マルチマスター・レプリケーション（Oracle Enterprise Edition が必要）
- 配置テンプレートを使用したマテリアライズド・ビュー環境の作成
- 競合解消の構成
- レプリケーション環境の管理
- トラブルシューティング・アクティビティの実行（必要な場合）
- ジョブ・キューの管理
- 遅延トランザクションの管理
- Oracle Enterprise Manager のレプリケーション・マネージメント・ツールを使用した、レプリケーション環境の作成、監視および管理

このマニュアルを使用するには、リレーショナル・データベースの概念、分散データベースの管理、PL/SQL（プロシージャ・レプリケーションを使用する場合）、およびアドバンスト・レプリケーション環境が実行されるオペレーティング・システムについてよく理解している必要があります。

このマニュアルの構成

このマニュアルの構成は次のとおりです。

第 1 章「Advanced Replication の概要」

Oracle9*i* を使用したレプリケーションの概念および用語について説明します。

第 2 章「マスター・レプリケーションの概要とアーキテクチャ」

マルチマスター・レプリケーションの概念およびアーキテクチャについて説明します。

第 3 章「マテリアライズド・ビューの概要とアーキテクチャ」

マテリアライズド・ビュー・レプリケーションの概念およびアーキテクチャについて説明します。この章では、マテリアライズド・ビュー環境を作成するための前提条件についても説明します。

第 4 章「配置テンプレートの概要およびアーキテクチャ」

配置テンプレートの概念およびアーキテクチャについて説明します。この章では、配置テンプレートの設計についても説明します。

第 5 章「競合解消の概念およびアーキテクチャ」

競合解消の概念およびアーキテクチャについて説明します。この章では、競合解消方法についても説明します。

第 6 章「レプリケーション環境の計画」

初期化パラメータの設定やレプリケーション環境の準備など、レプリケーション環境の計画について説明します。

第 7 章「レプリケーション・マネージメント・ツールの概要」

Oracle Replication Manager の機能の概要について説明します。このツールは、レプリケーション環境を作成、管理および監視するための Java ベースのツールです。

付録 A「レプリケーションに関する問題のトラブルシューティング」

レプリケーションに共通する問題の診断および解決方法を説明します。

付録 B「列長セマンティクスと Unicode」

列長セマンティクスおよび Unicode に対するレプリケーション・サポートに関する情報を提供します。

関連文書

詳細は、次の Oracle マニュアルおよびリソースを参照してください。

- 『Oracle9*i* レプリケーション・マネージメント API リファレンス』
- レプリケーション・マネージメント・ツールのオンライン・ヘルプ
- 『Oracle9*i* データベース概要』
- 『Oracle9*i* データベース管理者ガイド』
- 『Oracle9*i* SQL リファレンス』
- 『Oracle PL/SQL ユーザーズ・ガイドおよびリファレンス』（プロシージャ・レプリケーションを使用する場合）

特定のトピックの詳細は、Oracle9*i* マニュアルの他のマニュアルに含まれている場合があります。

このマニュアルに記載されている例の多くは、Oracle のインストール時にデフォルトでインストールされるシード・データベースのサンプル・スキーマを使用しています。これらのスキーマがどのように作成されているか、およびその使用方法については、『Oracle9*i* サンプル・スキーマ』を参照してください。

リリース・ノート、インストレーション・マニュアル、ホワイト・ペーパーまたはその他の関連文書は、OTN-J（Oracle Technology Network Japan）に接続すれば、無償でダウンロードできます。OTN-J を使用するには、オンラインでの登録が必要です。次の URL で登録できます。

`http://otn.oracle.co.jp/membership/`

OTN-J のユーザー名とパスワードを取得済みであれば、次の OTN-J Web サイトの文書セクションに直接接続できます。

`http://otn.oracle.co.jp/document/`

表記規則

このマニュアル・セットの本文とコード例に使用されている表記規則について説明します。

- 本文の表記規則
- コード例の表記規則

本文の表記規則

本文中には、特別な用語が一目でわかるように様々な表記規則が使用されています。次の表は、本文の表記規則と使用例を示しています。

表記規則	意味	例
太字	太字は、本文中に定義されている用語または用語集に含まれている用語、あるいはその両方を示します。	この句を指定する場合は、**索引構成表**を作成します。
固定幅フォントの大文字	固定幅フォントの大文字は、システムにより指定される要素を示します。この要素には、パラメータ、権限、データ型、Recovery Manager キーワード、SQL キーワード、SQL*Plus またはユーティリティ・コマンド、パッケージとメソッドの他、システム指定の列名、データベース・オブジェクトと構造体、ユーザー名、およびロールがあります。	この句は NUMBER 列に対してのみ指定できます。 BACKUP コマンドを使用すると、データベースのバックアップを作成できます。 USER_TABLES データ・ディクショナリ・ビューの TABLE_NAME 列を問い合せます。 DBMS_STATS.GENERATE_STATS プロシージャを使用します。
固定幅フォントの小文字	固定幅フォントの小文字は、実行可能ファイル、ファイル名、ディレクトリ名およびとサンプルのユーザー指定要素を示します。この要素には、コンピュータ名とデータベース名、ネット・サービス名、接続識別子の他、ユーザー指定のデータベース・オブジェクトと構造体、列名、パッケージとクラス、ユーザー名とロール、プログラム・ユニット、およびパラメータ値があります。 **注意：**一部のプログラム要素には、大文字と小文字の両方が使用されます。この場合は記載されているとおりに入力してください。	sqlplus と入力して SQL*Plus をオープンします。 パスワードは orapwd ファイルに指定されています。 /disk1/oracle/dbs ディレクトリ内で、データ・ファイルと制御ファイルのバックアップを作成します。 department_id、department_name および location_id の各列は、hr.departments 表にあります。 初期化パラメータ QUERY_REWRITE_ENABLED を true に設定します。 oe ユーザーで接続します。 これらのメソッドは、JRepUtil クラスで実装されます。

表記規則	意味	例
固定幅フォントの小文字のイタリック	固定幅フォントの小文字のイタリックは、プレースホルダまたは変数を示します。	*parallel_clause* を指定できます。 U*old_release*.SQL を実行します。*old_release* はアップグレード前にインストールしていたリリースです。

コード例の表記規則

コード例は、SQL、PL/SQL、SQL*Plus またはその他のコマンドラインを示します。次のように、固定幅フォントで、通常の本文とは区別して記載されています。

```
SELECT username FROM dba_users WHERE username = 'MIGRATE';
```

次の表は、コード例の記載上の表記規則と使用例を示しています。

表記規則	意味	例
[ ]	大カッコで囲まれている項目は、1 つ以上のオプション項目を示します。大カッコ自体は入力しないでください。	DECIMAL (*digits* [, *precision*])
{ }	中カッコで囲まれている項目は、そのうちの 1 つのみが必要であることを示します。中カッコ自体は入力しないでください。	{ENABLE | DISABLE}
|	縦線は、大カッコまたは中カッコ内の複数の選択肢を区切るために使用します。オプションのうち 1 つを入力します。縦線自体は入力しないでください。	{ENABLE | DISABLE} [COMPRESS | NOCOMPRESS]
...	水平の省略記号は、次のいずれかを示します。	
	■ 例に直接関係のないコード部分が省略されていること。	CREATE TABLE ... AS *subquery*;
	■ コードの一部が繰返し可能なこと。	SELECT *col1*, *col2*, ... , *coln* FROM employees;
. . .	垂直の省略記号は、例に直接関係のない数行のコードが省略されていることを示します。	SQL> SELECT NAME FROM V$DATAFILE; NAME ---------- /fs1/dbs/tbs_01.dbf /fs1/dbs/tbs_02.dbf . . . /fs1/dbs/tbs_09.dbf 9 rows selected.

表記規則	意味	例
その他の表記	大カッコ、中カッコ、縦線および省略記号以外の記号は、表示されているとおりに入力してください。	`acctbal NUMBER(11,2);` `acct  CONSTANT NUMBER(4) := 3;`
イタリック	イタリックの文字は、特定の値を指定する必要のあるプレースホルダまたは変数を示します。	`CONNECT SYSTEM/`*`system_password`* `DB_NAME = `*`database_name`

大文字	大文字は、システムにより指定される要素を示します。これらの用語は、ユーザー定義の用語と区別するために大文字で記載されています。大カッコで囲まれている場合を除き、記載されているとおりの順序とスペルで入力してください。ただし、この種の用語は大 / 小文字区別がないため、小文字でも入力できます。	`SELECT last_name, employee_id FROM employees;`
		`SELECT * FROM USER_TABLES;`
		`DROP TABLE hr.employees;`

小文字	小文字は、ユーザー指定のプログラム要素を示します。たとえば、表名、列名またはファイル名を示します。
	注意：一部のプログラム要素には、大文字と小文字の両方が使用されます。この場合は記載されているとおりに入力してください。
		`SELECT last_name, employee_id FROM employees;`
		`sqlplus hr/hr`
		`CREATE USER mjones IDENTIFIED BY ty3MU9;`

Advanced Replication の新機能

この章では、Oracle9*i* の新しいレプリケーション機能について説明します。現行のリリースに移行またはアップグレードするユーザーのために、前のリリースの新機能に関する情報も含まれています。

レプリケーションの新機能については、次の各項で説明します。

- Oracle9i のレプリケーションの新機能
- Oracle8i のレプリケーションの新機能
- Oracle8 のレプリケーションの新機能

Oracle9*i* のレプリケーションの新機能

Oracle9*i* のレプリケーションの機能と拡張について、次の各項で説明します。Oracle9*i* では、レプリケーション環境を静止する必要性を減らし、可用性を拡張しています。特定のデータベース管理操作を実行中でもレプリケーション・サイトをオープンしたままにしておけます。Oracle9*i* では、ユーザー定義型をレプリケートしたり、他のマテリアライズド・ビューを基にしてマテリアライズド・ビューを作成することもできます。さらに、Oracle9*i* では、高速リフレッシュできるマテリアライズド・ビューのタイプが増えました。

- **可用性の拡張**

  Advanced Replication では、マスター・グループの静止が必要な操作の数が減り、レプリケーション・データベースの可用性が拡張されています。マスター・グループ内の表では、このような管理操作を実行中にもトランザクション処理を続行できます。

  静止とは、マスター・グループのレプリケーション・アクティビティをすべて停止することです。マスター・グループの静止中は、マスター・グループ内の表に対してデータ操作言語（DML）文を実行できません。また、マスター・グループを静止する前に、遅延トランザクションをすべて伝播しておく必要があります。

  次の項では、リリース 9.0.1 で静止の必要性がなくなった操作について説明します。各操作に対応するレプリケーション・マネージメント API のプロシージャもリストされています。

  注意： Oracle Enterprise Manager のレプリケーション・マネージメント・ツールを使用する場合、現時点では、これらの操作の実行にはマスター・グループを静止する必要があります。マスター・グループを静止しない場合は、次の項にリストされているレプリケーション・マネージメント API プロシージャを使用してください。

- マスター・グループを静止しない新規マスター・サイトの追加

  マスター・グループを静止せずにマスター・グループに新規マスター・サイトを追加できます。この操作を完了するには、`DBMS_REPCAT` パッケージにある次のプロシージャを使用します。

  * `SPECIFY_NEW_MASTERS`
  * `ADD_NEW_MASTERS`
  * `RESUME_PROPAGATION_TO_MDEF`
  * `PREPARE_INSTANTIATED_MASTER`

- シングル・マスター環境での安全な変更によるマスター表の変更

  DBMS_REPCAT パッケージの ALTER_MASTER_REPOBJECT プロシージャを使用すると、シングル・マスター環境でレプリケート・オブジェクトに対して安全に変更を加えることが可能です。マルチマスター環境でこれらの変更を行うには静止が必要です。

  関連項目： これらのプロシージャの使用方法の詳細は、『Oracle9*i* レプリケーション・マネージメント API リファレンス』を参照してください。

- **ユーザー定義型のサポート**

  マルチマスター・レプリケーション環境およびマテリアライズド・ビュー・レプリケーション環境の両方で、ユーザー定義型とユーザー定義型を基にしたデータベース・オブジェクトをレプリケートできます。レプリケートできるデータベース・オブジェクトは、次のとおりです。

  – 列オブジェクトを含む表
  – オブジェクト表
  – ネストした表を含む表（ネストした表の記憶表を含む）
  – VARRAY を含む表
  – REF 列を含む表
  – 索引タイプ
  – ユーザー定義演算子

  Oracle9*i* ではマルチレベル・コレクションのレプリケーションもサポートしています。

  関連項目：

  - 2-12 ページ「ユーザー定義型のレプリケーション」
  - 3-32 ページ「ユーザー定義型を含むマテリアライズド・ビュー」

- **多重化マテリアライズド・ビュー**

  多重化マテリアライズド・ビュー機能を使用すると、マスター・サイトのマスター表ではなく他のマテリアライズド・ビューを基にしたマテリアライズド・ビューを作成できます。このようなマテリアライズド・ビューは、マスター表を基にしたマテリアライズド・ビューと同様に機能し、読取り専用または更新可能です。

  関連項目： 3-26 ページ「多重化マテリアライズド・ビュー」

- **マテリアライズド・ビューの高速リフレッシュの強化**

  リリース 9.0.1 では、パフォーマンスの向上により、マテリアライズド・ビューのリフレッシュがより高速になりました。また、リリース 9.0.1 ではマテリアライズド・ビューの高速リフレッシュ機能が拡張され、次のタイプのマテリアライズド・ビューを高速リフレッシュできるようになっています。

  - 1 対多副問合せを使用したマテリアライズド・ビュー
  - 多対多副問合せを使用したマテリアライズド・ビュー
  - 副問合せと UNION を使用したマテリアライズド・ビュー

  これには制限事項がいくつかあります。

  関連項目： 3-18 ページ「マテリアライズド・ビューを使用したデータのサブセット化」

- **レプリケーション環境でのパフォーマンスの監視**

  パフォーマンス監視のための新しい動的パフォーマンス・ビューが追加されました。

  - `V$MVREFRESH` には、現在リフレッシュ中のローカル・マテリアライズド・ビューに関する情報が含まれます。この情報は、マテリアライズド・ビュー・サイトにのみ表示されます。
  - `V$REPLPROP` には、レプリケーション・サイトで現在進行中のパラレル伝播に関する情報が含まれます。
  - `V$REPLQUEUE` には、遅延トランザクション・キューに関する統計情報が含まれます。

さらに、DEFSCHEDULE データ・ディクショナリ・ビューがパフォーマンス監視のために拡張されています。DEFSCHEDULE データ・ディクショナリ・ビューには次の列が追加されています。

- TOTAL_TXN_COUNT
- AVG_THROUGHPUT
- AVG_LATENCY
- TOTAL_BYTES_SENT
- TOTAL_BYTES_RECEIVED
- TOTAL_ROUND_TRIPS
- TOTAL_ADMIN_COUNT
- TOTAL_ERROR_COUNT
- TOTAL_SLEEP_TIME

関連項目： これらのビューの詳細は、『Oracle9*i* レプリケーション・マネージメント API リファレンス』を参照してください。

- **パラレル伝播での行レベル依存性の追跡**

トランザクション間の依存性は、データ・ブロック・レベルではなく行レベルで追跡できます。この機能により、パラレル伝播で遅延トランザクション・キューを適用するときに、より効率良く依存性を追跡して変更の順序付けができます。効率が上がることによりパフォーマンスが向上し、レプリケーション環境での拡張性が高まります。

関連項目： 2-49 ページ「レプリケーションの保護メカニズム」

- **CHAR 列長セマンティクスおよび Unicode に対するレプリケーション・サポート**

レプリケーションでは、マルチマスター環境およびマテリアライズド・ビュー環境の両方で、CHAR 列長セマンティクスと Unicode がサポートされます。

関連項目： 付録 B「列長セマンティクスと Unicode」

- **レプリケーション・マネージメント・ツールに対する拡張**

  Oracle9*i* ではレプリケーション・マネージメント・ツールが拡張され、次の機能が追加されています。

  – レポート機能

  – 再設計された「Deployment Template Wizard」と「Template Script Generation Wizard」（以前の「Offline Instantiation Wizard」）

  さらに、レプリケーション・マネージメント・ツールは Oracle9*i* の Oracle Enterprise Manager のコンソールに統合されています。

  関連項目： 第 7 章「レプリケーション・マネージメント・ツールの概要」およびレプリケーション・マネージメント・ツールのオンライン・ヘルプを参照してください。

Oracle8*i* のレプリケーションの新機能

この項に説明されている Oracle8*i*（リリース 8.1）のレプリケーション機能と拡張機能は、レプリケーションのパフォーマンスを最適化し、マテリアライズド・ビュー環境の配布とセキュリティの効率を向上させることを目的としています。次の項で説明する機能はすべて Oracle9*i* にも適用されます。

- **シングル・マスター・レプリケーション環境での可用性の拡張**

  リリース 8.1.7 以上では、シングル・マスター・レプリケーション環境でマスター・グループの静止が必要な操作の数が少なくなっています。次の各項では、シングル・マスター環境で静止が必要なくなった操作について説明します。各操作に対応するレプリケーション・マネージメント API のプロシージャもリストされています。

  注意： 可用性拡張機能はシングル・マスター・レプリケーションのみに適用され、マルチマスター環境には適用されません。

  - 全シングル・マスター・レプリケーション環境での可用性の拡張

    すべてのシングル・マスター環境で静止の必要性がなくなった操作を、次の表に示します。この表では、各操作に対応する DBMS_REPCAT パッケージのプロシージャもリストしています。さらに、リリース 8.1.7 の DBA Studio または Oracle9*i* の Oracle Enterprise Manager のコンソールでレプリケーション・マネージメント・ツールを使用する場合は、これらの操作で静止は不要です。

操作	DBMS_REPCAT のプロシージャ
削除の競合の解消方法の指定	ADD_DELETE_RESOLUTION
優先グループへのメンバーの追加	ADD_PRIORITY_*datatype*
サイト優先グループへの新規サイトの追加	ADD_SITE_PRIORITY_SITE
一意性競合の解消方法の指定	ADD_UNIQUE_RESOLUTION
更新の競合の解消方法の指定	ADD_UPDATE_RESOLUTION
指定された優先グループ・メンバーに対応付けられている優先レベルの変更	ALTER_PRIORITY
優先グループのメンバーの値の変更	ALTER_PRIORITY_*datatype*
指定されたサイトに対応付けられている優先レベルの変更	ALTER_SITE_PRIORITY
指定された優先レベルに対応付けられているサイトの変更	ALTER_SITE_PRIORITY_SITE
更新時および削除時に各マスター・サイトの古い列値をレプリケート表の非キー列と比較するかどうかの指定	COMPARE_OLD_VALUES
オブジェクトのレプリケート・オブジェクトとしての指定	CREATE_MASTER_REPOBJECT
競合解消のための優先グループの作成	DEFINE_PRIORITY_GROUP
競合解消のためのレプリケート・マスター・グループに対する新しいサイト優先グループの作成	DEFINE_SITE_PRIORITY
削除の競合の解消方法の削除	DROP_DELETE_RESOLUTION
優先グループのメンバーの削除（優先レベルによる削除）	DROP_PRIORITY
優先グループのメンバーの削除（値による削除）	DROP_PRIORITY_*datatype*
優先グループの削除	DROP_PRIORITY_GROUP
サイト優先グループの削除	DROP_SITE_PRIORITY
サイト優先グループからの指定サイト削除（名前による削除）	DROP_SITE_PRIORITY_SITE
一意性競合解消方法の削除	DROP_UNIQUE_RESOLUTION
更新の競合の解消方法の削除	DROP_UPDATE_RESOLUTION
特定のオブジェクトのレプリケーション・サポートに必要なトリガー、パッケージおよびプロシージャの生成	GENERATE_REPLICATION_SUPPORT
更新時および削除時にレプリケート表の各非キー列の古い列値を送信するかどうかの指定	SEND_OLD_VALUES

- 最小限の通信を使用しないシングル・マスター環境での可用性の拡張

  シングル・マスター環境で、どの更新可能マテリアライズド・ビューも競合の検出に最小限の通信を使用しない場合は、次の表にリストされているマスター・グループに対する特定の管理操作で静止が不要になります。この表では、各操作に対応する DBMS_REPCAT パッケージのプロシージャもリストしています。さらに、リリース 8.1.7 の DBA Studio または Oracle9*i* の Oracle Enterprise Manager のコンソールでレプリケーション・マネージメント・ツールを使用する場合は、これらの操作で静止は不要です。

  注意： 更新可能マテリアライズド・ビューのうち少なくとも 1 つで、競合の検出に最小限の通信を使用するときでも、5-35 ページの「シングル・マスター・サイト環境での静止の減少とマテリアライズド・ビュー」に説明されているアクションを実行する場合は、静止が不要です。

操作	DBMS_REPCAT のプロシージャ
既存の列グループへのメンバーの追加	ADD_GROUPED_COLUMN
空の列グループの作成	DEFINE_COLUMN_GROUP
列グループの削除	DROP_COLUMN_GROUP
列グループからのメンバーの削除	DROP_GROUPED_COLUMN
メンバーが 1 つ以上の新規列グループの作成	MAKE_COLUMN_GROUP
行レベル・レプリケーションの使用時に、表のどの列を比較するかの判断に、主キーではなく列または列グループを使用することの指定	SET_COLUMNS

関連項目： これらのプロシージャの使用方法の詳細は、『Oracle9*i* レプリケーション・マネージメント API リファレンス』を参照してください。

- **パフォーマンスの改善**

  PL/SQL レプリケーション・パッケージの内部化、およびマテリアライズド・ビュー・リフレッシュの最適化によって、レプリケーションのパフォーマンスが大幅に向上しました。

  - 内部適用パッケージ

    Oracle8 から続く取組みとして、Oracle8*i* では、より多くのレプリケーション・コードがデータベース・エンジン内に移動しました。リモート・サイトにレプリケート・トランザクションを適用するときに使用される PL/SQL 生成パッケージが内部化されました。その結果、レプリケート・トランザクションはより効率的にリモート・サイトで適用されます。また、パッケージが生成されないため、サイトをさらに短時間でインスタンス化できます。内部パッケージの改ざんが防止されており、セキュリティも高くなっています。

    関連項目： 第 2 章「マスター・レプリケーションの概要とアーキテクチャ」

  - マテリアライズド・ビューのリフレッシュの高速化

    マテリアライズド・ビューは、大規模リフレッシュ・グループをサポートするように最適化されました。副問合せマテリアライズド・ビューおよび NULL リフレッシュ（最後のリフレッシュ以降マスター表に変更がない場合）のサポートが改善されました。1 つのリフレッシュ・グループは最大 400 のマテリアライズド・ビューを含むことができ、リフレッシュ・グループ内のマテリアライズド・ビューのリフレッシュに必要なラウンドトリップ数が削減されました。

    関連項目： 第 3 章「マテリアライズド・ビューの概要とアーキテクチャ」

- **大量配布サポートの改善**

  拡大するフロント・オフィス・アプリケーション市場、特に大量配布アプリケーションに対するレプリケーション・サポートが改善されました。

  - 配置テンプレートを使用したマテリアライズド・ビュー・サイトのインスタンス化

    マテリアライズド・ビューの配置テンプレートは、オンラインまたはオフラインでインスタンス化できます。オンライン・インスタンシエーションの場合は、マテリアライズド・ビュー・サイトとターゲット・マスター・サイトが接続された状態でテンプレートをインスタンス化できます。オンライン・インスタンシエーションの利点は、データがインスタンス化実行時点のデータである点です。ただし、長時間にわたってアクティブな接続が必要となる可能性が高いという短所があります。

オフライン・インスタンシエーションの場合、DBA は、配置テンプレートと必要なデータをなんらかの記憶メディア（テープ、CD-ROM など）にパッケージ化して、マテリアライズド・ビュー・サイトに配布できます。マスター・サイトに接続する必要はなく、記憶メディアからテンプレートとデータを取り出してインスタンス化を実行できます。ユーザーは、オフライン・インスタンシエーションの完了後すぐに高速リフレッシュを実行できます。完全リフレッシュは必要ありません。オフライン・インスタンシエーションは、ネットワークに接続されていない多数のラップトップがターゲット・テンプレートをインスタンス化する大量配布環境に理想的なソリューションです。

関連項目： 4-9 ページ「配置テンプレートのパッケージ化とインスタンス化」

- マテリアライズド・ビュー配置テンプレートのパラメータ化

  配置テンプレートを使用すると、情報の大量配布が容易になり、フィールド・サービスや営業支援システムなどのアプリケーションをサポートできます。これらのテンプレートは、あるノードでインスタンス化されるマテリアライズド・ビューやその他のデータベース・オブジェクトをグループ化したものです。DBA は、これらのテンプレートを使用することで、マテリアライズド・ビュー環境を一元的にパッケージ化し、1 つ以上のサイトに簡単かつ安全にユーザー定義を配布できます。目標は、環境を一度作成した後で、テンプレートを必要なときにいつでも配置できるようにすることです。

  テンプレートのパラメータを使用することで、テンプレートを再定義せずにリモート・サイトでデータをサブセット化できます。また、テンプレートはパブリックまたはプライベートとして定義できます。パブリック・テンプレートはどのサイトでもインスタンス化できますが、プライベート・テンプレートは事前定義された認証サイトでのみインスタンス化できます。DBA は「Deployment Template Wizard」を使用して、テンプレートに追加するスキーマ・オブジェクトの選択、パラメータの選択および認可の定義を実行できます。配置テンプレートを使用すると、Oracle Enterprise Edition、Oracle Standard Edition および Oracle Personal Edition のマテリアライズド・ビュー・サイトにマテリアライズド・ビュー環境を作成できます。

  関連項目： 第 4 章「配置テンプレートの概要およびアーキテクチャ」

- 更新可能マテリアライズド・ビューの列のサブセット化

  更新可能マテリアライズド・ビューを行または列によりサブセット化できるようになりました。前のリリースでは行のサブセット化のみが可能でした。列のサブセット化により、リモート・サイトで必要な最小限のデータを配置できるため、接続時間が短縮されました。また、対応付けられたマスター・サイトが変更されても、マテリアライズド・ビュー・サイトは保護されます。マスター・サイトで列を追加してもマテリアライズド・ビュー・サイトに影響しません。また、マテリアライズド・ビュー・サイトである列を参照していない場合は、マスター・サイトでその列を削除してもマテリアライズド・ビュー・サイトに影響しません。

  関連項目：

  - 3-18 ページ「マテリアライズド・ビューを使用したデータのサブセット化」
  - 4-21 ページ「配置テンプレートを使用した列のサブセット化」
  - 4-22 ページ「行のサブセット化」

- セキュリティの改善

  マテリアライズド・ビュー・サイトでのインスタンス化に使用されるスクリプトはマスター・サイトで生成され、データへのアクセスを制御できますが、受信者とプロキシ・マテリアライズド・ビュー管理者に接続して、レプリケート・トランザクションの伝播やマテリアライズド・ビューのリフレッシュを行う必要があります。Oracle8*i* では、レプリケーションのセキュリティ・モデルが拡張され、信頼されないサイトへの権限の付与に関するセキュリティの問題点が排除されました。

  – マスター・サイトでのリモート・プロシージャ・コール（RPC）の受信者に必要なオブジェクトの権限は自動的に管理されるようになりました。信頼されないサイトには、必要なオブジェクト権限しか付与されません。

  – プロキシ・マテリアライズド・ビュー管理者は、余分な権限が付与されていなくても、レプリケーション・グループ内のオブジェクトにアクセスできるようになりました。

  関連項目：『Oracle9*i* レプリケーション・マネージメント API リファレンス』

- **レプリケーション・マネージメント・ツール**

  Oracle レプリケーション・マネージメント・ツールの機能が大幅に改善されました。主な拡張は、Oracle Enterprise Manager の新しい Java インタフェースに準拠するように再作成されたことです。レプリケーション・マネージメント・ツールは、ネットワーク上のどこからでも、また Windows プラットフォーム以外でも実行できます。レプリケーション・マネージメント・ツールは、標準の Web ブラウザを使用して実行することもできます。さらに、Oracle Enterprise Manager にはレプリケーション用のイベント・クラスも含まれており、これを使用すると、エラーやマテリアライズド・ビューのリフレッシュ不履行などを監視できます。

  Oracle8*i* のレプリケーション・マネージメント・ツールには、新たに次のウィザードが追加されました。

  – 「Deployment Template Wizard」。配置テンプレートを作成するときに使用します。

  – 「Offline Instantiation Wizard」。オフライン・インスタンシエーション用の配置テンプレートをパッケージ化するときに使用します。

  関連項目： 第 7 章「レプリケーション・マネージメント・ツールの概要」およびレプリケーション・マネージメント・ツールのオンライン・ヘルプを参照してください。

Oracle8 のレプリケーションの新機能

以降に示す Oracle8 リリース 8.0 のレプリケーション機能と拡張機能は、マルチマスター・レプリケーションのパフォーマンスを最適化し、特定のタイプのマテリアライズド・ビューのサブセットを高速リフレッシュすることを目的としています。Oracle8 では、LOB のサポートが追加されました。また、Oracle8 では管理性とセキュリティも向上しました。次の項で説明する機能はすべて Oracle8*i* および Oracle9*i* にも適用されます。

- パフォーマンスの強化

  Oracle8 では、次の新機能によってパフォーマンスが大幅に改善されました。

  - 遅延トランザクションのパラレル伝播

    データの整合性とトランザクション依存性を維持しながら、レプリケーション・トランザクション・ストリームの伝播をパラレル化することによって、スループット・パフォーマンスが大幅に向上しました。

    関連項目： 6-21 ページ「スケジュール・リンクのガイドライン」

  - レプリケーション・トリガーの内部化

    Oracle8 の古いバージョンでは外部化されていたレプリケーション・トリガーが内部化されました。内部トリガーには次の利点があります。

    * 応答時間パフォーマンスの改善
    * 処理オーバーヘッドの削減
    * 管理作業の低減

    関連項目： 第 2 章「マスター・レプリケーションの概要とアーキテクチャ」

  - データ伝播の削減

    Oracle8 では、ネットワークを介して伝播されるレプリケート・データの量が削減されました。伝播されるのは次の値のみです。

    * 更新された列の新しい値
    * 競合の検出および解消のために必要な列の古い値
    * 主キー値

これを行うには次のようにします。

* DBMS_REPCAT パッケージおよび DBMS_OFFLINE_SNAPSHOT パッケージ内の様々なプロシージャでの min_communication パラメータの使用
* DBMS_REPCAT パッケージの SEND_OLD_VALUES プロシージャおよび COMPARE_OLD_VALUES プロシージャの使用

この機能は、LOB をレプリケートするときのパフォーマンスを改善するのに特に重要です。

関連項目： 2-47 ページ「最小限の通信」

- **副問合せに基づくデータのサブセット化**

特定のタイプの副問合せを使用して定義されているマテリアライズド・ビューを高速リフレッシュできるようになりました。これにより、データのサブセットの定義とメンテナンスが容易になりました。この機能は、営業支援システムや支部の自動化などのような大量配布アプリケーションでは重要です。

関連項目： 3-18 ページ「マテリアライズド・ビューを使用したデータのサブセット化」

- **ラージ・オブジェクト・データ型（Large Object: LOB）のサポート**

Oracle8 では、次の LOB 型のレプリケーションをサポートしています。

– バイナリ・ラージ・オブジェクト（Binary Large Object: BLOB）
– キャラクタ・ラージ・オブジェクト（Character Large Object: CLOB）
– キャラクタ・ラージ・オブジェクトのグローバリゼーション・サポート（NCLOB）

注意： Oracle9*i* でのグローバリゼーション・サポートの旧称は、National Language Support です。

関連項目： マスター・サイトへのオブジェクト定義のレプリケーションの詳細は、レプリケーション・マネージメント・ツールのオンライン・ヘルプを参照してください。また、6-2 ページの「データ型に関する考慮事項」を参照してください。

- **管理方法の改善と使用方法の簡易化**

  Oracle8 では、次の機能によってデータベースの管理が簡単になりました。

  - ファイングレイン機能

    レプリケーション・マスター・グループは、他のレプリケーション・グループに影響を与えずに個々に静止できます。マスター・グループは、他のマスター・グループが静止している間も更新処理を継続できます。

    関連項目： 2-30 ページ「レプリケーションの操作モード」

  - 主キー・マテリアライズド・ビュー

    主キー・マテリアライズド・ビューを使用すると、高速リフレッシュ機能を保持しながら、マスター表を再編成できます。Oracle8 では、デフォルトで主キー・マテリアライズド・ビューを追加するとともに、引き続き `ROWID` マテリアライズド・ビューをサポートしています。

    関連項目： 3-7 ページ「使用可能なマテリアライズド・ビュー」を参照してください。また、Oracle8 以上での `ROWID` マテリアライズド・ビューの使用方法の詳細は、『Oracle9*i* データベース移行ガイド』を参照してください。

  - マスター・サイトでのマテリアライズド・ビューの登録

    Oracle8 では、マテリアライズド・ビューと対応付けられたマスター・サイトにマテリアライズド・ビューの情報が自動的に登録されます。これは、監視および分散管理に役立ちます。

    関連項目： 3-43 ページ「マスター・サイトまたはマスター・マテリアライズド・ビュー・サイトでのマテリアライズド・ビューの登録」

  - 高速リフレッシュ機能での表の再編成

    Oracle8 には、マスター・マテリアライズド・ビュー・ログの整合性を保持しながらマスター表を再編成できるユーティリティが用意されています。

    関連項目： マスター表の再編成の詳細は、『Oracle9*i* レプリケーション・マネージメント API リファレンス』を参照してください。

- エクスポート / インポートを使用したオフライン・インスタンシエーションのサポート

  エクスポート / インポートを使用したスキーマおよびデータベースのオフライン・インスタンシエーションがさらに自動化されました。

  関連項目： エクスポート / インポートを使用した新規マスター・サイトの追加方法の詳細は、『Oracle9*i* レプリケーション・マネージメント API リファレンス』を参照してください。

- 更新可能マテリアライズド・ビューでの遅延制約

  更新可能マテリアライズド・ビューで、宣言参照制約と一意性制約がサポートされるようになりました。

- パーティション表と索引

  Oracle8 では、パーティション表および索引のレプリケーションをサポートしています。マスター定義サイトの表と同一のパーティションをレプリケート表に作成する場合にこの機能を使用します。

- **システムベースのセキュリティ・モデルの強化**

  Oracle8 のシステムベースのセキュリティ・モデルでは、次の点が強化されました。

  – 1 つのシステムレベル・モデルが同期環境でも非同期環境でも同一の動作をすることにより、整合性が向上しました。

  – 受信サイトでの権限の不足によるトランザクションの失敗の可能性が低くなったため、信頼性が向上しました。

  – 1 人のユーザーが `repadmin` としても `repsys` としても動作できるようになり、リンクが単純化されました。

  – ユーザーレベル・モデルが必要なくなりました。

  – 1 人以上のマテリアライズド・ビュー所有者がマテリアライズド・ビューのリフレッシュを実行できるようになりました。

  関連項目：『Oracle9*i* レプリケーション・マネージメント API リファレンス』

- **レプリケーション・マネージメント・ツールの新機能**

  新しいレプリケーション・マネージメント・ツールには、システムを迅速に構成するのに役立つウィザードがいくつか用意されています。

  - アカウント、スキーマおよびリンクの作成を含め、マルチマスター構成を設定するための「Setup Wizard」
  - アカウント、スキーマおよびリンクの作成を含め、マテリアライズド・ビュー・サイト構成を設定するための「Setup Wizard」
  - マテリアライズド・ビュー・グループを設定するための「Materialized View Group Wizard」

  レプリケーション・マネージメント・ツールは、次のような Oracle8 レプリケーションの新機能の大部分をサポートしています。

  - 各サイトのデータベースのバージョンの自動認識と、リリース 9.x、リリース 8.x およびリリース 7.3 のデータベースを管理する動的構成
  - マテリアライズド・ビュー・ログの管理サポート
  - ジョブ、リフレッシュ・グループおよびスケジュール・リンクの日付式および間隔式の簡単な入力
  - レプリケーション・サポートの再生成が必要な表を示すフラグ（Oracle8 のデータベースの場合）
  - サイトのレプリケーション管理者およびプロパゲータの登録のサポート
  - マスター・グループのデータベース・オブジェクトを表示するデータベース・オブジェクト・フォルダ
  - すべてのマスター・サイトのマスター・グループの妥当性チェックのサポート

  関連項目： 第 7 章「レプリケーション・マネージメント・ツールの概要」およびレプリケーション・マネージメント・ツールのオンライン・ヘルプを参照してください。

1

Advanced Replication の概要

この章では、Advanced Replication に関する基本的な概念および用語について説明します。この章の内容は、次のとおりです。

- レプリケーションの概要
- レプリケーションを使用するアプリケーション
- レプリケーションのオブジェクト、グループおよびサイト
- レプリケーション環境のタイプ
- レプリケーション環境の管理ツール
- レプリケーションの競合
- マルチマスター・レプリケーションのその他のオプション

注意： Trusted Oracle を使用している環境でレプリケーションを使用する方法は、Oracle セキュリティ関連製品のドキュメントを参照してください。

レプリケーションの概要

レプリケーションとは、分散データベース・システムを構成する複数のデータベースのデータベース・オブジェクト（表など）をコピーおよびメンテナンスするプロセスです。1 つのサイトに適用された変更は、ローカルで取得および格納された後、リモートの各場所に転送され、適用されます。Advanced Replication は、Oracle サーバーの完全に統合された機能で、別の独立したサーバーではありません。

レプリケーションは、分散データベース・テクノロジを使用して複数サイト間でデータを共有する機能です。ただし、レプリケート・データベースと分散データベースは同じものではありません。分散データベースの場合は、多くの場所からデータを使用できますが、特定の表が存在するのは 1 つの場所のみです。たとえば、`employees` 表は分散データベース・システム内の `ny.world` データベースのみに含まれますが、この分散データベース・システムには `hk.world` データベースと `la.world` データベースも含まれています。レプリケーションでは、複数の場所で同一のデータを使用できます。たとえば、`employees` 表は、`ny.world`、`hk.world` および `la.world` で使用できます。

レプリケーションを使用するのは、次の理由によります。

可用性	レプリケーションを使用すると、様々な方法でデータにアクセスできるので、アプリケーションの可用性が向上します。1 つのサイトが使用できなくなった場合でも、それ以外の場所に対して問合せや更新を続行できます。つまり、レプリケーションを使用することで、フェイルオーバーによる完全な保護を確立できます。
パフォーマンス	レプリケーションを行うと、複数サイト間でアクティビティを均衡化できるので、共有データに高速にローカル・アクセスできます。各ユーザーは同時に別々のサーバーにアクセスでき、その結果全サーバーの負荷が軽減されます。また、ユーザーはアクセス・コストが最も低いレプリケーション・サイト（通常、ユーザーから地理的に最も近いサーバー）のデータを利用できます。
モバイル・コンピューティング	**マテリアライズド・ビュー**とは、ある一時点におけるターゲット表の完全または部分的なコピー（レプリカ）です。マテリアライズド・ビューを使用すると、ユーザーは中央のデータベース・サーバーに接続せずにデータベースのサブセットを操作できます。その後、中央のデータベース・サーバーとの接続が確立された時点で、必要に応じてマテリアライズド・ビューを同期化（リフレッシュ）できます。ユーザーがマテリアライズド・ビューをリフレッシュすると、マテリアライズド・ビュー・サイトのすべての変更が中央のデータベースに反映されます。また、ユーザーが接続されていない間に中央のデータベースで発生した変更が、マテリアライズド・ビューに反映されます。
ネットワーク負荷の軽減	レプリケーションを使用すると、複数の場所にデータを分散できます。したがって、アプリケーションは中央の 1 つのサーバーではなく様々な地域サーバーにアクセスできます。このように構成することで、ネットワーク負荷は大幅に軽減されます。

大量配布　　組織では、データの使用や操作の能力が高いアプリケーションを多数配置する必要性が高まってきています。Advanced Replication では、配置テンプレートを使用することで、複数のマテリアライズド・ビュー環境を迅速に作成できます。個々のニーズにあわせてマテリアライズド・ビュー環境をカスタマイズするには、変数を使用します。たとえば、営業部門の自動化のために配置テンプレートを使用できます。この場合、このテンプレートには様々な販売地域や営業担当者を示す変数を含めることができます。

レプリケーションの使用方法の詳細は、以降の章で説明します。

関連項目： 分散データベースの詳細は、『Oracle9*i* データベース管理者ガイド』を参照してください。

レプリケーションを使用するアプリケーション

レプリケーションは、要件が異なる様々なアプリケーションをサポートしています。アプリケーションの中には、比較的自律的な個別のマテリアライズド・ビュー・サイトに適しているものもあります。たとえば、営業支援システム、フィールド・サービス、小売りなどの大量配布アプリケーションでは、通常、中央のデータベース・システムと、中央のデータベースと非接続状態にあることが多い多数の小規模リモート・サイトとの間で定期的にデータを同期化する必要があります。営業担当員は、中央のデータベースに接続されているかどうかに関係なく、トランザクションを完了できる必要があります。この場合、リモート・サイトは自律的である必要があります。

これに対し、コール・センターやインターネット・システムなどのアプリケーションでは、複数のサーバー上のデータを継続的かつほぼ即時に同期化し、常に同じ内容のサービスを利用できる必要があります。たとえば、インターネット上の小売り Web サイトでは、顧客が閲覧する情報がどのサイトのオンライン・カタログでも同じであることが必要です。この場合、サイトの自律性よりもデータ整合性が重要です。

Advanced Replication は、前述の 2 種類のアプリケーションでそれぞれに使用することも、2 種類のアプリケーションを組み合せたシステムで使用することもできます。実際、1 つのアドバンスト・レプリケーション環境で、大量配布レプリケーションとサーバー / サーバー・レプリケーションの両方をサポートできるので、一貫した環境に統合することが可能です。このような環境では、たとえば、営業支援システムと顧客サービスのコールセンターの間でデータを共有できます。

Advanced Replication は、異なるリリースの Oracle を使用する環境や別々のオペレーティング・システムで Oracle を実行する環境でデータをレプリケートできます。したがって、このような環境でデータを使用するアプリケーションは、Advanced Replication を使用できます。

レプリケーションのオブジェクト、グループおよびサイト

次の項では、レプリケーション・オブジェクト、レプリケーション・グループおよびレプリケーション・サイトなど、レプリケーション・システムの基本構成要素について説明します。

レプリケーション・オブジェクト

レプリケーション・オブジェクトは、分散データベース・システム内の複数のサーバーに存在するデータベース・オブジェクトです。レプリケーション環境では、1 つのサイトのレプリケーション・オブジェクトに加えられた更新がすべて、その他の各サイトにあるコピーに適用されます。Advanced Replication を使用すると、次のタイプのオブジェクトをレプリケートできます。

- 表
- 索引
- ビューおよびオブジェクト・ビュー
- パッケージおよびその本体
- プロシージャおよびファンクション
- ユーザー定義型および型の本体
- トリガー
- シノニム
- 索引タイプ
- ユーザー定義演算子

表に関しては、レプリケーションでは高度な機能（パーティション表、索引構成表、ユーザー定義型を基にした列を含む表およびオブジェクト表など）をサポートしています。

レプリケーション・グループ

レプリケーション環境では、**レプリケーション・グループ**を使用してレプリケーション・オブジェクトを管理します。レプリケーション・グループは、論理的に関連付けられたレプリケーション・オブジェクトの集合です。

レプリケーション・グループ内に関連するデータベース・オブジェクトを編成すると、多数のオブジェクトをまとめて簡単に管理できます。通常、レプリケーション・グループを作成および使用して、特定のデータベース・アプリケーションのサポートに必要なスキーマ・オブジェクトを編成します。ただし、レプリケーション・グループおよびスキーマは、相互に対応している必要はありません。レプリケーション・グループには、複数のスキーマのオブジェクトを含めることができます。また 1 つのスキーマには、複数のレプリケーション・グループ内のオブジェクトを含めることができます。ただし、各レプリケーション・オブジェクトは、1 つのレプリケーション・グループのメンバーにしかなれません。

レプリケーション・サイト

レプリケーション・グループは、複数の**レプリケーション・サイト**に存在できます。レプリケーション環境では、2 つの基本的なタイプのサイトがサポートされます。**マスター・サイト**と**マテリアライズド・ビュー・サイト**です。1 つのサイトは、1 つのレプリケーション・グループのマスター・サイトであり、同時に別のレプリケーション・グループのマテリアライズド・ビュー・サイトにもなることができます。ただし、同一レプリケーション・グループ内のマスター・サイトとマテリアライズド・ビュー・サイトになることはできません。

マスター・サイトとマテリアライズド・ビュー・サイトの違いは次のとおりです。

- マスター・サイトのレプリケーション・グループは、**マスター・グループ**と呼ばれます。マテリアライズド・ビュー・サイトのレプリケーション・グループはマスター・グループを基にしており、**マテリアライズド・ビュー・グループ**と呼ばれます。また、すべてのマスター・グループには、**マスター定義サイト**が 1 つずつ存在します。レプリケーション・グループのマスター定義サイトは、レプリケーション・グループおよびグループ内のオブジェクトを管理する制御センターとして機能するマスター・サイトです。
- マスター・サイトでは、レプリケーション・グループ内のすべてのオブジェクトの完全なコピーがメンテナンスされます。一方、マテリアライズド・ビュー・サイトのマテリアライズド・ビューには、マスター・グループ内のすべての表データまたは表データのサブセットを含めることができます。たとえば、`hr_repg` マスター・グループに表 `employees` および `departments` が含まれている場合、マスター・グループに含まれているすべてのマスター・サイトでは、`employees` および `departments` の完全なコピーをメンテナンスする必要があります。ただし、あるマテリアライズド・ビュー・サイトには `employees` 表のマテリアライズド・ビューのみが含まれ、別のマテリアライズド・ビュー・サイトには `employees` 表と `departments` 表が両方とも含まれていることがあります。
- マルチマスター・レプリケーション環境のすべてのマスター・サイトは、相互に直接通信を行って、レプリケーション・グループのデータの変更を絶えず伝播します。マテリアライズド・ビュー・サイトには、ある一時点における表データのイメージ、すなわちマテリアライズド・ビューが含まれています。通常、マテリアライズド・ビューは、定期的にリフレッシュされてマスター・サイトと同期化されます。マテリアライズド・ビューは**リフレッシュ・グループ**に編成できます。リフレッシュ・グループ内のマテリアライズド・ビューは、1 つ以上のマテリアライズド・ビュー・グループに含めることができます。各マテリアライズド・ビューは、リフレッシュ・グループ内のすべてのマテリアライズド・ビューのデータがトランザクションの一貫性を保つ特定の時点と一致するように、同時にリフレッシュされます。

レプリケーション環境のタイプ

Advanced Replication は、次のタイプのレプリケーション環境をサポートしています。

- マルチマスター・レプリケーション
- マテリアライズド・ビュー・レプリケーション
- マルチマスターとマテリアライズド・ビューのハイブリッド構成

マルチマスター・レプリケーション

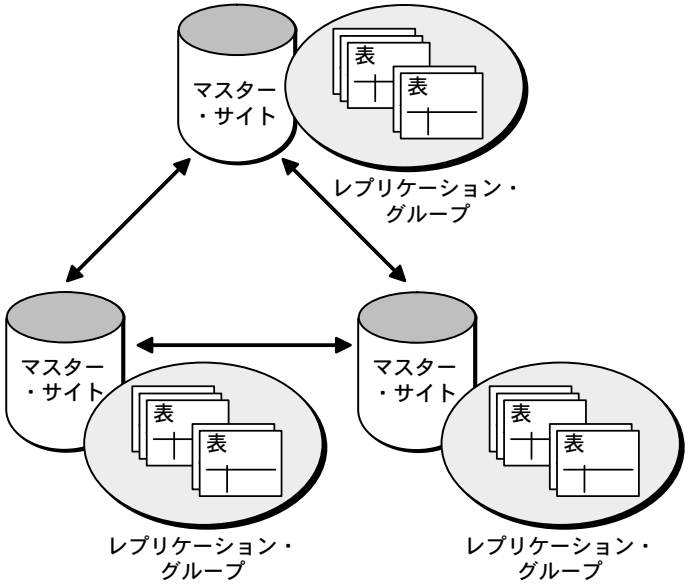
マルチマスター・レプリケーション（peer-to-peer レプリケーションまたは *n*-way レプリケーションとも呼ばれます）を使用すると、同等ピアとして機能する複数サイトでレプリケート・データベース・オブジェクトのグループを管理できます。マルチマスター・レプリケーション環境では各サイトがマスター・サイトで、各サイトが他のマスター・サイトと通信します。

アプリケーションは、マルチマスター構成のサイトにある任意のレプリケート表を更新できます。マルチマスター環境でマスター・サイトとして機能している Oracle データベース・サーバーで、すべての表レプリカのデータが自動的に収集され、グローバルなトランザクション一貫性およびデータ整合性が保証されます。

マルチマスター・レプリケーションを実装する最も一般的な方法は**非同期レプリケーション**です。その他の方法としては、同期レプリケーションおよびプロシージャ・レプリケーションがあります。これらについてはこの章で後述します。非同期レプリケーションを使用する場合、データ操作言語（DML）による表の更新に関する情報は、変更が発生したマスター・サイトの遅延トランザクション・キューに格納されます。このような変更は、**遅延トランザクション**と呼ばれます。遅延トランザクションは、関係する他のマスター・サイトに一定の間隔でプッシュ（または伝播）されます。伝播間隔は、変更できます。

非同期レプリケーションを使用すると、同じ行の値が 2 つの別のマスター・サイトでほぼ同時に更新される場合があるため、データ競合が発生する可能性があります。ただし、競合は技術によって回避することができます。競合が発生した場合は、Oracle が提供する組込みメカニズムにより問題を解決できます。解消されない競合に関する情報は、エラー・ログに格納されます。

図 1–1　マルチマスター・レプリケーション



マスター・グループの静止

マスター・グループで管理作業を実行するために、マスター・グループのレプリケーション・アクティビティをすべて停止することが必要な場合があります。たとえば、新しいマスター・グループ・オブジェクトを追加するには、マスター・グループのレプリケーション・アクティビティをすべて停止する必要があります。マスター・グループのレプリケーション・アクティビティをすべて停止することを、グループの**静止**といいます。マスター・グループの静止中は、マスター・グループ内のオブジェクトに対して DML 文を発行できません。また、マスター・グループを静止する前に、遅延トランザクションをすべて伝播しておく必要があります。ユーザーは静止中のマスター・グループ内の表に対して、問合せを引き続き実行できます。

マテリアライズド・ビュー・レプリケーション

マテリアライズド・ビューには、ある一時点におけるターゲット・**マスター**表の完全コピーまたは部分的なコピーが含まれます。ターゲット・マスターは、マスター・サイトのマスター表またはマテリアライズド・ビュー・サイトのマスター・マテリアライズド・ビューのいずれかです。**マスター・マテリアライズド・ビュー**は、別のマテリアライズド・ビューのマスターとして機能するマテリアライズド・ビューです。**多重化マテリアライズド・ビュー**は、マスター表ではなく他のマテリアライズド・ビューを基にしたマテリアライズド・ビューです。

マテリアライズド・ビューには、次の利点があります。

- ローカル・アクセスが可能なため、応答時間を短縮し、可用性を向上できます。
- ユーザーはローカルなマテリアライズド・ビューに問合せを実行できるため、マスター・サイトまたはマスター・マテリアライズド・ビュー・サイトに対する問合せ負荷が軽減されます。
- ターゲット・マスターのデータ・セットのうち、選択したサブセットのみをレプリケート可能にすることにより、データ・セキュリティを高めることができます。

マテリアライズド・ビューは、読取り専用、更新可能または書込み可能のいずれかで、このようなマテリアライズド・ビューには前述したリストにない他の利点があります。

読取り専用マテリアライズド・ビュー

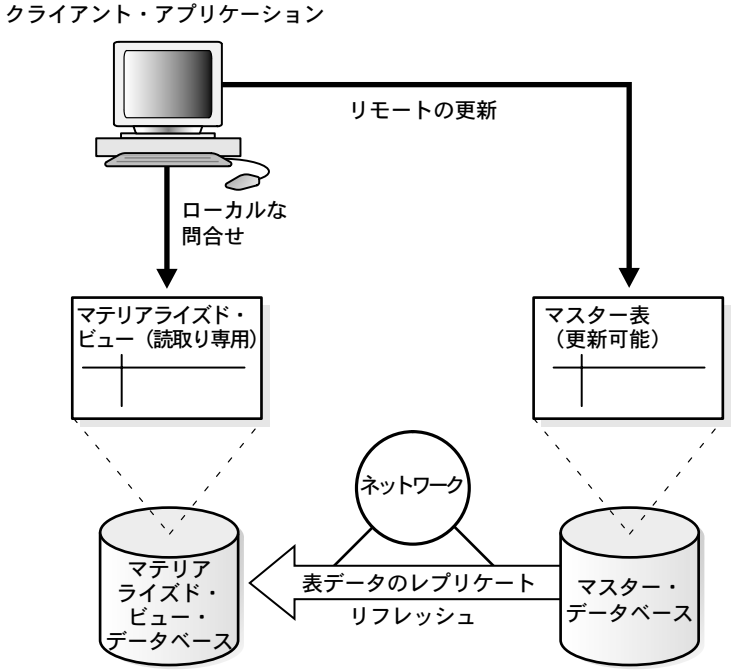
基本的な構成では、マスター・サイトまたはマスター・マテリアライズド・ビュー・サイトのデータを基に作成された表データに対して、マテリアライズド・ビューは読取り専用アクセスを提供します。アプリケーションでは**読取り専用マテリアライズド・ビュー**からデータを問い合せることにより、ネットワークが使用できるかどうかに関係なくマスター・サイトへのネットワーク・アクセスを回避できます。ただし、アプリケーションでデータ操作言語（DML）による変更を実行する場合は、システムのあらゆる場所からマスター・サイトのデータにアクセスする必要があります。図 1–2 は、基本的な読取り専用レプリケーションを示します。読取り専用マテリアライズド・ビューのマスター表およびマスター・マテリアライズド・ビューは、レプリケーション・グループに属する必要はありません。

読取り専用マテリアライズド・ビューには、次の利点があります。

- 更新できないため、競合の可能性がありません。
- 複合マテリアライズド・ビューをサポートします。複合マテリアライズド・ビューの例として、集合演算または CONNECT BY 句を含むマテリアライズド・ビューがあります。

> **関連項目：** 複合マテリアライズド・ビューの詳細は、3-7 ページの「使用可能なマテリアライズド・ビュー」を参照してください。

図 1–2　読取り専用マテリアライズド・ビューのレプリケーション

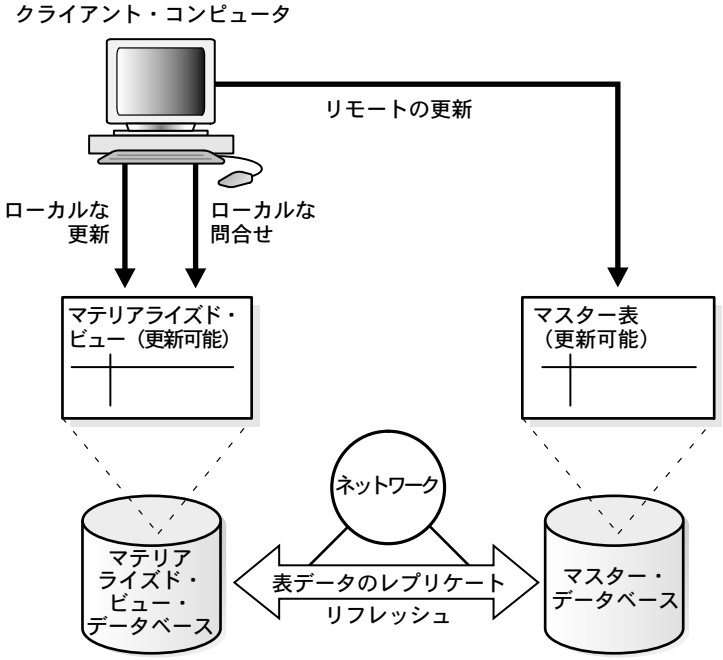


更新可能マテリアライズド・ビュー

より高度な構成では、**更新可能マテリアライズド・ビュー**を作成できます。更新可能マテリアライズド・ビューを使用すると、ユーザーはマテリアライズド・ビューに対して行の挿入、更新および削除の操作を実行することで、ターゲット・マスター表またはマスター・マテリアライズド・ビューの行を挿入、更新および削除できます。更新可能マテリアライズド・ビューには、ターゲット・マスター内のデータのサブセットも含められます。図 1–3 は、更新可能マテリアライズド・ビューを使用したレプリケーション環境を示します。

更新可能マテリアライズド・ビューは、レプリケーションをサポートするように設定されている表やその他のマテリアライズド・ビューに基づいています。実際、更新可能マテリアライズド・ビューは、別のレプリケーション・グループに基づいたマテリアライズド・ビュー・グループに含まれている必要があります。

図 1–3　更新可能マテリアライズド・ビューのレプリケーション



更新可能マテリアライズド・ビューには、次の特性があります。

- 常に 1 つの表に基づいています。ただし、副問合せで複数の表を参照できます。
- 増分リフレッシュ（または高速リフレッシュ）を実行できます。
- 更新可能マテリアライズド・ビューに加えられた変更は、マテリアライズド・ビューのリモートのマスター表またはマスター・マテリアライズド・ビューに伝播されます。次に、マスターへの更新が他のすべてのレプリケーション・サイトにカスケードして伝播されます。

更新可能マテリアライズド・ビューには、次の利点があります。

- マスター・サイトまたはマスター・マテリアライズド・ビュー・サイトに接続されていない状態でも、ローカルのレプリケート・データ・セットに対して問合せおよび更新ができます。
- データの更新をサポートしていますが、リソースはマルチマスター・レプリケーションの場合よりも少なくて済みます。マルチマスター・レプリケーションでは定期的な間隔で変更が伝播されますが、マテリアライズド・ビューは必要に応じてリフレッシュできるため、ネットワーク・リソースにかかる負荷を軽減できます。さらに、マテリアライズド・ビューは非常に少ないデータしか含まないデータベース内にも常駐させることが可能なため、マテリアライズド・ビュー・クライアントのためのディスク領域とメモリー要件は、マスター・サイトを含む Oracle サーバーに対する要件より少なくて済みます。

書込み可能マテリアライズド・ビュー

FOR UPDATE 句を使用してマテリアライズド・ビューを作成し、このマテリアライズド・ビューをマテリアライズド・ビュー・グループに追加しないことが可能です。この場合、ユーザーはこのマテリアライズド・ビューに対してデータ操作言語（DML）による変更を実行できますが、この変更はマスターにプッシュされず、マテリアライズド・ビューがリフレッシュされた場合は変更が失われます。このようなマテリアライズド・ビューは、**書込み可能マテリアライズド・ビュー**と呼ばれます。

マテリアライズド・ビューを使用した行と列のサブセット化

行と列のサブセット化により、マスター表またはマスター・マテリアライズド・ビューのデータの部分的なコピーを含むマテリアライズド・ビューを作成できます。このようなマテリアライズド・ビューは、すべてのデータ・セットを必要としない地域事業所や営業担当員に役立ちます。

行のサブセット化を使用すると、マスターの必要な行のみをマテリアライズド・ビューに含めることができます。これには WHERE 句を使用します。列のサブセット化を使用すると、マスターの必要な列のみをマテリアライズド・ビューに含めることができます。これを行うには、マテリアライズド・ビューの作成中に、SELECT 文に該当する列を指定します。

注意： 更新可能マテリアライズド・ビューの列のサブセット化は、配置テンプレートおよびレプリケーション・マネージメント・ツールを使用する場合にのみサポートされます。この制限は、読取り専用マテリアライズド・ビューの列のサブセット化にはあてはまりません。

関連項目：

- 3-18 ページ「マテリアライズド・ビューを使用したデータのサブセット化」
- 配置テンプレートを使用した列のサブセット化の詳細は、4-21 ページの「配置テンプレートを使用した列のサブセット化」を参照してください。

マテリアライズド・ビューのリフレッシュ

マテリアライズド・ビューとマスター表またはマスター・マテリアライズド・ビューとの間で一貫性を保つには、マテリアライズド・ビューを定期的に**リフレッシュ**する必要があります。マテリアライズド・ビューをリフレッシュするには、次の 3 つの方法があります。

- マテリアライズド・ビュー・ログを使用して、最後のリフレッシュ以降に変更されている行のみを更新する**高速リフレッシュ**。
- マテリアライズド・ビュー全体を更新する**完全リフレッシュ**。
- 可能なときに高速リフレッシュを実行する**強制リフレッシュ**。高速リフレッシュが実行できない場合は、完全リフレッシュが実行されます。

リフレッシュ・グループ

マテリアライズド・ビュー間のトランザクションの一貫性を保つことが重要なときは、マテリアライズド・ビューを**リフレッシュ・グループ**に編成できます。リフレッシュ・グループをリフレッシュすることで、リフレッシュ・グループ内のすべてのマテリアライズド・ビューのデータが、トランザクションの一貫性を保つ特定の時点のデータと確実に一致します。読取り専用マテリアライズド・ビューと更新可能マテリアライズド・ビューは、どちらもリフレッシュ・グループに含められます。リフレッシュ・グループ内のマテリアライズド・ビューを個別にリフレッシュすることも可能です。ただし、リフレッシュ・グループ内のその他のマテリアライズド・ビューはリフレッシュされないため、リフレッシュ・グループを編成する意味がなくなります。

マテリアライズド・ビュー・ログ

マテリアライズド・ビュー・ログは、マテリアライズド・ビューのマスター・サイトまたはマスター・マテリアライズド・ビュー・サイトにある表で、DML によってマスター表またはマスター・マテリアライズド・ビューに加えられたすべての変更を記録します。マテリアライズド・ビュー・ログは 1 つのマスター表またはマスター・マテリアライズド・ビューと対応付けられます。それぞれのマスター表またはマスター・マテリアライズド・ビューには、マスターからリフレッシュされるマテリアライズド・ビューの数に関係なく、1 つのマテリアライズド・ビュー・ログがあります。マテリアライズド・ビューの高速リフレッシュは、マテリアライズド・ビューのマスターにマテリアライズド・ビュー・ログがある場合にのみ可能です。マテリアライズド・ビューを高速リフレッシュすると、マテリアライズド・ビューと対応付けられたマテリアライズド・ビュー・ログのエントリで、最後のリフレッシュ以降に追加されたエントリがマテリアライズド・ビューに適用されます。

配置テンプレート

配置テンプレートを使用すると、多くのリモート・マテリアライズド・ビュー・サイトの配置およびメンテナンス作業が簡単になります。また、配置テンプレートを使用すると、マスター・サイトでマテリアライズド・ビュー定義の集合を定義でき、その定義中のパラメータを使用してマテリアライズド・ビューを個々のユーザーまたはユーザーのタイプにあわせてカスタマイズできます。

たとえば、営業用に 1 つのテンプレートを作成し、サービス技術員用に別のテンプレートを作成する場合などです。この場合、販売地域または顧客サポート・レベルなどのパラメータ値を使用できます。リモート・ユーザーは、マスター・サイトに接続するときに、使用可能なテンプレートのリストを問い合せることができます。ユーザーがテンプレートを**インスタンス化**すると、リモート・サイトで適切なマテリアライズド・ビューが作成され、データが移入されます。パラメータ値は、リモート・ユーザーが指定するか、マスター・サイトに保持されている表の値を利用します。

オンライン・インスタンシエーションおよびオフライン・インスタンシエーション ユーザーがマテリアライズド・ビュー・サイトでテンプレートをインスタンス化すると、オブジェクト DDL（たとえば、`CREATE MATERIALIZED VIEW` 文や `CREATE TABLE` 文）が実行されて、マテリアライズド・ビュー・サイトに適切なスキーマ・オブジェクトが作成され、そのオブジェクトに適切なデータが移入されます。テンプレートのインスタンス化は、マスター・サイトに接続した状態でネットワークを介して行うことも（オンライン・インスタンシエーション）、あるいはマスター・サイトから切断された状態で行うこともできます（オフライン・インスタンシエーション）。

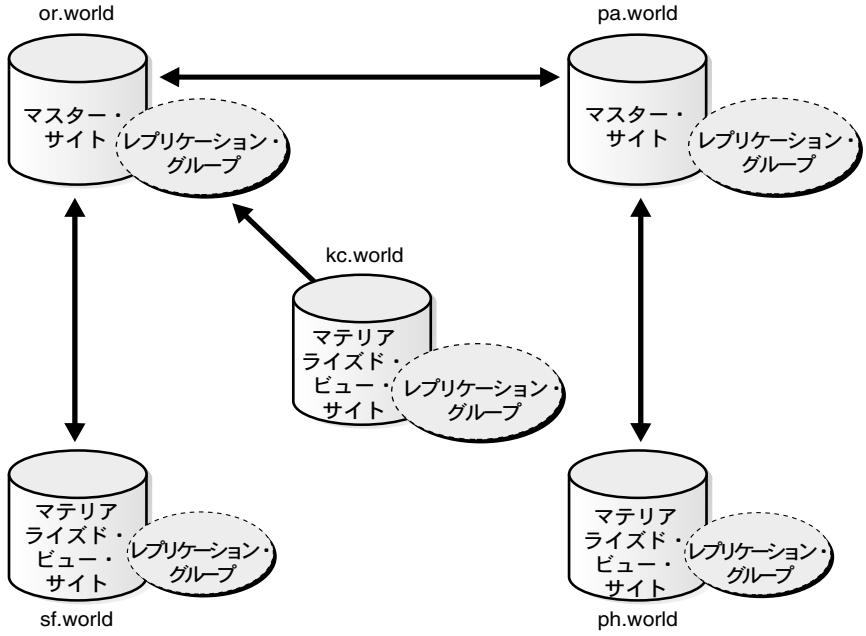
オフライン・インスタンシエーションは、ピーク期間中のサーバー負荷を軽減したり、リモート接続時間を短縮するために使用されます。テンプレートをオフラインでインスタンス化するには、テンプレートおよび必要なデータをテープや CD-ROM などの記憶メディアにパッケージ化します。そして、マスター・サイトではなく、テンプレートとデータが入った記憶メディアからデータをプルします。

マルチマスターとマテリアライズド・ビューのハイブリッド構成

マルチマスター・レプリケーションとマテリアライズド・ビューは、様々なアプリケーションの要件を満たすように**ハイブリッド**（複合）構成として組み合せられます。ハイブリッド構成では、マスター・サイトの数に制限がなく、各マスターに複数のマテリアライズド・ビュー・サイトを対応付けられます。

たとえば、図 1–4 に示すように、2 つのマスター間のマルチマスター（または *n*-way）レプリケーションでは、2 つの地域をサポートするデータベース間の全表レプリケーションをサポートできます。マテリアライズド・ビューをマスター上に定義すると、全表または表のサブセットを各地域内のサイトにレプリケートできます。

図 1–4　ハイブリッド構成



マテリアライズド・ビューとレプリケート・マスター表との主な違いは次のとおりです。

- レプリケート・マスター表にはレプリケートされている表全体のデータが含まれている必要がありますが、マテリアライズド・ビューはマスター表のデータのサブセットをレプリケートできます。
- マルチマスター・レプリケーションを使用すると、変更が発生するたびに各トランザクションの変更内容をレプリケートできます。マテリアライズド・ビューのリフレッシュを使用すると、一定の間隔でバッチ処理されるため、マルチマスター・レプリケーションより少ない頻度で複数のトランザクションの内容をより効率的に伝播できます。
- 同一データの複数のコピーに変更が加えられることにより競合が発生する場合は、競合の検出と解消は常にマスター・サイトまたはマスター・マテリアライズド・ビュー・サイトで発生します。

スケジュール・リンク

マスター・サイトとマテリアライズド・ビュー・サイトは、**スケジュール・リンク**を使用して他のサイトにデータの変更を伝播します。スケジュール・リンクは、ユーザー定義スケジュールを使用して遅延トランザクションをプッシュするデータベース・リンクです。スケジュール・リンクによって、あるマスター・サイトから別のマスター・サイトへの遅延トランザクション・キューの伝播方法、あるいは、あるマテリアライズド・ビュー・サイトからそのマスター・サイトへの遅延トランザクション・キューの伝播方法が決定されます。スケジュール・リンクを作成すると、遅延トランザクション・キューをシステムの別のサイトにプッシュするジョブがローカル・ジョブ・キューに作成されます。

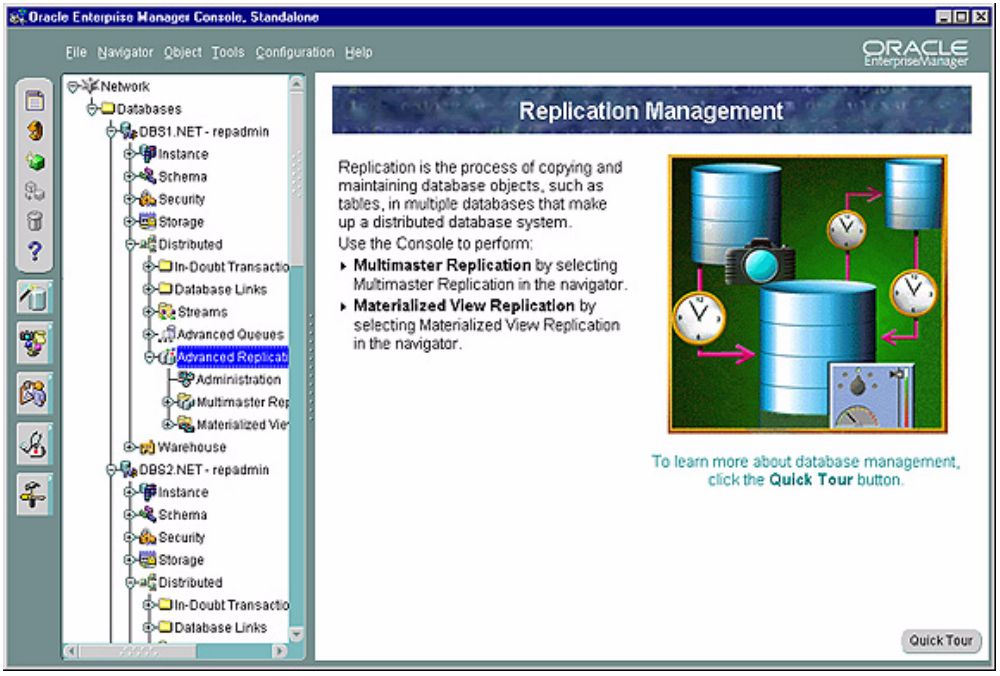
レプリケーション環境の管理ツール

レプリケーション環境を構成、管理および監視するために、いくつかのツールを使用できます。Oracle Enterprise Manager のレプリケーション・マネージメント・ツールでは、レプリケーション環境の管理に役立つ強力な GUI インタフェースが提供されます。また、レプリケーション・マネージメント API では、使い慣れた API を使用してレプリケーション管理のためのカスタマイズ・スクリプトを作成できます。レプリケーション・カタログを使用して、レプリケーション環境に関する情報を管理することもできます。

Oracle Enterprise Manager のレプリケーション・マネージメント・ツール

レプリケーション環境の構成と管理に役立つように、Oracle Enterprise Manager のコンソールには洗練されたレプリケーション・マネージメント・ツールが提供されています。このマニュアルの他の項には、このツールに関する情報と使用例が含まれていますが、このツールの詳細は、主にこのツールのオンライン・ヘルプを参照してください。

図 1–5　Oracle Enterprise Manager のレプリケーション・マネージメント・ツール



関連項目：　レプリケーション・マネージメント・ツールの概要は、第 7 章「レプリケーション・マネージメント・ツールの概要」を参照してください。レプリケーション・マネージメント・ツールの使用方法の詳細は、レプリケーション・マネージメント・ツールのオンライン・ヘルプを参照してください。

レプリケーション・マネージメント API

レプリケーション・マネージメント API は、アドバンスト・レプリケーション環境を構成する際に使用するプロシージャおよびファンクションをカプセル化した PL/SQL パッケージのセットです。レプリケーション・マネージメント API は、レプリケーション・マネージメント・ツールの機能をコマンドラインで実行します。つまり、レプリケーション・マネージメント・ツールは、レプリケーション・マネージメント API のプロシージャとファンクションを使用して処理を実行します。たとえば、レプリケーション・マネージメント・ツールを使用して新しいマスター・グループを作成する場合、このツールは DBMS_REPCAT パッケージ内の CREATE_MASTER_REPGROUP プロシージャをコールして作業を完了します。レプリケーション・マネージメント API を使用すると、レプリケーション環境を管理するカスタム・スクリプトを簡単に作成できます。

> **関連項目：** レプリケーション・マネージメント API の使用方法の詳細は、『Oracle9*i* レプリケーション・マネージメント API リファレンス』を参照してください。

レプリケーション・カタログ

レプリケーション環境の各マスター・サイトおよびマテリアライズド・ビュー・サイトには、それぞれ**レプリケーション・カタログ**があります。サイトのレプリケーション・カタログは、そのサイトのレプリケーション・オブジェクトとレプリケーション・グループに関する管理情報をメンテナンスしている、データ・ディクショナリの表およびビューの固有のセットです。レプリケーション環境に関係する各サーバーでは、そのレプリケーション・カタログの情報を使用して、レプリケーション・グループのオブジェクトのレプリケーションを自動化できます。

> **関連項目：** レプリケーション・カタログの詳細は、『Oracle9*i* レプリケーション・マネージメント API リファレンス』を参照してください。

分散スキーマ管理

レプリケーション環境では、すべての DDL 文は、Oracle Enterprise Manager のレプリケーション・マネージメント・ツールまたはレプリケーション・マネージメント API の DBMS_REPCAT パッケージを使用して発行する必要があります。具体的には、DBMS_REPCAT パッケージを使用する場合、マスター・グループにオブジェクトを追加するには CREATE_MASTER_REPOBJECT プロシージャを、レプリケート・オブジェクトを変更するには ALTER_MASTER_REPOBJECT プロシージャを使用します。EXECUTE_DDL プロシージャを使用することもできます。

レプリケーション・マネージメント・ツールまたは DBMS_REPCAT パッケージのいずれかを使用する場合、レプリケーション環境に属するすべてのサイトにすべての DDL 文がレプリケートされます。レプリケート・オブジェクトの作成にエクスポート / インポートを使用できる場合もあります。

注意： SQL*Plus などのツールを使用して直接 DDL 文を発行する場合は、その他のサイトにレプリケートされません。

レプリケーションの競合

非同期のマルチマスター・レプリケーション環境および更新可能マテリアライズド・ビュー・レプリケーション環境では、たとえば異なるサイトから発行された 2 つのトランザクションがほぼ同時に同じ行を更新する場合などに発生するレプリケーション競合に対処する必要があります。データ競合が発生した場合は、ビジネス・ルールに従って競合を解消し、すべてのサイトを適正に収束するためのメカニズムが必要です。

Advanced Replication では、レプリケート環境で発生したすべての競合がログに記録されるのみでなく、様々なビルトインの競合解消方法が用意されています。これらの方法を使用してデータベースの競合解消システムを定義すると、ビジネス・ルールに従って競合を解消できます。ビルトインの競合解消方法によって競合を解消できない場合は、独自の競合解消方法を作成して使用することもできます。

関連項目：

- データ競合が発生しないようにデータベースを設計する方法、および競合が発生した場合に使用する競合解消方法を作成する方法の詳細は、第 5 章「競合解消の概念およびアーキテクチャ」を参照してください。
- レプリケーション・マネージメント・ツールを使用して競合解消方法を構成する方法の詳細は、レプリケーション・マネージメント・ツールのオンライン・ヘルプを参照してください。
- レプリケーション・マネージメント API を使用して競合解消方法を作成する方法の詳細は、『Oracle9*i* レプリケーション・マネージメント API リファレンス』を参照してください。

マルチマスター・レプリケーションのその他のオプション

マルチマスター・レプリケーションを実装する最も一般的な方法は非同期レプリケーションです。ただし、他にもオプションが 2 つあります。同期レプリケーションとプロシージャ・レプリケーションです。

同期レプリケーション

マルチマスター・レプリケーション環境では、非同期レプリケーションまたは同期レプリケーションを使用してデータをコピーできます。非同期レプリケーションを使用する場合、1 つのマスター・サイトに加えられた変更は、一定時間後に関係するすべてのマスター・サイトに反映されます。同期レプリケーションの場合は、1 つのマスター・サイトに加えられた変更は即時に、関係するすべてのマスター・サイトに反映されます。

同期レプリケーションを使用して表を更新すると、関係するすべてのマスター・サイトで更新内容が即時にレプリケートされます。つまり、それぞれのトランザクションはすべてのマスター・サイトと関連があります。したがって、なんらかの理由から 1 つのマスター・サイトでトランザクションを処理できなかった場合、そのトランザクションはすべてのマスター・サイトでロールバックされます。

同期レプリケーションを使用して競合の発生を回避できますが、処理を円滑に行うには非常に安定した環境が必要です。たとえば、ネットワーク上の問題により 1 つのマスター・サイトと通信できない場合、ユーザーがレプリケート表を問い合せることはできますが、通信が再び確立されるまではどのトランザクションも完了しません。また、同期レプリケーションをシミュレートするように非同期レプリケーションを構成することもできます。

> **関連項目：** 非同期レプリケーション環境での同期レプリケーションのシミュレーション方法の詳細は、6-23 ページの「連続的なプッシュのスケジューリング」を参照してください。

プロシージャ・レプリケーション

バッチ処理アプリケーションでは、1 つのトランザクション内で大量のデータを変更できます。この場合、通常の行レベル・レプリケーションを実行すると、大量のデータ変更のためにネットワークの負荷が増大する可能性があります。このような問題を回避するために、レプリケーション環境で動作するバッチ処理アプリケーションでは、**プロシージャ・レプリケーション**を使用して、データ・レプリカを収集する簡単なストアド・プロシージャ・コールをレプリケートできます。プロシージャ・レプリケーションでは、アプリケーションが表の更新に使用するストアド・プロシージャのコールのみがレプリケートされます。データの変更自体はレプリケートされません。

プロシージャ・レプリケーションを使用するには、システムのデータを変更するパッケージをすべてのサイトにレプリケートする必要があります。パッケージをレプリケートした後に、各サイトでこのパッケージ用の**ラッパー**を生成する必要があります。アプリケーションがデータを変更するパッケージ・プロシージャをローカル・サイトでコールすると、ラッパーにより、最終的にレプリケーション環境内の他のすべてのサイトで同じパッケージ・プ

ロシージャがコールされます。プロシージャ・レプリケーションは、非同期または同期で実行できます。

競合の検出とプロシージャ・レプリケーション

プロシージャ・レプリケーションを使用してデータをレプリケートする場合、データをレプリケートするプロシージャにより、レプリケート・データの整合性が保証される必要があります。つまり、このようなプロシージャは、レプリケーションの競合を回避または検出し、競合を適切に解消するように設計する必要があります。したがって、プロシージャ・レプリケーションは、主に大規模バッチ操作によってのみデータベースを変更する場合に使用されます。このような状況では、多数のトランザクションが同じデータで競合することはないため、レプリケーションの競合はほとんど発生しません。

関連項目： プロシージャ・レプリケーションの詳細は、『Oracle9*i* レプリケーション・マネージメント API リファレンス』を参照してください。

2

マスター・レプリケーションの概要とアーキテクチャ

この章では、シングル・マスター・レプリケーション環境およびマルチマスター・レプリケーション環境での、Oracle のマスター・レプリケーション・サイトの概念およびアーキテクチャについて説明します。

この章の内容は、次のとおりです。

- マスター・レプリケーションの概念
- マスター・レプリケーションのアーキテクチャ

マスター・レプリケーションの概念

マスター・レプリケーションのアーキテクチャの詳細を理解するには、マスター・レプリケーションの概念について知っておく必要があります。マスター・レプリケーションの使用方法と目的がわかると、アーキテクチャの個々の要素がどのように相互に作用してマルチマスター・レプリケーション環境が作成されるのか理解が深まります。

マスター・レプリケーションについて

Oracle には、シングル・マスター・レプリケーションとマルチマスター・レプリケーションという、2 つのタイプのマスター・レプリケーションがあります。マルチマスター・レプリケーションは、複数のマスター・サイトを含んでおり、各マスター・サイトが同等ピアとして機能します。シングル・マスター・レプリケーションでは、マテリアライズド・ビュー・レプリケーションをサポートする 1 つのマスター・サイトが、数百、数千のマテリアライズド・ビュー・サイトをサポートします。また、1 つ以上のマテリアライズド・ビュー・サイトをサポートする 1 つのマスター・サイトが、複数マスター・サイト環境の一部として含まれるハイブリッド・レプリケーション環境（マルチマスター・レプリケーションとマテリアライズド・ビュー・レプリケーションの組合せ）を構成することも可能です。

マテリアライズド・ビューは、マスター・サイトのマスター表またはマテリアライズド・ビュー・サイトのマテリアライズド・ビューのいずれかに基づきます。マテリアライズド・ビューがマテリアライズド・ビューに基づいている場合は、多重化マテリアライズド・ビュー環境が作成されます。このような環境では、他のマテリアライズド・ビューの基になるマテリアライズド・ビューがマスター・マテリアライズド・ビューと呼ばれます。

> **関連項目：** 多重化マテリアライズド・ビューの詳細は、第 3 章「マテリアライズド・ビューの概要とアーキテクチャ」を参照してください。

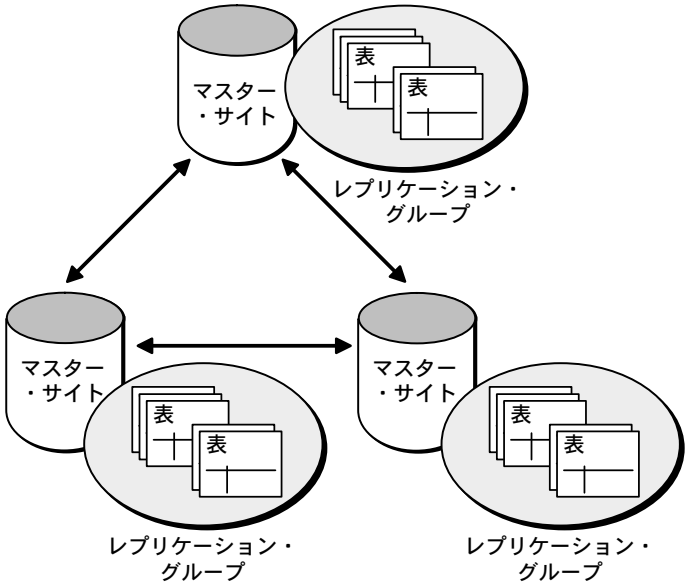
マルチマスター・レプリケーション

マルチマスター・レプリケーション（peer-to-peer レプリケーションまたは *n*-way レプリケーションとも呼ばれます）は、同等な複数のマスター・サイトから構成されており、どのマスター・サイトでも更新が可能です。個々のマスター・サイトに対する更新処理は、関係するすべてのマスター・サイトに伝播されます。図 2-1 は、マルチマスター・レプリケーション・システムを示します。

マルチマスター・レプリケーション環境でマスター・サイトとして機能している Oracle データベース・サーバーで、すべての表レプリカのデータが自動的に収束され、グローバルなトランザクション一貫性とデータ整合性が保証されます。競合解消は、各マスター・サイトで独自に行われます。マルチマスター・レプリケーションでは、各マスター・サイトのレプリケート表の完全なレプリカが提供されます。

ハイブリッド・レプリケーション環境（複数のマスター・サイトと各マスター・サイトがサポートする 1 つ以上のマテリアライズド・ビュー・サイトから構成される環境）の場合、ターゲット・マスター・サイトからマルチマスター・レプリケーション環境の他のすべてのマスター・サイトにマテリアライズド・ビューのすべての更新内容が伝播されます。次に、各マスター・サイトが、マテリアライズド・ビューのリフレッシュ時に変更をマテリアライズド・ビューに伝播します。

図 2-1 マルチマスター・レプリケーション



シングル・マスター・レプリケーション

1 つのマスター・サイトを、1 つ以上のマテリアライズド・ビュー・サイトのターゲット・マスター・サイトとして機能させることもできます。マルチマスター・レプリケーションでは、1 つのサイトに対して行った更新処理はその他のすべてのマスター・サイトに伝播されましたが、シングル・マスター・レプリケーションでは、マテリアライズド・ビューにより、更新がそのターゲット・マスター・サイトにのみ反映されます。

競合解消は、マスター・サイトまたはマスター・マテリアライズド・ビュー・サイトでのみ処理されます。マテリアライズド・ビュー・レプリケーションには、レプリケート表の全部または一部のレプリカを含めることができます。

関連項目： 1 つのマスター・サイトを使用するマテリアライズド・ビュー・レプリケーションの詳細は、第 3 章「マテリアライズド・ビューの概要とアーキテクチャ」を参照してください。

マスター・サイト

マスター・サイトは、マルチマスター・レプリケーション環境のノード、あるいはシングル・マスター・レプリケーション環境またはマルチマスター・レプリケーション環境の 1 つ以上のマテリアライズド・ビュー・サイトのマスター（あるいはその両方）として設定できます。レプリケート・オブジェクトはマスター・サイトに格納され、ユーザーがアクセスできます。

マスター定義サイト マルチマスター・レプリケーション環境では、1 つのマスター・サイトがマスター・グループのマスター定義サイトとして機能します。この特定のサイトで、マルチマスター・レプリケーション環境の管理作業やメンテナンス作業の多くが実行されます。

マスター定義サイトはマルチマスター環境のどのマスター・サイトでもかまいませんが、各マスター・グループ内に設定できるのは 1 つのみです。また、必要に応じて、マスター定義サイトを別のマスター・サイトに変更することもできます。

マテリアライズド・ビュー・レプリケーションをサポートするシングル・マスター・サイトは、デフォルトでマスター定義サイトになります。

マルチマスター・レプリケーションの使用目的

基本的に、レプリケーションを使用するのは、必要なときに必要な場所でデータを確実に使用するためです。次の項では、情報の配布要件が異なるいくつかの環境について説明します。使用するレプリケーション環境には、次のような要件が 1 つ以上ある可能性があります。

フェイルオーバー

マルチマスター・レプリケーションは、ミッション・クリティカルなデータベースの可用性を保護するのに役立ちます。たとえば、マルチマスター・レプリケーション環境では、システムまたはネットワーク障害が発生してプライマリ・サイトが使用できなくなった場合に備え、データベース内のデータをレプリケートしてフェイルオーバー・サイトを確立できます。このようなフェイルオーバー・サイトは、プライマリ・サイトが同時に稼働しているときでも、アプリケーションからアクセスできる完全に機能したデータベースとして機能します。

Oracle Net を使用すると、接続時に自動的に起動されるフェイルオーバーを構成できます。この機能により、最初のマスター・サイトに障害が発生した場合、Oracle Net は別のマスター・サイトに接続を切り替えることができます。接続時自動フェイルオーバーを構成するには、`tnsnames.ora` ファイルで、`FAILOVER_MODE` パラメータを on に設定し、複数の接続記述子を指定します。

> **関連項目：** 接続時フェイルオーバーの構成方法の詳細は、『Oracle9*i* Net Services 管理者ガイド』を参照してください。

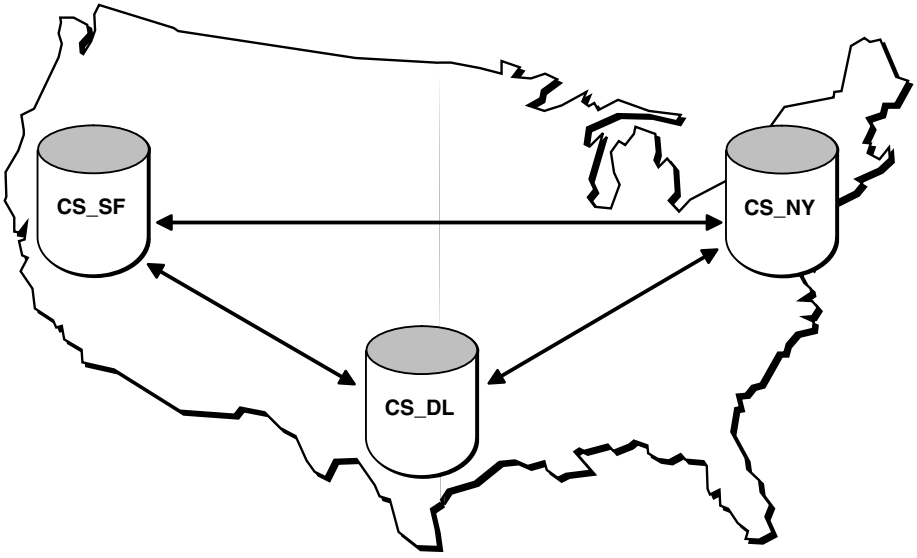
ロード・バランシング

マルチマスター・レプリケーションは、次のような目的で複数ポイントのデータベース情報にアクセスすることが必要なトランザクション処理アプリケーションに役立ちます。

- 過度のアプリケーション負荷の分散
- 連続的な可用性の保証
- よりローカライズされたデータ・アクセスの提供

アプリケーション負荷を分散することが必要なアプリケーションとして一般的なものに、サービス指向のアプリケーションがあります。

図 2-2 複数ポイントの更新アクセスをサポートするマルチマスター・レプリケーション



Oracle Real Application Clusters と Advanced Replication との比較

Advanced Replication と Oracle Real Application Clusters のどちらを使用するかを検討する必要があるのは、ロード・バランシングと耐障害性の分野です。

ロード・バランシング : Advanced Replication は複数のデータベースに対する読取りロード・バランシングを提供し、Oracle Real Application Clusters は複数のインスタンスに対する読取りおよび書込みロード・バランシングを提供します。書込みはそれぞれ各レプリケーション・サイトで実行する必要があるため、レプリケーションでは書込みロード・バランシングは提供されていません。

耐障害性 : Advanced Replication の場合、各レプリケーション・サイトが地理的に異なる地域に置かれている可能性があるため、自然災害、停電またはサボタージュに対する耐障害性は高くなります。Oracle Real Application Clusters は、物理的に同一の環境にあるクラスタまたはその他の大規模パラレル・システム上で動作するため、Oracle Replication では対処できる物理的問題に対する耐障害性はありません。

相互運用性 : Advanced Replication は、Oracle が稼働している様々なプラットフォームおよびオペレーティング・システム間でデータをレプリケートできます。Oracle Real Application Clusters 環境内のインスタンスは、同一プラットフォーム上で実行する必要があります。

非接続マテリアライズド・ビュー環境のサポート

マテリアライズド・ビュー・レプリケーションを使用すると、ユーザーは、マスター・サイトに接続せずに、マスター・サイトからのレプリケート・データの全部またはサブセットをリモートで格納できます。たとえば、営業支援システムでは、個人のラップトップ（非接続デバイス）に個々の営業担当員に関連するデータのサブセットを格納します。

マスター・サイトは、マテリアライズド・ビュー環境のターゲットとして機能します。マスター・サイトのサポートには、次のものがあります。

- すべてのマテリアライズド・ビュー・サイトをサポートするシングル・マスター・サイト。競合解消がマスター・サイトまたは（多重化マテリアライズド・ビュー環境の）マスター・マテリアライズド・ビュー・サイトでのみ実行されるため、データの不一致が発生する確率が低下します。
- マルチマスター・レプリケーションとマテリアライズド・ビュー・レプリケーションの組合せ。この組合せでは、マルチマスター構成の様々なマスターがマテリアライズド・ビュー・グループのターゲットになります。この構成を使用すると、ネットワーク負荷が複数のマスター・ノードに分散され、いずれかのマスター・ノードが使用できなくなった場合に備えて拡張性と可用性を向上できます。

マルチマスター・レプリケーション・プロセス

マルチマスター・レプリケーションには、非同期と同期という 2 つのタイプがあります。

非同期レプリケーション（ストア / フォワード・レプリケーションとも呼ばれます）では、ローカルで加えられたすべての変更を取得してキューに格納し、一定の間隔でリモート・サイトに伝播して変更を適用します。この形式のレプリケーションでは、すべてのサイトでデータが収束するまでに一定の時間がかかります。

同期レプリケーション（リアルタイム・レプリケーションとも呼ばれます）では、1 つのトランザクションの一部として、レプリケーション環境に関係するすべてのサイトで変更を適用し、レプリケート・プロシージャを実行します。いずれかのサイトでデータ操作言語（DML）文またはプロシージャが失敗すると、トランザクション全体がロールバックされます。同期レプリケーションを使用すると、各サイトのデータ整合性がリアルタイムで保証されます。

マスター・サイトの伝播モードは、非同期から同期に、または同期から非同期に変更できます。マスター・グループ内のマスター・サイトの伝播モードを変更した場合、すべてのマスター・グループ・オブジェクトに対してレプリケーション・サポートを再度生成する必要があります。レプリケーション・サポートを再生成すると、内部トリガーがアクティブ化され、内部パッケージが再生成されて、すべてのマスター・サイトにあるオブジェクトのレプリケーションがサポートされます。また、マルチマスター・レプリケーション環境では、同期レプリケーションおよび非同期レプリケーションの両方を混在させることも可能です。

> **関連項目：** 詳細は、2-42 ページの「複合モードのマルチマスター・システムについて」を参照してください。

非同期レプリケーション

非同期レプリケーションでは、DML またはレプリケート・プロシージャを実行すると、マルチマスター・レプリケーション環境のすべてのマスター・サイトに個別に伝播されます。各トランザクションにおいてデータが伝播されるのは、DML またはレプリケーション・プロシージャをローカルで実行した後です。

非同期レプリケーションがデフォルトのレプリケーション・モードです。非同期レプリケーションでは、同期レプリケーションの場合よりもネットワーク・リソースやハードウェア・リソースの使用を抑えることができるため、可用性およびパフォーマンスが高くなります。

ただし、非同期レプリケーションを使用すると、変更内容が伝播されるまでの一定時間、レプリケーション環境にある別のマスター・サイトのデータ・セットと一致しません。また、非同期レプリケーションでは、データの競合が発生する可能性もあります。

非同期レプリケーション・プロセスは次のとおりです。

1. DML 文を発行または、レプリケート・プロシージャのラッパーを実行します。

   表がレプリケーション用に設定された後、ユーザーがこの表に対してコミットした DML は、他のすべてのマスター・サイトへのレプリケーション用に取得されます。

   挿入、更新または削除の対象となる各行に対して、内部トリガーによって遅延**リモート・プロシージャ・コール（RPC）**が作成され、遅延トランザクション・キューに入れられます。遅延トランザクション・キューには、すべての遅延 RPC が入っています。

   プロシージャがレプリケートされており、そのラッパーがマスター・サイトで実行されると、プロシージャ・コールが遅延トランザクション・キューに入れられます。

2. 遅延 RPC を遅延トランザクション・キューに格納します。

   遅延トランザクション・キュー内の各トランザクションには接続先リストがあり、遅延トランザクションの伝播先が定義されています。このリストには、発行元サイト以外の全マスター・サイトが記載されています。各サイトには遅延トランザクション・キューが 1 つあり、この 1 つのキューが複数のレプリケーション・グループにより使用されます。

3. 接続先へ遅延トランザクション・キューのエントリを送信します。

   スケジュールした間隔で、または必要に応じて、遅延トランザクション・キュー内の遅延トランザクションがターゲットの接続先に伝播されます。接続先ごとにスケジュール間隔を変えられます。

4. 遅延トランザクション・キューのエントリがリモートの接続先で適用されます。

   遅延トランザクションがターゲットの接続先へ伝播されると、接続先サイトで内部パッケージがコールされて各遅延 RPC が適用されます。接続先サイトで遅延トランザクションの適用に失敗した場合、遅延トランザクションは再送されて、接続先サイトのエラー・キューに入れられます。DBA は、このエラー・キューを使用してエラー条件を修正し、遅延トランザクションを再び適用できます。

   遅延トランザクション・キューのエントリがリモートの接続先で適用されるときに、データ競合がないかがチェックされます。データ競合が検出された場合、その競合はリモートの接続先でログに記録され、オプションで競合解消方法が起動します。

5. 遅延トランザクションがすべてのリモート・マスター・サイトに正常にプッシュされた後、元のサイトではこのトランザクションが遅延トランザクション・キューからすぐにはパージされません。ユーザーが定義した間隔で実行されるパージ・ジョブにより、後でパージできます。

関連項目： 詳細は、第 5 章「競合解消の概念およびアーキテクチャ」を参照してください。

同期レプリケーション

同期レプリケーションでは、1 つのトランザクションが終了するまでの間に、ローカル・サイトで加えられた変更をレプリケーション環境内の同期リンクしたマスター・サイトに伝播します。いずれかのマスター・サイトで伝播に失敗した場合、ローカルのマスター・サイトで最初に加えられた変更を含め、トランザクション全体がロールバックされます。この厳しい規則により、レプリケーション環境全体のデータ整合性を保つことができます。非同期レプリケーションとは異なり、マスター・サイト間のデータが一致しない時間はありません。

> **関連項目：** 1 つの環境で同期レプリケーションと非同期レプリケーションを混在させる方法の詳細は、2-42 ページの「複合モードのマルチマスター・システムについて」を参照してください。

同期レプリケーションでは、レプリケーション環境でデータ競合が発生しないことも保証されています。このようなメリットがある一方で、停止時間に関する制限が多いハードウェア・リソースやネットワーク・リソースを必要とするというデメリットもあります。たとえば、ノードが 6 つあるマルチマスター環境で、1 つのマスター・サイトが使用できなくなった場合、どのマスター・サイトでもトランザクションは完了しません。

ただし、非同期レプリケーションでは、ダウンしたサイトが使用可能になるまで元のサイトで遅延トランザクションが保持されます。その間に、他のレプリケーション・サイトにはトランザクションが正常に伝播され適用されます。

また、同期レプリケーションを使用すると、問合せがローカルで実行されるため高パフォーマンスを維持できるのに対して、更新処理には時間がかかります。これは、すべての更新内容が正しく伝播され、リモートの接続先サイトへ正しく適用できるように、2 フェーズ・コミット・プロトコルを使用しているためです。

> **関連項目：** 2 フェーズ・コミットの詳細は、『Oracle9*i* データベース管理者ガイド』を参照してください。

同期レプリケーション・プロセスは次のとおりです。

1. DML 文を発行または、レプリケート・プロシージャのラッパーを実行します。

   表がレプリケーション用に設定された後、ターゲット表に対してユーザーがコミットした DML が、他のすべてのマスター・レプリケーション・サイトへのレプリケーション用に取得されます。

   プロシージャがレプリケートされており、そのラッパーがマスター・サイトで実行されると、プロシージャ・コールがレプリケーション用に取得されます。

2. DML またはラッパーを実行すると即時に接続先のサイトに伝播されます。

   内部トリガーは、DML を取得し、これらのアクションを即時にレプリケーション環境の各マスター・サイトに伝播します。そして、これらのアクションをプロパゲータのデータベース・リンクのセキュリティ・コンテキストで適用し、内部 RPC を使用して接続先サイトにこれらのアクションを適用します。

内部トリガーと同様に、レプリケート・プロシージャのラッパーは、レプリケーション環境の各マスター・サイトにプロシージャ・コールを即時に伝播します。

いずれかのマスター・レプリケーション・サイトでトランザクションが失敗した場合、すべてのマスター・サイトでトランザクションがロールバックされます。この方法により、すべてのマスター・サイト間でのデータ整合性が保証されます。トランザクションがいずれかのサイトで失敗するとロールバックを実行する必要があるので、同期レプリケーションは、可用性の高いネットワーク、データベースおよびそれに関連するハードウェアに大きく依存します。

競合解消の概念

表がレプリケートされるときに、任意のレプリケーション・サイト（マスター・サイトまたはマテリアライズド・ビュー・サイトのいずれか）のレプリケート表に適用される DML のうち、接続先サイトでデータ競合を引き起こす DML は、接続先サイトの Oracle サーバーによって自動的に検出されます。マテリアライズド・ビュー・サイトで発生したデータ競合は、マテリアライズド・ビューのターゲット・マスター・サイトまたはマスター・マテリアライズド・ビュー・サイトで検出され、解消されます。

たとえば、次のマスター・グループが 1 時間に 1 回変更を伝播するようにスケジュールされていると仮定します。

時間	マスター・サイト A	マスター・サイト B	ステータス
午前 8:00	マスター・サイト B に変更を伝播	マスター・サイト A に変更を伝播	データの収束
午前 8:15	行 1 を更新	-	-
午前 8:30	-	行 1 を更新	-
午前 9:00	マスター・サイト B に変更を伝播	マスター・サイト A に変更を伝播	行 1 で競合検出

2 つのデータ伝播の間の時間を伝播間隔とみなした場合、1 つの間隔中に複数のサイトが同じ行を更新すると競合が発生します。

前述した更新の競合のみでなく、挿入および削除時にも競合が発生します。次のようなタイプの競合について考えてみます。各競合は、同じ間隔内に競合するアクションが起こると発生します。

競合タイプ	要約
更新の競合	DML 文が他のサイトに伝播される前に、複数の DML 文が複数のレプリケーション・サイトで同一行に適用される場合。
一意性競合	複数のサイトで挿入を実行した結果、各挿入の主キー（またはその他の一意の列のセット）に同じ値が存在する場合、またはあるサイトで更新を実行した結果、主キー（またはその他の一意の列のセット）が変更され、その主キーに別のサイトで挿入した値と同じ値が存在する場合。
削除の競合	あるサイトで行が削除され、別のサイトで同じ行に対して更新が行われる場合。この結果、存在しない行を更新しようとする可能性や、または複数のサイトで同じ間隔で同じ行を削除しようとする可能性があります。

関連項目： その他のタイプのデータ競合の詳細は、第 5 章「競合解消の概念およびアーキテクチャ」を参照してください。

データ競合が検出されると、次のアクションが発生します。

1. 競合解消方法によって、データ競合の解消が試みられます。
2. 競合が解消されない場合は、接続先サイトのエラー・キューでデータ競合がログに記録されます。

エラー・キューにデータ競合がログされた場合、データベース管理者はデータ競合を手動で解消する必要があります。

オラクル社が提供する競合解消方法またはユーザー定義の競合解消方法を使用する場合、Oracle サーバーはデータ競合の自動的な解消を試みます。ユーザーが実装する競合解消方法は、レプリケーション環境で定義されているビジネス・ルールに準拠し、データ収束を保証する必要があります。実装する競合解消方法の要件に合うように、表の変更が必要になる可能性があります。たとえば、最新のタイムスタンプによる競合の解消方法では、実装対象の表にタイムスタンプ列が必要になります。

ユーザー定義型のレプリケーション

Oracle の**オブジェクト型**はユーザー定義データ型で、これを使用すると現実の複雑なエンティティ（顧客や注文など）をデータベース内に単一エンティティ（**オブジェクト**）としてモデル化できます。オブジェクト型は、CREATE TYPE . . .AS OBJECT 文を使用して作成します。オブジェクト型とオブジェクトは、マルチマスター・レプリケーション環境のマスター・サイト間でレプリケートできます。

表の 1 列を占有する Oracle オブジェクトを、**列オブジェクト**と呼びます。一般に、列オブジェクトが含まれている表にはその他の列も含まれており、このような列のデータ型は組込みデータ型（VARCHAR2 や NUMBER など）の場合があります。**オブジェクト表**とは、各行がオブジェクトを表す特殊な表です。オブジェクト表の各行は**行オブジェクト**です。

コレクションをレプリケートすることもできます。コレクションとは、VARRAY データ型およびネストした表のデータ型に基づいたユーザー定義データ型です。VARRAY は CREATE TYPE . . .AS VARRAY 文を、ネストした表は CREATE TYPE . . .AS TABLE 文を使用して作成します。

注意：

- ユーザー定義型とユーザー定義型に基づいたオブジェクトをレプリケートするには、マスター・サイト間の互換性のレベルが 9.0.1 以上の必要があります。互換性のレベルは、COMPATIBLE 初期化パラメータによって制御されます。
- Advanced Replication では、型の継承をサポートしていません。

関連項目： ユーザー定義型、列オブジェクト、オブジェクト表およびコレクションの詳細は、『Oracle9*i* アプリケーション開発者ガイド - オブジェクト・リレーショナル機能 』を参照してください。この項では、このマニュアルに記載されている基本的な情報を理解していることを前提としています。

レプリケーション・サイトでの型の一致

ユーザー定義型には、オブジェクト、ネストした表および VARRAY を含み、CREATE TYPE 文で作成された型がすべて含まれます。ユーザー定義型に基づいたスキーマ・オブジェクトをレプリケートするには、全レプリケーション・サイトに同一のユーザー定義型自体が存在する必要があります。

ユーザー定義型とそれに基づいたスキーマ・オブジェクトをレプリケートする場合は、次の条件が適用されます。

- 全レプリケーション・サイトで、オブジェクト識別子（OID）、スキーマ所有者およびレプリケートされるユーザー定義型のタイプ名が同一である必要があります。

- ユーザー定義型がオブジェクト型の場合は、全レプリケーション・サイトで、オブジェクト型の属性の順序とデータ型が一致する必要があります。属性の順序とデータ型は、オブジェクト型を作成するときに設定します。たとえば、次のようなオブジェクト型の場合を考えます。

  ```
  CREATE TYPE cust_address_typ AS OBJECT
       (street_address     VARCHAR2(40),
        postal_code        VARCHAR2(10),
        city               VARCHAR2(30),
        state_province     VARCHAR2(10),
        country_id         CHAR(2));
  /
  ```

  全レプリケーション・サイトで、street_address はこの型の最初の属性で VARCHAR2(40)、postal_code は 2 番目の属性で VARCHAR2(10)、city は 3 番目の属性で VARCHAR2(30)、というようになっている必要があります。

- 全レプリケーション・サイトで、ユーザー定義型のハッシュコードが一致する必要があります。Oracle では、ユーザー定義型が検査され、ハッシュコードが割り当てられます。この検査では、型の属性、属性の順序および型の名前が調べられます。複数の型についてこれらの項目がすべて同じである場合、型のハッシュコードは同じです。型のハッシュコードは、DBA_TYPE_VERSIONS データ・ディクショナリ・ビューを問い合せて表示できます。

全レプリケーション・サイトでユーザー定義型を同一にするには、次のいずれかの方法でユーザー定義型を作成する必要があります。

- レプリケーション・マネージメント API の使用
- CREATE TYPE 文の使用
- エクスポート / インポートの使用

レプリケーション・マネージメント API の使用 オラクル社では、ユーザー定義型を含むレプリケート・オブジェクトの作成、変更または削除には、レプリケーション・マネージメント API の使用をお薦めします。レプリケーション・マネージメント API を使用しない場合、レプリケーション・エラーが発生することがあります。たとえば、前述の条件を満足するユーザー定義型をマスター・グループの全レプリケーション・サイトに追加する場合は、マスター定義サイトでその型を作成し、DBMS_REPCAT パッケージ内の CREATE_MASTER_REPOBJECT プロシージャを使用して、マスター・グループにその型を追加します。

> **関連項目：**『Oracle9*i* レプリケーション・マネージメント API リファレンス』

CREATE TYPE 文の使用 型を作成するには、レプリケーション・サイトで CREATE TYPE 文を使用できます。事前に全レプリケーション・サイトで型を作成し、これをレプリケーション・グループに追加するときに実行する必要がある場合があります。

この方法を選択する場合は、次のことを確認する必要があります。

- 全レプリケーション・サイトで型が同じスキーマ内にあること。
- 全レプリケーション・サイトで型の属性および順序が同一であること。
- 全レプリケーション・サイトで型の各属性のデータ型が同一であること。
- 全レプリケーション・サイトで型のオブジェクト識別子が同一であること。

型のオブジェクト識別子は、DBA_TYPES データ・ディクショナリ・ビューを問い合せると検出できます。たとえば、cust_address_typ のオブジェクト識別子（OID）を検出するには、次の問合せを入力します。

```
SELECT TYPE_OID FROM DBA_TYPES WHERE TYPE_NAME = 'CUST_ADDRESS_TYP';

TYPE_OID
--------------------------------
6F9BC33653681B7CE03400400B40A607
```

または、多数のレプリケーション・サイトに新しい型を作成する場合は、型の作成中に各サイトで同一の OID を指定します。この場合、次の問合せを実行すると、グローバルに一意の OID を識別できます。

```
SELECT SYS_GUID() OID FROM DUAL;
```

型の OID を認識した後、次の手順を実行して、型が存在していないレプリケーション・サイトで型を作成します。

1. 型を所有するユーザーとしてレプリケーション・サイトにログインします。レプリケーション・サイトにこのユーザーがない場合は、ユーザーを作成します。
2. CREATE TYPE 文を発行し、OID を指定します。

```
CREATE TYPE oe.cust_address_typ OID '6F9BC33653681B7CE03400400B40A607'
     AS OBJECT (
     street_address     VARCHAR2(40),
     postal_code        VARCHAR2(10),
     city               VARCHAR2(30),
     state_province     VARCHAR2(10),
     country_id         CHAR(2));
/
```

これで、レプリケーション・サイトで型を使用できます。

関連項目：『Oracle9*i* レプリケーション・マネージメント API リファレンス』

エクスポート / インポートの使用 エクスポート・ユーティリティおよびインポート・ユーティリティを使用すると、レプリケーション・サイト間の型の一致を維持できます。ユーザー定義型に基づいたオブジェクト表またはユーザー定義型に基づいた列オブジェクトを含む表をエクスポートするとき、エクスポートを実行するユーザーにこれらの型に対するアクセス権がある場合は、ユーザー定義型も自動的にエクスポートされます。このような表を別のレプリケーション・サイトでインポートしたときのユーザー定義型は、エクスポートを実行したサイトと同一の型になります。

したがって、エクスポート / インポートを使用して新規レプリケーション・サイトで事前にレプリケーション表を作成しておき、これらの表をレプリケーション・グループに追加するときに「use existing object」オプションを指定できます。この方法を使用すれば、レプリケーション・サイトで型が一致するようになります。

> **関連項目：** エクスポート / インポートの詳細は、『Oracle9*i* データベース・ユーティリティ』を参照してください。

オブジェクト表

オブジェクト表をレプリケートする場合は、次の条件が適用されます。

- 全レプリケーション・サイトでオブジェクト表の OID を同じにする必要があります。
- 全レプリケーション・サイトでオブジェクト表の各行オブジェクトの OID を同じにする必要があります。

この条件を満たすには、レプリケーション・マネージメント API を使用して、オブジェクト表をレプリケーション・グループに追加し、オブジェクト表を変更し、さらにレプリケーション・グループからオブジェクト表を削除します。たとえば、オブジェクト表をマスター・グループに追加するときに DBMS_REPCAT パッケージ内の CREATE_MASTER_REPOBJECT プロシージャを使用すると、この条件が満たされます。この条件を満たすために、エクスポート / インポートを使用してレプリケーション・サイトでオブジェクト表を事前に作成することもできます。

もう 1 つのオプションとしては、複数のレプリケーション・サイトでオブジェクト表を作成するときにオブジェクト表の OID を指定する方法があります。このオプションを使用する場合は、次の手順を実行します。

1. DUAL ビューに対する問合せを実行し、グローバルに一意の OID を求めます。

```
SELECT SYS_GUID() OID FROM DUAL;

OID
--------------------------------
81D98C325D4A45D0E03408002074B239
```

2. 各レプリケーション・サイトで、手順 1 で戻された OID を使用して categories_tab オブジェクト表を作成します。

```
CREATE TABLE categories_tab5 OF category_typ
   OID '81D98C325D4A45D0E03408002074B239'
   (category_id PRIMARY KEY);
```

コレクション列を含む表

コレクション列とは、VARRAY データ型およびネストした表のデータ型に基づいた列です。Oracle ではコレクション列のレプリケーションをサポートしています。コレクション列を含む表をレプリケーション・グループに追加すると、コレクション列内のデータは自動的にレプリケートされます。コレクション列が VARRAY の場合、4KB より大きい VARRAY は BLOB として格納されます。

コレクション列がネストした表の場合は、ネストした表の記憶表内の各行に対して行レベルのレプリケーションが実行されます。たとえば、記憶表の 5 つの行での変更により、5 つの別々のリモート・プロシージャ・コール（RPC）が発生し、競合検出およびオプションの競合解消のフェーズが 5 つになります。記憶表は索引構成表として格納できます。

さらに、ネストした表を含む行に対する DML により、親表に対する RPC と、ネストした表の記憶表で影響を受ける各行に対する RPC がそれぞれ別々に発生します。Oracle では、親表の作成中に整合性制約を明示的に指定しないかぎり、親表の行と記憶表の行との間の参照整合性検査は実行しません。競合をすべて検出するために、レプリケート表にはこのような制約を指定することをお薦めします。

ネストした表とその記憶表との間の競合が検出されるように、この 2 つの間には遅延可能な外部キー制約を定義することをお薦めします。遅延可能な外部キー制約がないと、競合により記憶表に行が挿入されアクセスできなくなることがあります。遅延可能な外部キー制約が定義してあると、このような場合はエラーが発生して競合が検出されます。制約を遅延させるには、SET CONSTRAINTS 文の DEFERRED 句を使用します。

次のアクションは、レプリケート表内のネストした表の記憶表に対して直接実行することはできません。

- レプリケーション・グループへの記憶表の追加
- 記憶表の変更
- 記憶表の削除
- 記憶表に対するレプリケーション・サポートの生成

これらのアクションは、記憶表の親表に対して実行するときに間接的に発生します。さらに、記憶表内の列のサブセットはレプリケートできません。

REF 列を含む表

REF とは Oracle の組込みデータ型で、オブジェクト表内の行オブジェクトを指す論理的なポインタです。有効範囲付 REF は、指定されたオブジェクト表に対する参照のみを含む REF で、有効範囲なし REF にはデータベース内のすべてのオブジェクト表に対する参照を含められます。有効範囲付 REF は、有効範囲なし REF と比べて必要とする記憶領域が少なく、より効率的なアクセスを提供します。Oracle では、REF 型の表のレプリケーションをサポートします。

有効範囲付 REF 有効範囲付 REF を含む表がレプリケートされ、REF により参照されているオブジェクト表がレプリケートされない場合は、有効範囲付 REF を含む表をレプリケートする前に、参照されているオブジェクト表のないサイトに表を作成する必要があります。そうしないと、有効範囲付 REF により、参照されているオブジェクト表が検出できないときに、この表のレプリケートによりエラーが発生します。このような場合は、通常、参照されているオブジェクト表もレプリケートすることが最善の方法です。レプリケートしないと、様々なレプリケーション・サイトでこの表の同期がなくなる可能性があるためです。

有効範囲なし REF 有効範囲なし REF を含む表がレプリケートされ、REF により参照されているオブジェクト表がレプリケートされない場合、REF で参照オブジェクトが検出できない場合に、レプリケート・サイトで参照先がない REF が発生する可能性があります。レプリケートされた REF が有効になるには、参照されているオブジェクト表が各レプリケーション・サイトに存在する必要があります。

WITH ROWID オプションを使用して作成された REF REF 列に WITH ROWID オプションを指定すると、REF で参照されている行オブジェクトの行 ID のヒントがメンテナンスされます。REF に含まれている行 ID を使用して、参照されているオブジェクトを直接検出できます。OID 索引から行 ID をフェッチする必要はありません。WITH ROWID オプションは、有効範囲付 REF に対してはサポートされていません。

WITH ROWID オプションを使用して作成された REF をレプリケートすると、REF が最初に作成または変更されたサイトを除き、各レプリケーション・サイトで行 ID ヒントが正しくなくなります。他のサイトでは REF に含まれている ROWID 情報が無意味になります。行 ID ヒントは自動的には修正されません。無効な行 ID ヒントは、パフォーマンス上の問題の原因になります。この場合は、ANALYZE 文を使用して、行 ID ヒントが正しくないレプリケーション・サイトで行 ID ヒントを修正する必要があります。

> **関連項目：** ANALYZE 文の詳細は、『Oracle9*i* SQL リファレンス 』を参照してください。

マスター・レプリケーションのアーキテクチャ

レプリケーション環境は、レプリケーション・マネージメント・ツールのオンライン・ヘルプおよび『Oracle9*i* レプリケーション・マネージメント API リファレンス』で説明している手順や例に従えば作成できますが、レプリケーションのアーキテクチャを理解すると、レプリケーションをサポートするデータベース環境の設定、レプリケーション環境のチューニングやトラブルシューティングを、必要に応じて実施するための重要な情報が得られます。この項では、メカニズムとプロセスの観点からレプリケーションのアーキテクチャについて説明します。

マスター・サイトのメカニズム

Oracle では、レプリケーション環境をサポートするために、マルチマスター・レプリケーション環境またはシングル・マスター・レプリケーション環境のいずれかに関係する各マスター・サイトで次のメカニズムを使用しています。これらのマスター・サイト・メカニズムの一部は、特定の環境でのみ必要となります。

マスター・サイトのロールおよびユーザー

セキュリティ要件によっては、管理、伝播および受信の 3 つのロールが 1 人のレプリケーション管理者に統合されている場合があります。実際、ほとんどのマルチマスター・レプリケーション環境では、1 人のユーザーがレプリケーションの管理、伝播および受信のロールを実行します。セキュリティ要件がより厳格な場合は、次のロールを別々のユーザーに割り当てることができます。

注意： このコンテキストのロールという用語は、SQL 用語のロールとは関連がありません。レプリケーションにおけるロールは、PL/SQL のストアド・プロシージャまたは個々の権限（あるいはその両方）を使用して付与されます。

レプリケーション管理者 レプリケーション管理者は、レプリケーション環境のマスター・サイトに関連するすべての管理機能を実行します。一般的に、1 つのレプリケーション環境に対して 1 人のレプリケーション管理者を置くことをお薦めします。レプリケーション管理者は、レプリケーションをサポートするようにデータベースを準備することに加えて次の作業を実行します。

- 個々のマスター・レプリケーション・グループの作成とメンテナンス
- 各マスター・サイトの追加と削除
- キューの管理
- レプリケーション環境の状態（通常および静止中）の制御

レプリケーション管理者のデフォルトのユーザー名は `REPADMIN` ですが、別のユーザー名を使用することもできます。

プロパゲータ プロパゲータは、遅延トランザクション・キューに含まれる各トランザクションを接続先に伝播する作業を実行します。プロパゲータは、各データベースに 1 つ存在します。つまり、異なるスキーマを管理するために複数のレプリケーション管理者を置くことはできますが、それぞれのデータベースには 1 つのプロパゲータしか配置できません。

受信者 受信者は、プロパゲータから遅延トランザクションを受信し、それを適用します。受信者に遅延トランザクション内のコールを適用する適切な権限が付与されていない場合、遅延トランザクション全体が接続先のエラー・キューに入れられます。受信者を登録するには、DBMS_REPCAT_ADMIN パッケージ内の REGISTER_USER_REPGROUP プロシージャを使用します。

データベース・リンク

データベース・リンクは、マスター・サイトとマテリアライズド・ビュー・サイト間でデータをレプリケートするためのパイプを提供します。マルチマスター環境では、各マスター・サイトからその他の各マスター・サイトに対してデータベース・リンクが存在します。つまり、各マスター・サイトにつき N - 1 個（N はマスター・サイトの合計数）のデータベース・リンクが存在します。

図 2-3 データベース・リンクを表す矢印

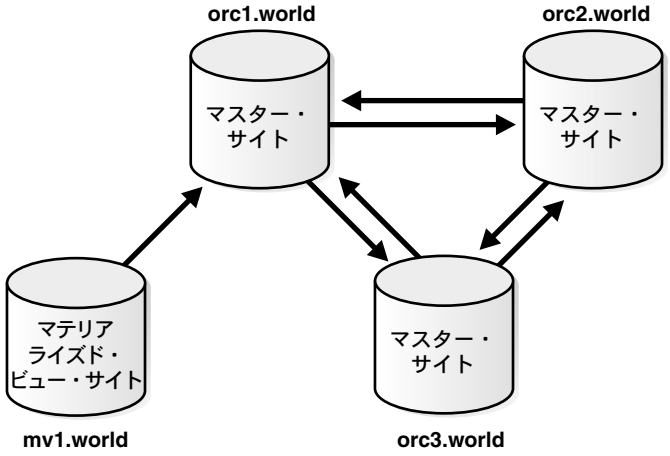


図 2-3 では、各マスター・サイトから他のマスター・サイトに対して 2 つ（N-1、すなわちこの例では 3-1=2）のデータベース・リンクが存在します。この構成では、マルチマスター・レプリケーションに必要な両方向通信チャネルがマスター・サイト間で保証されます。マテリアライズド・ビュー・サイトの場合は、マテリアライズド・ビュー・サイトからマスター・サイトへのリンクのみが必要な点に注意してください。マスター・サイトからマテリアライズド・ビュー・サイトへのデータベース・リンクは不要です。

最も基本的なデータベース・リンクは、個々のマスター・サイトのレプリケーション管理者と、関係する他のマスター・サイトのレプリケーション管理者とを結ぶデータベース・リンクです。

ただし、通常はレプリケーション環境にデータベース・リンクのセットを追加します。レプリケーション管理者のデータベース・リンクを作成する前に、CONNECT TO 句を指定せずに各マスター・サイト間にパブリック・データベース・リンクを作成します。パブリック・データベース・リンクは、リモート・データベースのサービス名を指定する USING 句を使用して各データベース・リンクのターゲットを指定します。

パブリック・データベース・リンクを作成した後で、レプリケーション管理者のプライベート・データベース・リンクを作成できます。プライベート・データベース・リンクの作成時は CONNECT TO 句を指定しますが、対応付けられたパブリック・データベース・リンクにより、USING 句の指定が不要になります。

パブリック・データベース・リンクとプライベート・データベース・リンクの両方を使用すると、データベース・リンクの管理に必要な労力が軽減されます。次の利点もあります。

- 複数のセットのプライベート・データベース・リンクで同じパブリック・データベース・リンクを共有でき、しかもデータベース・リンクの管理が簡単になります。
- データベース・リンクのターゲット・データベースを変更した一方で、ターゲット・データベースのサービス名がそのままの場合、変更する必要があるのは tnsnames.ora ファイルのターゲット・データベースのエントリのみです。USING 句により指定されるのはリモート・ターゲット・データベースのサービス名であることを覚えておいてください。ターゲットが同じプライベート・データベース・リンクはすべて、パブリック・データベース・リンクの USING 句で定義した接続先にリンクされます。

  たとえば、データベースを別のサーバーに移動しても同じサービス名をそのまま使用する場合は、各レプリケーション・サイトで tnsnames.ora ファイルのリモート・データベースのエントリを更新すれば、データベース・リンクを再作成する必要はありません。

前述のように、レプリケーション管理者は、通常、マルチマスター環境の管理作業および伝播作業を実行します。これらの作業を実行するのは 1 人のユーザーであるため、レプリケーション管理者用にプライベート・データベース・リンクを 1 セットのみ作成する必要があります。

ただし、レプリケーション管理者以外のユーザーが伝播を実行するマルチマスター・レプリケーション環境では、これらのユーザーのそれぞれに対して適切なプライベート・データベース・リンク・セットを作成する必要があります。

関連項目：

- データベース・リンクおよびその作成方法の詳細は、『Oracle9*i* データベース管理者ガイド』を参照してください。
- サービス名と tnsnames.ora ファイルの詳細は、『Oracle9*i* Net Services 管理者ガイド』を参照してください。

レプリケーション・マネージメント・ツールにより作成されたデータベース・リンク

Oracle Enterprise Manager のレプリケーション・マネージメント・ツールの「Setup Wizard」を使用してレプリケーション・サイトを設定する場合、デフォルトでは、tnsnames.ora ファイルまたは Oracle Management Server のサービス名の説明を含む USING 句を使用して、「Setup Wizard」によりデータベース・リンクが作成されます。

たとえば、tnsnames.ora ファイルでサイトのエントリが次のように指定されているとします。

```
HQ.MYCOMPANY.COM =
(DESCRIPTION=
   (ADDRESS=(PROTOCOL=TCP)(HOST=server1)(PORT=1521))
   (CONNECT_DATA=(SID=hqdb)(SERVER=DEDICATED)))
```

ここで、サービス名は HQ.MYCOMPANY.COM で、その説明は最初の等号の後に続くテキストです。次の文は、「Setup Wizard」により作成された HQ.MYCOMPANY.COM サイトへのデータベース・リンクの例を示します。

```
CREATE PUBLIC DATABASE LINK "HQ.MYCOMPANY.COM" USING
(DESCRIPTION=
   (ADDRESS=(PROTOCOL=TCP)(HOST=server1)(PORT=1521))
   (CONNECT_DATA=(SID=hqdb)(SERVER=DEDICATED)))
```

「Setup Wizard」はサービス名自体ではなくサービス名の説明を使用しますが、これは、tnsnames.ora ファイル内の情報がサイトごとに異なる可能性があるからです。たとえば、「Setup Wizard」がサービス名のみを使用し、サービス名の説明を使用しなかった場合は、全サイトに同一サービス名が存在し、全サイトの tnsnames.ora ファイル内の情報が同じであることをユーザーが確認する必要があります。レプリケーション・マネージメント・ツールではこの要件をチェックできません。

「Setup Wizard」は、サービス名の説明を使用することで、データベース・リンクが全レプリケーション・サイトで有効なことを保証します。このタイプのデータベース・リンクの弱点は、まれにデータベースのサービス名の説明が変更された場合に、データベース・リンクを削除して再作成が必要になることです。説明ではなくサービス名のみを使用してデータベース・リンクを作成した場合は、全サイトで tnsnames.ora ファイルを変更してもデータベース・リンクをそのまま使用できます。

注意：「Setup Wizard」のデフォルト動作は、このウィザードのカスタマイズ画面を編集して上書きできます。

接続修飾子 接続修飾子を使用すると、同じリモート・データベースにリンクされている複数のデータベース・リンクに異なるパスを指定して接続を確立できます。たとえば、データベース ny では、ny.world という名前の 2 つのパブリック・データベース・リンクに異なるパスを指定して、リモート・データベースに接続できます。

- ny.world@ethernet: イーサネット・リンクを使用して ny に接続するリンクです。
- ny.world@modem: モデム・リンクを使用して ny に接続する別のリンクです。

レプリケーションでは、接続修飾子の使用により、複数のマスター・グループの伝播特性をより厳密に制御できます。たとえば、各マスター・サイトに 3 つの別のマスター・グループがあり、接続修飾子を指定していないと想定します。この場合、遅延トランザクション・キューの伝播のスケジューリング特性はどのマスター・グループも同じです。この 3 つのマスター・グループのうち、1 つのマスター・グループが遅延トランザクションを 1 時間に 1 回伝播し、残りの 2 つのマスター・グループが 1 日に 1 回伝播したとすると、コストが高くなる可能性があります。

接続修飾子をマスター・グループと対応付けると、遅延トランザクション・キューの伝播のスケジューリング特性を、前述のようなデータベース・レベルではなくマスター・グループ・レベルで個別に定義できます。

> **関連項目：** データベース・リンク用の接続修飾子の定義の詳細は、『Oracle9*i* データベース管理者ガイド』を参照してください。

新規マスター・グループを作成するときに、そのグループに対応するすべてのスケジュール・リンクに 1 つの接続修飾子を使用するよう指定できます。ただし、マスター・グループに接続修飾子を使用すると、各マスター・サイトで接続修飾子を使用してデータベース・リンクを作成しないと情報が伝播されません。マスター・グループを作成した後は、グループの接続修飾子の削除、追加または変更はできません。

注意： 接続修飾されたリンクと複数のマスター・グループを使用するマルチマスター環境でのトランザクションの一貫性を保持するため、接続修飾子が異なる複数グループのレプリケーション・オブジェクトを 1 つのトランザクションで操作できません。

注意： 接続修飾子を使用する場合は、すべてのマスター・サイトの OPEN_LINKS 初期化パラメータの値を大きくすることが必要な場合があります。デフォルトでは、プロセスごとのオープン・リンク数は 4 つです。用途に応じて、必要な値を見積もってください。OPEN_LINKS の詳細は、6-4 ページの「初期化パラメータ」および『Oracle9*i* データベース・リファレンス』を参照してください。

レプリケーション・オブジェクト

レプリケーション環境の中で最も理解しやすいのは、レプリケート・オブジェクト自体です。レプリケート・オブジェクトの中では、レプリケート表がレプリケーション環境の基盤です。次の項では、関連するデータベース・オブジェクトのレプリケーションについて説明します。特に、次のタイプのデータベース・オブジェクトをレプリケートする場合の利点および制限事項を中心に説明します。

- 表
- 索引
- パッケージおよびその本体
- プロシージャおよびファンクション
- ユーザー定義型および型の本体
- トリガー
- ビュー、オブジェクト・ビューおよびシノニム
- 索引タイプ
- ユーザー定義演算子

表 ほとんどの場合、レプリケート表がレプリケーション環境の基盤です。レプリケーション用の表を選択し、レプリケーション・サポートを生成すると、表に対して DML が適用されているかどうかを検出するために、内部トリガーによって監視が行われます。

関連項目： 2-28 ページ「内部トリガー」

表をレプリケートするときは、表構造と表データをリモート・データ・サイトにレプリケートすることも、表構造のみをレプリケートすることもできます。また、同じ名前と構造の表がターゲットのレプリケーション・サイトにすでに存在する場合は、レプリケーション環境内の既存のオブジェクトを使用することもできます。

注意 :

- 自己参照型整合性制約がある表では、正しい順序で削除が実行される保証はありません。このような表で削除を実行する場合は、プロシージャ・レプリケーションを使用します。詳細は、『Oracle9*i* レプリケーション・マネージメント API リファレンス』を参照してください。
- 循環依存性を持つ表または自己参照型制約のある表を含むマスター・グループに新規マスター・サイトを追加する場合は、新規マスター・サイトで表定義を再作成し、データを手動でロードする必要があります。循環依存性の例としては、「表 A には表 B に対する外部キー制約があり、表 B には表 A に対する外部キー制約がある」などがあります。

表レプリケーションのその他の用途

表のレプリケーションは、表データに加えられたすべての変更をレプリケーション環境に関係するすべてのサイトにレプリケートすることを目的としていますが、これ以外の用途もあります。

オブジェクトおよびデータのトランスポート : オブジェクトがターゲットの接続先サイトにレプリケートされると、それ以降レプリケーション・サポートは自動的には生成されません。オブジェクトとデータのトランスポートにより、オブジェクトとデータをリモートの接続先に簡単に配布できます。レプリケーション・オブジェクトの削除およびレプリケーション・サポートの生成を行っていない場合は、表（またはその他のオブジェクト）およびデータはリモートの接続先サイトに残り、リモートの接続先サイトの変更は一切レプリケートされません。この方法により、標準のデータベース環境とデータ・セットを新しいデータベース環境に配布できます。

オブジェクトのトランスポート : 同様に、データをコピーせずにターゲットの接続先サイトに表をレプリケートできます。この方法により、接続先サイトにオブジェクトが作成されますが、データは移入されません。したがって、空のデータベース環境を即時に配布できます。

索引 レプリケーション用の表が選択され、リモート接続先サイトで作成されると、表内での制約の施行に使用する索引がリモート・サイトで自動的に作成されます。ただし、パフォーマンス上の理由から索引を使用する場合は、レプリケーション用の索引を明示的に選択して、レプリケーション環境に関係する他のマスター・サイトに作成する必要があります。索引が別のサイトにレプリケートされると、ローカルで作成されたかのように動作します。索引のレプリケーション・サポートを生成する必要はありません。

Oracle ではドメイン索引のレプリケーションをサポートしています。ドメイン索引の記憶表の定義はレプリケートできますが、記憶表には通常は `ROWID` 情報が含まれているため、記憶表自体はレプリケートできません。

関連項目：

- 外部キー列に対する索引のレプリケーションの詳細は、6-2 ページの「外部キー」を参照してください。
- 拡張索引の詳細は、『Oracle9*i* Data Cartridge Developer's Guide』を参照してください。

パッケージおよびその本体 レプリケーション用のパッケージとパッケージ本体を選択し、必要なレプリケーション・サポートを生成すると、プロシージャ・レプリケーションを実行できます。プロシージャ・レプリケーションを使用すると、レプリケーション環境でシリアルに実行されて大量の行を処理する、大規模なバッチ指向操作のパフォーマンスが向上します。

レプリケーション・サポートを使用するプロシージャのパラメータはすべて、`IN` パラメータであることが必要です。`OUT` モードおよび `IN/OUT` モードはサポートされていません。これらのパラメータには次のデータ型がサポートされています。

- `VARCHAR2`
- `NVARCHAR2`
- `NUMBER`
- `DATE`
- `RAW`
- `ROWID`
- `CHAR`
- `NCHAR`
- バイナリ・ラージ・オブジェクト（Binary Large Object: `BLOB`）
- キャラクタ・ラージ・オブジェクト（Character Large Object: `CLOB`）
- 各国語キャラクタ・ラージ・オブジェクト（National Character Large Object: `NCLOB`）
- ユーザー定義データ型

レプリケート・プロシージャは、パッケージ内で宣言する必要があります。スタンドアロン・プロシージャにはレプリケーション・サポートはありません。

関連項目： プロシージャ・レプリケーションの使用方法の詳細は、『Oracle9*i* レプリケーション・マネージメント API リファレンス』を参照してください。

注意： 2-24 ページの「表レプリケーションのその他の用途」で説明した概念と同様に、レプリケーション・サポートを生成せずにレプリケーション用のパッケージおよびパッケージ本体を選択すると、このオブジェクトをリモート・サイトに簡単に配布できます。ただしその場合、パッケージに対するコールはレプリケートされません。

プロシージャおよびファンクション パッケージの一部として宣言されていないプロシージャおよびファンクションは、レプリケーション・サポートされません。スタンドアロンのプロシージャとファンクションを使用してプロシージャ・レプリケーション環境を作成できませんが、レプリケーションを使用してこれらのスタンドアロンのプロシージャとファンクションをレプリケーション環境の各サイトに配布することはできます。スタンドアロンのプロシージャまたはファンクションがレプリケーションを使用してリモート・サイトで作成されると、作成されたオブジェクトはレプリケーション・サポートされず、ローカルで作成されたかのように動作します。

ユーザー定義型および型の本体 ユーザー定義型を含むスキーマ・オブジェクトをレプリケートするには、全レプリケーション・サイトに同一のユーザー定義型が存在する必要があります。

関連項目： 2-12 ページ「ユーザー定義型のレプリケーション」

トリガー 各マスター・サイトにアプリケーション・ロジックまたはデータベース・ロジックがあることを確認するために、レプリケーション用のトリガーを選択できます。トリガーのレプリケーションの重要な例として、DML が表に適用されたときに表に自動的にタイムスタンプを挿入するトリガーのレプリケーションがあります。

トリガーが再起動されないようにするには、API コールをトリガーに挿入して、トリガーがローカル・コールまたはリモート・コールによって起動されているかどうかを検出することが重要です。これにより、トリガーが行を更新して、その更新が別のマスターサイトに伝播されたときトリガー再起動の原因となる状況を回避できます。次のコード例の 5 行目を見てください。

```
1) CREATE OR REPLACE TRIGGER hr.insert_time
2)    BEFORE
3)       INSERT OR UPDATE ON hr.employees FOR EACH ROW
4)    BEGIN
5)       IF DBMS_REPUTIL.FROM_REMOTE = FALSE THEN
6)          :NEW.TIMESTAMP := SYSDATE;
7)       END IF;
8)    END;
```

DBMS_REPUTIL.FROM_REMOTE ファンクションによって挿入または更新がローカルで開始されたと判断された場合、定義されているアクション（タイムスタンプ割当て）が発生します。挿入または更新がリモート・サイトから開始されたと判断された場合は、タイムスタンプ値は割り当てられません。

> **関連項目：** レプリケート・トリガーの作成方法の詳細は、『Oracle9*i* レプリケーション・マネージメント API リファレンス』を参照してください。

ビュー、オブジェクト・ビューおよびシノニム ビュー、オブジェクト・ビューまたはシノニムをレプリケーション環境に関係する他のマスター・サイトに配布するには、レプリケーションを使用します。オブジェクトが他のサイトにレプリケートされると、そのオブジェクトはローカルで作成されたかのように動作します。変更を取得するためにオブジェクトを監視する内部トリガーまたはパッケージはありません。ただし、オブジェクトはレプリケート・オブジェクトなので、レプリケーション・マネージメント・ツールまたはレプリケーション・マネージメント API を使用して削除または変更できます。

索引タイプ Oracle では索引タイプのレプリケーションをサポートしています。レプリケーション・マネージメント・ツールまたは DBMS_REPCAT パッケージ内の CREATE_MASTER_REPOBJECT プロシージャを使用して、索引タイプの実装に使用するタイプとタイプ本体のファンクションを明示的にレプリケートする必要があります。

> **関連項目：** 拡張索引の詳細は、『Oracle9*i* Data Cartridge Developer's Guide』を参照してください。

ユーザー定義演算子 オブジェクト指向アプリケーションの開発者は、ユーザー定義演算子と呼ばれるドメイン固有の演算子（Contains、Within_Distance、Similar など）を使用して、組込みのリレーショナル演算子（+、-、/、*、LIKE など）のリストを拡張できます。ユーザー定義演算子をレプリケーション環境に関係する他のマスター・サイトに配布するには、レプリケーションを使用します。オブジェクトが他のサイトにレプリケートされると、演算子はローカルで作成されたかのように動作します。変更を取得するためにオブジェクトを監視する内部トリガーまたはパッケージはありません。ただし、レプリケート・オブジェクトであるため、レプリケーション・マネージメント API を使用して削除または変更できます。

> **関連項目：** 『Oracle9*i* Data Cartridge Developer's Guide』

順序のレプリケーションの代替方法

異なるデータベースの 2 つの順序で同じ値が生成されることがあるため、順序のレプリケーションはサポートしていません。

順序のレプリケーションの代替方法が 3 つあり、これらの方法を使用すると、一意の値の生成と一意性のデータ競合が確実に回避されます。次の SELECT 文を実行すれば、一意識別子を取り出せます。

```
SELECT SYS_GUID() OID FROM DUAL;
```

この SQL 演算子は、グローバルに一意な 16 バイトの識別子を返します。この値は、タイムスタンプと日付スタンプおよびマシン識別子を使用してグローバルに一意な識別子を生成するアルゴリズムに基づいています。グローバルに一意な識別子は、次の形式で表示されます。

```
4595EF13AB785E73E03400400B40F58B
```

`sys_guid()` ファンクションを使用する第 2 の方法として、各マスター・サイトに順序を作成し、サイト名（またはその他のグローバルに一意な値）とローカルの順序を連結する方法があります。この方法は、順序の値の重複や、2-10 ページの「競合解消の概念」で説明している挿入競合の発生を回避するのに役立ちます。

さらに、レプリケーション環境内で一意の値を生成するように各マスター・サイトで順序を作成する方法があります。これを実現するには、`CREATE SEQUENCE` 文で開始値、増分値および最大値の組合せを使用します。たとえば、次のように設定すると想定します。

パラメータ	マスター・サイト A	マスター・サイト B	マスター・サイト C
START WITH	1	3	5
INCREMENT BY	10	10	10
範囲の例	1, 11, 21, 31, 41,...	3, 13, 23, 33, 43,...	5, 15, 25, 35, 45,...

同様の方法を使用して、各サイトに対して一意の範囲を生成する `START WITH` と `MAXVALUE` を指定することにより、各マスター・サイトごとに異なる範囲を定義できます。

内部トリガー

内部トリガーは、レプリケート・データの更新に関する情報の取得および格納に使用します。内部トリガーによって、ローカル・サイトで加えられたデータ変更をリモート・サイトで再生成するリモート・プロシージャ・コール（RPC）が作成されます。遅延 RPC は、遅延トランザクション・キューに格納され、レプリケーション環境に関係する他のマスター・サイトに伝播されます。データ・レプリケーションをサポートする内部トリガーは、Oracle サーバー実行モジュール内の必須コンポーネントです。したがって、システム・リソースを最小限に使用して、レプリケート・データへの更新内容を高速に取得し、格納できます。

遅延トランザクション

前述の内部トリガーによって生成される RPC を伝播（送信および実行）することにより、データ・レプリケーション情報が転送されます。この RPC は、遅延トランザクション・キューに格納されます。各 RPC には、接続先サイトでの内部プロシージャの実行コマンドのみでなく、ターゲット・サイトにレプリケートされるデータも含まれています。Oracle では分散トランザクション・プロトコルを使用しているため、グローバルなデータベース整合性が自動的に保護され、データの耐障害性が保証されます。

内部プロシージャ

内部トリガーによって作成された遅延 RPC がレプリケーション環境に関係する他のマスター・サイトに伝播されると、遅延 RPC は接続先サイトの内部プロシージャによってリモート・サイトで適用されます。この内部プロシージャは、表のレプリケーション・サポートを生成するときに自動的にアクティブ化されます。また、この内部プロシージャは、発行元サイトの遅延トランザクション・キューから受信される RPC に基づいて実行されます。

キュー

次のキューは、Advanced Replication によって生成されるトランザクションを管理します。

遅延トランザクション・キュー このキューには、マスター・グループ内の別の接続先に伝播されるトランザクション（たとえば、DML）が格納されます。遅延伝播用に、サイトの遅延トランザクション・キューの内部トリガーによって生成される RPC が格納されます。また、トランザクションを構成するすべての RPC がトランザクションとして伝播されてリモートで適用されるように、開始トランザクションに関する情報も記録されます。Oracle のレプリケーション機能では、Oracle Advanced Queuing のメカニズムを使用して遅延トランザクション・キューが実装されます。

注意： `ENABLE RESTRICTED SESSION` 句を指定した SQL 文 `ALTER SYSTEM` で制限付きセッションを使用可能にすると、遅延トランザクションは伝播されません。制限付きセッションが使用禁止になっている場合は、伝播されます。

エラー・キュー このキューには、ローカル・サイトでの適用に失敗した遅延トランザクションに関する情報が格納されます。エラー・キューにはレプリケーション環境のその他の各マスター・サイトのエラー情報は表示されません。エラー状態が解消されると、トランザクションを再度実行するか、エラー・キューからトランザクションを削除できます。

ジョブ・キュー 伝播プロセスは、**ジョブ・キュー・メカニズム**および**遅延トランザクション**を使用して管理されます。各サーバーには、ローカル・ジョブ・キューがあります。サーバーのジョブ・キューは、ジョブの実行を要求する PL/SQL コールやジョブの実行時期など、ローカル・ジョブに関する情報を格納するデータベース表です。レプリケーション環境の一般的なジョブには、遅延トランザクションをリモートのマスター・サイトにプッシュするジョブ、遅延トランザクション・キューから適用済みのトランザクションをパージするジョブおよびマテリアライズド・ビューのリフレッシュ・グループをリフレッシュするジョブがあります。

管理メカニズム

通常はレプリケーション環境をサポートする管理作業を処理するために、いくつかのメカニズムが必要です。このメカニズムを使用すると、レプリケーション環境をオンまたはオフにしたり、レプリケーション環境の作成時または変更時に生成される管理作業を監視できます。

レプリケーションの操作モード

レプリケーション環境の操作モードは 3 つあります。

通常 通常モードのレプリケーション環境では、レプリケーションを使用できます。レプリケーション環境はこのモードで実行されます。レプリケート・オブジェクトに対してトランザクションを実行でき、トランザクションは適切に伝播されます。

静止 静止モードは、レプリケーション環境が通常モードから静止中モードに移行する段階のモードです。レプリケーション環境の静止中は、ユーザーは、レプリケート・オブジェクトに対してトランザクションを実行できませんが、既存の遅延トランザクションは伝播されます。静止表に対する問合せはできます。各接続先へのすべての遅延トランザクションが正しく伝播終了すると、レプリケーション環境は静止中モードに変わります。

静止中 静止中モードのレプリケーション環境は、通常のレプリケーションが使用禁止になっており、主に管理目的（レプリケート・オブジェクトの追加や削除など）に使用されている状態と考えられます。このモードでは、レプリケーションが停止されます。静止中状態では、レプリケーションをオフにしないかぎり、ユーザーは静止中のマスター・グループのレプリケート・オブジェクトに対してトランザクションを実行できません。レプリケーションをオフにすると、レプリケーションが再開されたときにデータの不一致が発生することがあります。トランザクションには、レプリケート表に対する DML またはレプリケート・プロシージャのラッパーの実行が含まれます。マスター表が静止中の場合は、そのマスター表に基づくマテリアライズド・ビューからターゲットのマスター表に変更内容を伝播できませんが、マテリアライズド・ビューをローカルで変更することはできます。

レプリケーション環境は、マスター・グループ・レベルで静止します。マスター・グループに含まれているすべてのマスター・サイトが影響を受けます。マスター・グループが静止中状態になると、遅延トランザクション・キューのトランザクションが他のマスター・サイトに正常に伝播されたか、エラー・キューに入れられたことを確認できます。ユーザーは静止中のマスター・グループに属する表に対しては、引き続き問合せを実行できます。

1 つのマスター・グループを静止しても、その他のマスター・グループには影響しません。通常モードのマスター・グループでは、その他のマスター・グループの静止中も更新処理を継続できます。

レプリケーション・モードの制御

レプリケーションの 3 つの操作モードは、中断および再開の 2 つのメカニズムで制御されます（静止モードは通常モードから静止中モードへの移行段階のモードです）。

中断 中断メカニズムを実行すると、マスター・グループのレプリケーション操作モードが、通常から静止中への移行モードである静止モードになります。遅延トランザクション・キューに、マスター・グループへ未伝播の遅延トランザクションがないときは、レプリケーション環境は静止中モードに変わります。

中断メカニズムを実行できるのは、レプリケーション環境が通常モードの場合のみです。レプリケーション環境を変更する必要がある場合は、中断メカニズムを実行します。

再開 再開メカニズムを実行すると、マスター・グループのレプリケーション操作モードが静止中モードから通常モードに変わります。レプリケーション環境で管理作業（たとえば、レプリケート・オブジェクトの追加）を実行していた場合は、再開メカニズムを実行する前に管理要求キュー（`DBA_REPCATLOG`）が空であることを確認する必要があります。

管理要求

レプリケーション環境を構成および管理するために、関係する各サーバーは Oracle のレプリケーション・マネージメント API を使用します。サーバーのレプリケーション・マネージメント API は、レプリケーション機能を構成するために管理者が使用できるプロシージャおよびファンクションをカプセル化した PL/SQL パッケージのセットです。また、レプリケーション・マネージメント・ツールでは、各サイトのレプリケーション・マネージメント API のプロシージャおよびファンクションを使用して作業を実行します。

管理要求とは、Oracle のレプリケーション・マネージメント API のプロシージャまたはファンクションをコールすることです。たとえば、レプリケーション・マネージメント・ツールを使用して新しいマスター・グループを作成するときに、このツールは、`DBMS_REPCAT.CREATE_MASTER_REPGROUP` プロシージャをコールして作業を完了させます。管理要求には、追加のレプリケーション・マネージメント API コールを生成して要求を処理するものもあります。

管理要求のメカニズム レプリケーション・マネージメント・ツールを使用するか、DBMS_REPCAT パッケージのプロシージャをコールしてレプリケーション・システムを管理する場合、内部メカニズムによって要求は同期ブロードキャストされます。なんらかの理由で同期でのブロードキャストが失敗した場合は、エラー・メッセージが戻され、それを含むトランザクションがロールバックされます。

Oracle サーバーが管理要求を受信すると、DBA_REPCATLOG ビューに要求が記録され、対応する DDL 文が DBA_REPCATLOG ビューの子表に記録されます。マスター・サイトのマスター・グループの管理要求を表示すると、別のマスター・サイトからのコールバックを待機している要求が含まれる場合があります。このような要求は、AWAIT_CALLBACK 要求と呼ばれます。DBA_REPCATLOG ビュー内の管理要求がすべて適用され、エラーがすべて解決されるまで、マスター・レプリケーション・アクティビティは再開できません。

レプリケーション・マネージメント・ツールを使用してレプリケーション・グループの管理要求を作成するたびに、そのグループに対してジョブが存在しない場合はローカル・ジョブ・キューにジョブが自動的に挿入されます。このジョブは定期的に DBMS_REPCAT.DO_DEFERRED_REPCAT_ADMIN プロシージャを実行します。要求を同期ブロードキャストすると、レプリケートされた変更を各マスター・サイトに適用するためにこのジョブが即時に開始されます。

エラーがない場合、バックグラウンド・プロセスがジョブ実行用に使用可能なときは DO_DEFERRED_REPCAT_ADMIN が実行されます。バックグラウンド・プロセスの実行頻度は自動的に判断されます。未処理ジョブを実行するために十分な数のバックグラウンド・プロセスを使用できない場合、処理が遅れることがあります。

注意： サイトで JOB_QUEUE_PROCESSES がゼロに設定されている場合は、そのサイトのすべてのグループに対して管理要求を手動で適用する必要があります。

関連項目： JOB_QUEUE_PROCESSES の詳細は、6-4 ページの「初期化パラメータ」および『Oracle9*i* データベース・リファレンス』を参照してください。

マスター・サイトで DO_DEFERRED_REPCAT_ADMIN をコールするたびに、DBA_REPCATLOG ビューがチェックされ、実行が必要な要求があるかどうかが確認されます。管理要求が 1 つでも存在すると、要求が適用され、ローカル・ビューが適切に更新されます。このイベントは、各マスター・サイトで非同期に起こります。

DO_DEFERRED_REPCAT_ADMIN は、所定の順序でローカルの管理要求を実行します。DO_DEFERRED_REPCAT_ADMIN がマスター定義サイトではないマスター・サイトで実行される場合、可能な範囲で処理が実行されます。ただし、レプリケート表へのデータの移入などのいくつかの非同期アクティビティでは、マスター定義サイトとの通信が必要になります。通信が不可能な場合は、DO_DEFERRED_REPCAT_ADMIN によって管理要求の実行が停止され、管理要求が順序を無視して実行されるのを防ぎます。マスター定義サイトで管理要求を更新または削除する最終ステップなど、マスター定義サイトとの通信の中には遅延可能なものも

ありますが、そのような通信によって DO_DEFERRED_REPCAT_ADMIN による追加の要求の実行が妨げられることはありません。

各マスター・サイトで管理要求が成功または失敗したことが、各サイトの DBA_REPCATLOG ビューに記述されます。マスター・グループごとに、各管理要求に対応する状態がレプリケーション・マネージメント・ツールに表示されます。最後に、各マスター・サイトは、その管理要求の状態をマスター定義サイトに伝播します。マスター・サイトで要求が正常に完了した場合、マスター定義サイトの DBA_REPCATLOG ビューからサイトのコールバックが削除されます。

すべてのサイトで要求が正常に完了した場合は、マスター定義サイトを含むすべてのサイトの DBA_REPCATLOG ビューのエントリがすべて削除されます。マスター定義サイト以外のサイトで要求が失敗した場合は、マスター・サイトの要求が削除され、マスター定義サイト内の対応する AWAIT_CALLBACK 要求が ERROR ステータスと失敗の理由で更新されます。

変更を同期でブロードキャストすることで、すべてのサイトで変更を確実に認識し、同期が取れた状態に保持できます。後の特定の時点でサイトに変更を適用でき、変更の適用に最も適した時点を自由に選択できます。

オブジェクトにレプリケーション・サポートが必要な場合、オブジェクトの変更後にレプリケーション・サポートを再生成する必要があります。Oracle では、内部トリガーをアクティブ化し、パッケージを再生成して、すべてのマスター・サイトにある変更後のオブジェクトのレプリケーションをサポートしています。

注意： エラーなしにこれらのプロシージャを完了させるには DDL をマスター定義サイトで正常に適用する必要がありますが、これによって、DDL が各マスター・サイトで正常に適用されることが保証されるわけではありません。レプリケーション・マネージメント・ツールにより、すべての管理要求の状態が表示されます。また、DBA_REPCATLOG ビューには、一時的な状態および要求によって生成される非同期のエラー・メッセージが含まれます。

DDL 変更によって影響を受けるマテリアライズド・ビュー・サイトは、そのマテリアライズド・ビュー・サイトのリフレッシュを次に実行するときに更新されます。すべてのマスター・サイトが他のマスター・サイトと通信できるのに対して、マテリアライズド・ビュー・サイトが通信できるのは、対応付けられたマスター・サイトのみです。

マテリアライズド・ビューのマスターに変更を加えた結果、マテリアライズド・ビューの形式を変更する必要がある場合は、マテリアライズド・ビューを削除してから再作成します。

管理要求キュー

DBA_REPCATLOG ビュー（通常、管理要求キューと呼ばれます）には、レプリケーション環境を管理および変更する管理要求が格納されます。一部の DBMS_REPCAT プロシージャは、実行すると管理要求キューにリストされます。たとえば、既存のマスター・グループにレプリケート表を追加すると、DBMS_REPCAT.CREATE_MASTER_REPOBJECT プロシージャという要求が表示されます。

管理要求キューを表示するには、DBA_REPCATLOG ビューを問い合せるか、レプリケーション・マネージメント・ツールの「Administrative Requests」ダイアログ・ボックスを表示します。

各要求には、要求の状態を表すステータスがあります。状態は、次のとおりです。

- **READY**: READY 状態は、要求を実行する準備が整っている状態を示します。管理要求キューを監視しているときに、ある要求がしばらくの間 READY 状態である場合、その要求の前にコールバック待機中の要求が存在している可能性があります。通常、READY 状態の管理要求は、ジョブが要求を実行するのを待機しています。このような要求は、DBMS_REPCAT パッケージ内の DO_DEFERRED_REPCAT_ADMIN プロシージャを使用して手動で実行できます。
- **AWAIT_CALLBACK**: AWAIT_CALLBACK 状態は、ある要求が別のサイトで実行され、その実行が確認されるまで待機している状態を示します。要求がコールバックを受信すると、その要求は削除されるか、状態が変更されます。要求が正常に適用された場合はその要求キューから削除され、失敗した場合はそのステータスが ERROR に変更されます。この状態は、マスター定義サイトでのみ発生します。
- **ERROR**: 要求を正常に実行できない場合、ERROR 状態になります。エラー番号が ERRNUM 列に表示され、エラー・メッセージが管理要求キューの MESSAGE 列（レプリケーション・マネージメント・ツールを使用している場合は Error 列）に表示されます。

> **注意：** 要求が ERROR 状態にある場合は、エラー番号による説明に従ってエラー状態を解消し、要求を再送信してください。

- **DO_CALLBACK**: マスター・サイトの要求が DO_CALLBACK 状態にある場合、要求が成功したか失敗したかどうかをマスター・サイトからマスター定義サイトに通知する必要があります。この状態は、マスター定義サイト以外のマスター・サイトでのみ発生します。

各マスター・サイトの管理要求キューには、そのマスター・サイトで実行される管理要求のみがリストされています。ただし、マスター・グループのマスター定義サイトには、各マスター・サイトで実行される管理要求がリストされています。DBA は、マスター定義サイトの管理要求キューを使用すると、レプリケーション環境のすべてのマスター・サイトの管理要求を監視できます。

注意： ENABLE RESTRICTED SESSION 句を指定した SQL 文 ALTER SYSTEM で制限付きセッションを使用可能にすると、管理要求は実行されません。制限付きセッションが使用禁止になっている場合は、実行されます。

編成メカニズム

Oracle では、編成メカニズムを使用して前述のマスター・サイトを編成し、管理メカニズムを使用して個別のレプリケーション・グループを作成します。編成メカニズムの中で最も重要なのは、マスター・グループです。追加の編成メカニズムは、レプリケート表の競合解消に使用する列をグループ化するのに役立ちます。

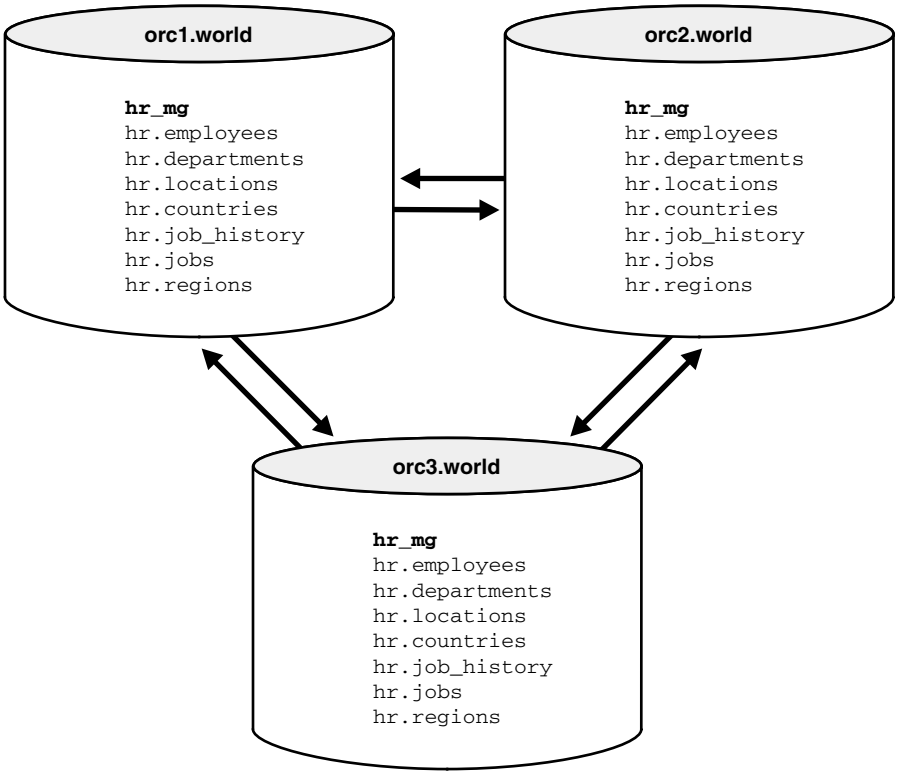
マスター・グループ

レプリケーション環境では、**レプリケーション・グループ**を使用してレプリケーション・オブジェクトを管理します。レプリケーション・グループは、常にトランザクションの一貫性を保つように更新されるレプリケーション・オブジェクトの集合です。

レプリケーション・グループ内に関連するデータベース・オブジェクトを編成すると、多数のオブジェクトをまとめて簡単に管理できます。通常、レプリケーション・グループを作成および使用して、特定のデータベース・アプリケーションのサポートに必要なスキーマ・オブジェクトを編成します。ただし、レプリケーション・グループとスキーマが相互に対応している必要はありません。レプリケーション・グループのオブジェクトは、複数のデータベース・スキーマから作成でき、スキーマには異なるレプリケーション・グループのメンバーであるオブジェクトを入れることができます。制限は、レプリケーション・オブジェクトが 1 つのグループのメンバーにしかなれないことです。

マルチマスター・レプリケーション環境では、レプリケーション・グループは**マスター・グループ**と呼ばれます。異なるサイトの対応するマスター・グループには、レプリケーション・オブジェクトの同一のセットが含まれている必要があります（2-23 ページの「レプリケーション・オブジェクト」を参照してください）。図 2-4 は、各マスター・サイトにレプリケート・オブジェクトの同一のレプリカが含まれているマスター・グループ hr_mg を示します。

図 2-4 すべてのサイトで同じレプリケーション・オブジェクトを含むマスター・グループ hr_mg

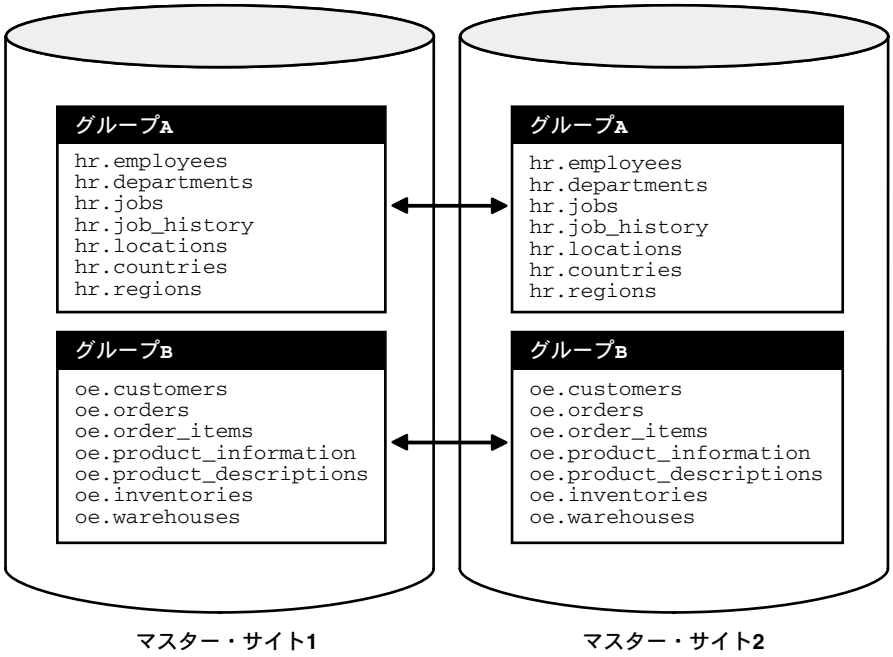


マスター・サイトのマスター・グループ編成は、マテリアライズド・ビュー・サイトでのレプリケーション・オブジェクトの編成に不可欠です。

> **関連項目：** マテリアライズド・ビュー・サイトでの編成メカニズムの詳細は、3-53 ページの「編成メカニズム」を参照してください。

また、各マスター・サイトに複数のレプリケーション・グループが含まれている例を図 2-5 に示します。各マスター・サイトの各グループに含まれているオブジェクトのセットは、同一であることが必要です。

図 2-5 各マスター・サイトで同一のマスター・グループ



列グループ

列グループは、競合解消ルーチンと関係するすべての列をグループ化する編成メカニズムを提供します。グループのいずれかの列で競合が発生した場合、残りの列を使用して競合を解消できます。たとえば、表の列グループに min_price、list_price、cost_price および timestamp というフィールドが含まれており、list_price フィールドで競合が発生した場合、タイムスタンプによる競合解消ルーチンを使用していたと想定すると、timestamp フィールドを使用して競合を解消できます。

表のすべての列を 1 つの列グループにまとめる必要はありません。すべての列を 1 つの列グループにまとめれば設定や管理は簡単になりますが、レプリケート表のパフォーマンスが低下し、データ競合が発生する可能性が高くなることがあります。「パフォーマンス・メカニズム」の項で説明しているように、表の 1 つの列グループで競合が発生した場合、最小限の通信機能によって、表のその他の列グループからはデータが送信されません。したがって、すべての列を 1 つの列グループにまとめると、DBMS_REPCAT パッケージ内の SEND_OLD_VALUES プロシージャおよび COMPARE_OLD_VALUES プロシージャを使用しないかぎり、最小限の通信機能の利点がなくなってしまう可能性があります。

関連項目： 列グループの詳細は、第 5 章「競合解消の概念およびアーキテクチャ」を参照してください。

伝播メカニズム

伝播は、レプリケーション環境のすべてのマスター・サイトにすべてのアクションを送信または配布するメカニズムであり、レプリケーションにおいて不可欠な要素です。

伝播のタイプ

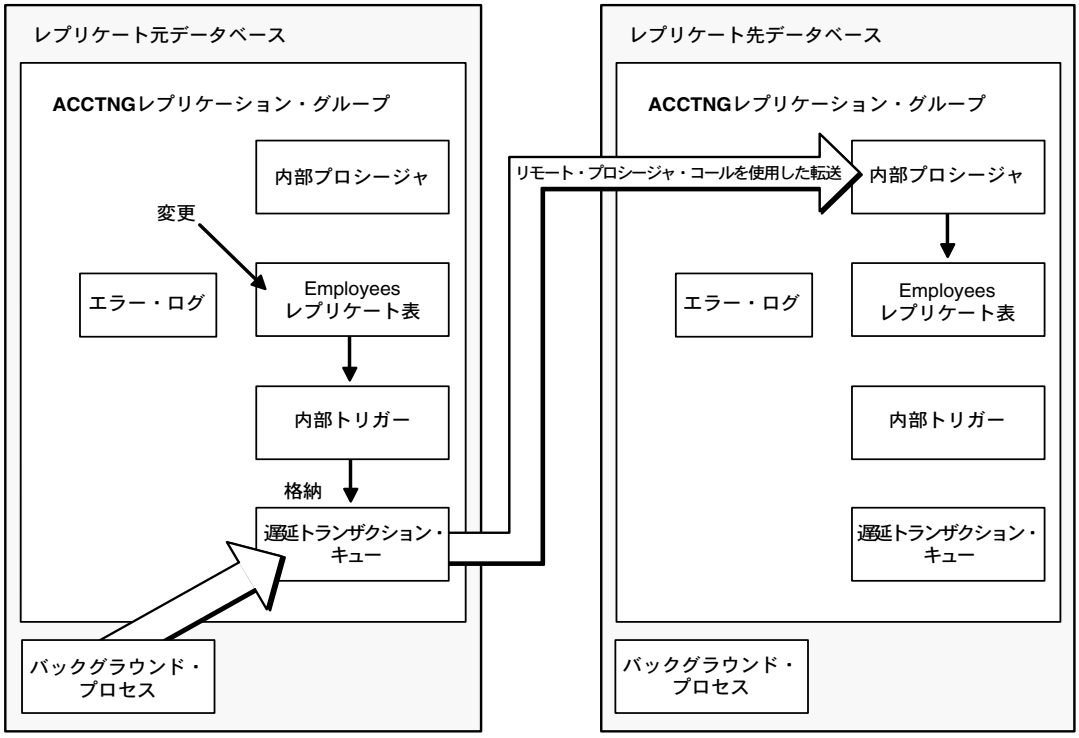
内部トリガーはレプリケート表に適用されるすべての DML を取得するため、レプリケーション環境の他のマスター・サイトに DML を**伝播**（送信）する必要があります。内部トリガーの詳細は、2-28 ページの「内部トリガー」を参照してください。

Advanced Replication では、非同期レプリケーションおよび同期レプリケーションの両方をサポートしています。

非同期 一般的なレプリケーション構成では、**非同期データ・レプリケーション**を使用します。非同期データ・レプリケーションは、アプリケーションで表のローカル・レプリカを更新し、レプリケーション情報をローカル・キューに格納し、その後で他のレプリケーション・サイトにレプリケーション情報を転送する場合に発生します。このため、非同期データ・レプリケーションは**ストア / フォワード・レプリケーション**とも呼ばれます。

図 2-6 に示すように、Oracle では内部トリガー、遅延トランザクション、遅延トランザクション・キューおよびジョブ・キューを使用して、更新可能マテリアライズド・ビューからマスター表に対する伝播の他、レプリケーション環境のマスター・サイト間でも、データレベルの変更を非同期に伝播します。

図 2-6 非同期データ・レプリケーションのメカニズム



同期 Oracle では、特定の要件を持つアプリケーションに対しては同期データ伝播もサポートしています。**同期データ伝播**は、アプリケーションによって表のローカル・レプリカが更新され、同じトランザクション内で同じ表の少なくとも 1 つの他のレプリカも更新されたときに発生します。したがって、同期データ・レプリケーションは**リアルタイム・データ・レプリケーション**とも呼ばれます。アプリケーションでレプリケート・サイトを継続的に同期化する必要があるとき以外は、同期レプリケーションを使用しないでください。

図 2-7 同期データ・レプリケーションのメカニズム

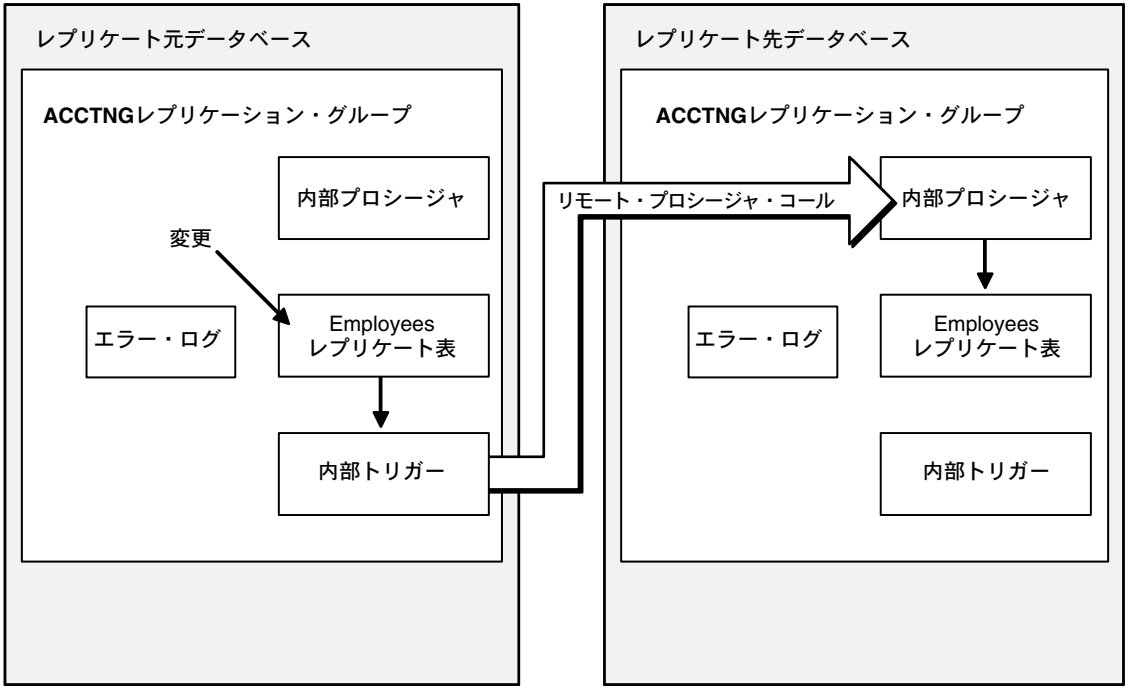


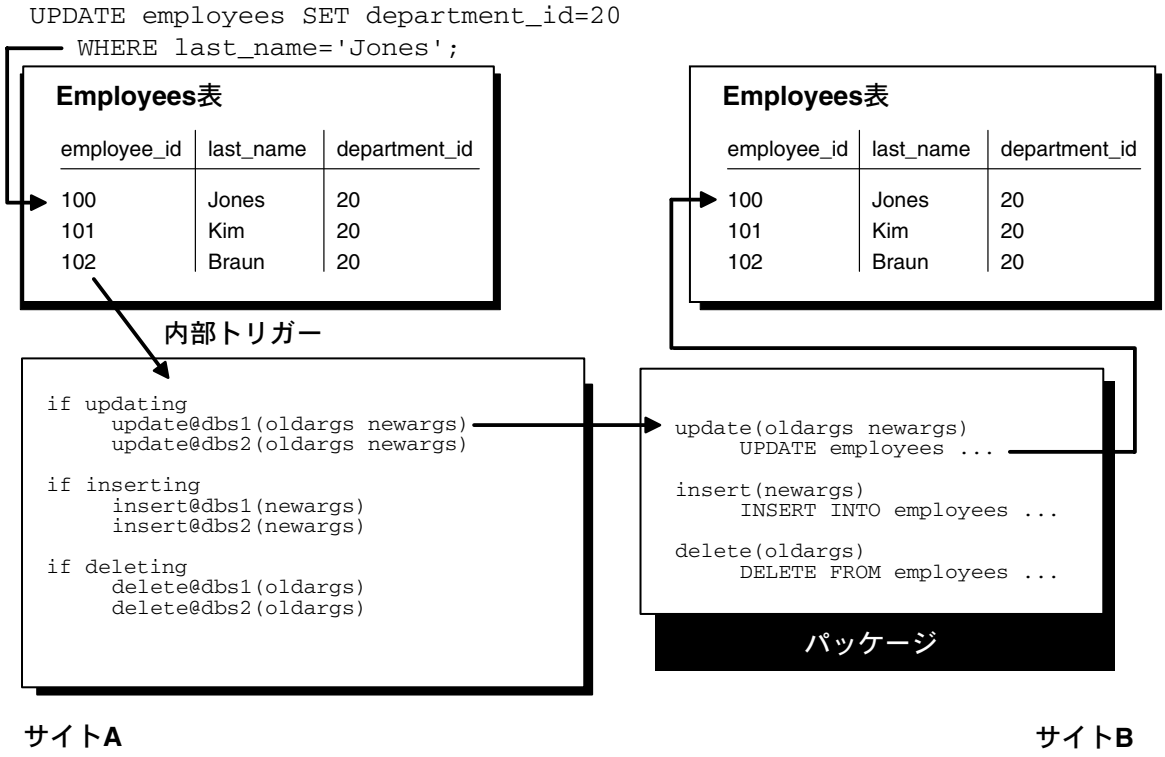
図 2-7 に示すように、同じ内部トリガーを使用して、データレベルの変更をその他のレプリケーション・サイトに非同期にレプリケートして、行レベルの同期データ・レプリケーションをサポートするリモート・プロシージャ・コール（RPC）を生成します。ただし、この RPC の実行は延期されません。かわりに、データ・レプリケーション RPC は、ローカル・レプリカを変更する同一のトランザクションの境界内で実行されます。したがって、レプリケート表を管理する同期リンクされたすべてのサイトで、データレベルの変更が可能な必要があります。可能でない場合は、トランザクションはロールバックされます。

同期データ伝播

図 2-8 に示すように、アプリケーションによってローカルのレプリケート表に DML による変更が加えられ、レプリケーション・グループで行レベルの同期レプリケーションを使用しているときは常に、変更内容は内部トリガーによってレプリケーション環境の他のマスター・サイトに同期伝播されます。アプリケーションでローカルな変更が適用されると、内部トリガーからリモート・マスター・サイトで生成されたプロシージャに対して、レプリケーション・プロパゲータのセキュリティ・コンテキストでコールが発行されます。障害が発生した場合は、すべての分散トランザクションがコミットまたはロールバックされます。

関連項目： 分散トランザクションの詳細は、『Oracle9*i* データベース管理者ガイド』参照してください。

図 2-8 行レベルの同期レプリケーションを使用した変更の伝播



制限事項 2 つの異なるサイトで同じ行が同時に更新されるときは、同期レプリケーションで使用されるロック・メカニズムにより、デッドロックが発生することがあります。アプリケーションでレプリケート表への同期更新が実行されると、まずローカル行がロックされ、次に AFTER ROW トリガーを使用して対応するリモート行がロックされます。各サイトでトランザクションがコミットされると、ロックは解放されます。

注意： レプリケーション・データのリアルタイム伝播を使用するレプリケーション・システムは、システム内のすべてのサイトが同時に使用可能な場合にのみ機能するため、システムとネットワークの可用性に大きく依存します。

同期を取ってレプリケートされるトランザクションの接続先 同期レプリケーションのサポートに必要なリモート・プロシージャ・コールは、各オブジェクトの内部トリガーに組み込まれています。レプリケート・オブジェクトのレプリケーション・サポートを生成すると、すべてのマスター・サイトでトリガーがアクティブ化され、新しいサイトに必要なリモート・プロシージャ・コールが追加されます。反対に、マスター・グループからマスター・サイトを削除すると、内部トリガーからコールが削除されます。

競合の検出 マスター・グループのすべてのサイトが互いに同期通信している場合は、アプリケーションでレプリケーションの競合が発生することはありません。ただし、別のサイトに非同期で変更を送信するサイトが 1 つでもあると、レプリケーション環境のいずれかのサイトで競合が発生する可能性があります。

変更が同期伝播されている場合は、エラーが発生し、ロールバックが必要になります。変更内容が非同期で伝播されている場合は、競合が自動的に検出されてエラー・キューでログに記録されるか、適切な解消方法を指定している場合にはその競合が解消されます。

関連項目： 第 5 章「競合解消の概念およびアーキテクチャ」

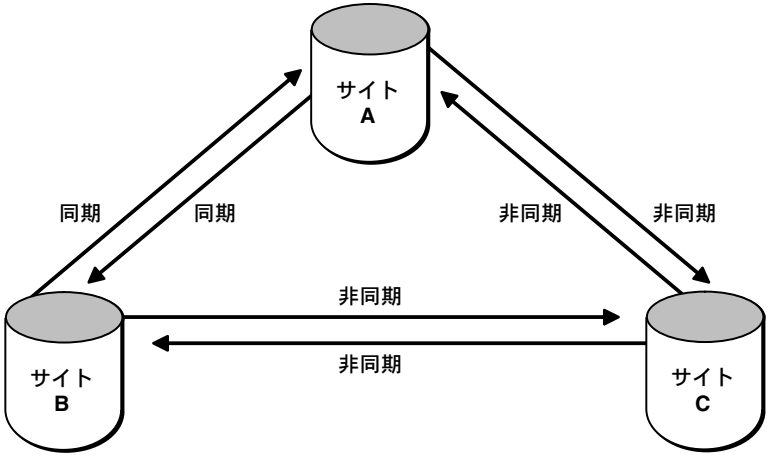
複合モードのマルチマスター・システムについて

状況によっては、一部のマスター・サイトでマスター・グループの変更内容を非同期で伝播し、他のマスター・サイトでは同期伝播するような複合モード環境の設定が必要な場合があります。新規マスター・サイトをデータ伝播モードが異なるグループに追加するときは、その順番が重要となることがあります。

たとえば、A、B および C の 3 つのマスター・サイトがあると想定します。まずマスター定義サイトとしてサイト A を作成し、次に同期伝播モードでサイト B を追加した場合、サイト A からサイト B へも、サイト B からサイト A へも変更は同期を取って送信されます。どのサイトでも遅延トランザクションは作成されないため、どちらかのサイトでリンクをスケジュールする必要はありません。

次に、非同期伝播モードでマスター・サイト C を作成するとします。伝播モードは、図 2-9 に示すようになります。

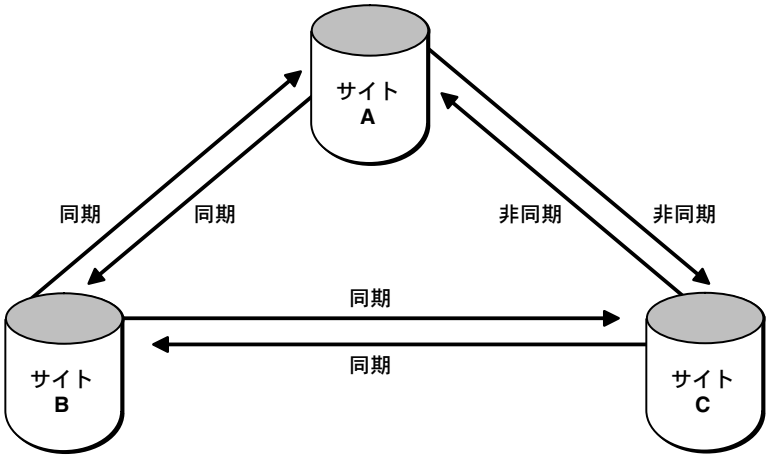
図 2-9 伝播モードの選択



次に、サイト A からサイト C、サイト B からサイト C、サイト C からサイト A およびサイト B への遅延トランザクション・キューの伝播をスケジュールする必要があります。

別の例として、サイト A をマスター定義サイトとして作成し、サイト C を非同期伝播モードで追加した後、サイト B を同期伝播モードで追加する場合を考えてみます。伝播モードは、図 2-10 に示すようになります。

図 2-10 順序に関する考慮事項



複合モードのマルチマスター・システムに新規マスター・サイトを追加するときは、その追加が既存のサイトとのデータ伝播モードに与える影響を考慮してください。

伝播の開始

同期伝播を使用する場合、DML が即時に伝播され、自動的に開始されます。非同期伝播を使用する場合は、2 つの方法で遅延トランザクションを伝播できます。

- **スケジュール・ジョブ :** 通常、設定したスケジュール間隔に従って遅延トランザクションを自動的に伝播するには、スケジュール・ジョブを使用します。
- **手動による伝播 :** ストアド・プロシージャを実行したり、レプリケーション・マネージメント・ツールを使用して、変更を手動で伝播することもできます。遅延トランザクションがジョブ・キューによって自動的に伝播されるまで待機しない場合は、遅延トランザクションを手動で伝播する必要があります。

パフォーマンス・メカニズム

他のエンタープライズ・データベース・ソリューションと同様に、データベース管理者にとってパフォーマンスは常に重要な課題です。Advanced Replication には、レプリケーション環境のパフォーマンスを高めるいくつかのメカニズムがあります。

パラレル伝播

パラレル伝播では、スループットを向上させるため、複数のパラレル伝送ストリームを使用してレプリケート・トランザクションを非同期で伝播します。必要に応じて依存トランザクションの実行が命令され、グローバルなデータベースの整合性が保証されます。

パラレル伝播では、使用可能なパラレル処理のプールが使用されます。これは、パラレル問合せ、パラレル・ロードおよびパラレル・リカバリなどの他のパラレル操作に使用されるプールと同じです。各サーバー・プロセスでは、シングル・ストリームを介してトランザクションが伝播されます。これらのサーバー・プロセスは、パラレル・コーディネータ・プロセスにより制御されます。コーディネータでは、トランザクション依存性を追跡しサーバー・プロセスに作業を割り当て、その進行状況を追跡します。

操作の実行中は、パラレル処理がサーバーのパラレル操作に対応付けられた状態になります。操作が完了すると、サーバー・プロセスは他のパラレル操作の処理に使用できます。たとえば、接続先への遅延トランザクション・キューのパラレル・プッシュが実行されるときは、キューのプッシュに使用されるすべてのパラレル処理は、プッシュが完了するまで占有されます。

サーバー用のパラレル処理のプールを正しく構成するには、レプリケーション・システムの構成に関する問題を考慮する必要があります。

- シリアル伝播を使用するようにすべてのスケジュール・リンクを構成すると、レプリケーション・システムではパラレル処理は使用されません。したがって、レプリケーションを考慮して、パラレル処理のプールのサイズを調整する必要はありません。シリアル伝播は、通常、下位互換性が必要な場合にのみ使用します。

- 1 つ以上のスケジュール・リンクをパラレル伝播を使用するように構成する場合、それぞれのリンクが変更を接続先にプッシュするために使用するパラレル処理の数を考慮する必要があります。さらに、別の操作によって使用されないように、各プッシュでパラレル・サーバーを保持する時間を考慮する必要があります。たとえば、遅延秒数値を大きくして連続的な伝播用にスケジュール・リンクを構成すると、接続先にトランザクションをプッシュするために使用するパラレル処理が保持されます。したがって、サーバーの他のパラレル操作で使用するプロセスを十分に確保するために、対応するデータベース・サーバーのパラレル処理の数を増やす必要があります。

パラレル問合せプロセスのデータベース・サーバーのプールを構成するには、次の初期化パラメータを使用します。

- PARALLEL_AUTOMATIC_TUNING
- PARALLEL_MAX_SERVERS
- PARALLEL_MIN_SERVERS

関連項目：

- 6-4 ページ「初期化パラメータ」
- 『Oracle9*i* データベース・リファレンス』
- 『Oracle9*i* データベース移行ガイド』

パラレル伝播の実装 ほとんどのユーザーの場合、パラレル伝播パラメータ値を 1 に設定すると、十分なパフォーマンスが得られます。1 に設定すると、シリアル伝播ではなく、前項で説明した最適化されたデータの交換方式が使用されます。ただし、一部のユーザーは、パラレル伝播値をさらにチューニングすることが必要な場合があります。

パラレル伝播値をさらにチューニングする場合、次の手順の実行をお薦めします。

1. パラレル伝播値を 1 に設定します。
2. 使用しているデータベース環境をテストし、伝播スループットを慎重に測定します。

   パラレル伝播値が 1 でパフォーマンス目標が達成される場合は、パラレル伝播が実装されたことになるため、残りの手順を実行する必要はありません。

注意： パラレル伝播値を大きくすると、パラレル伝播のパフォーマンスが向上する一方で、追加のパラレル・スレーブのサポートに必要なリソースが増大します。

---

3. パラレル伝播値を 1 に設定した場合よりもスループットを向上させたい場合は、パラレル伝播値を 2 に設定します。
4. 使用しているデータベース環境をテストし、伝播スループットを慎重に測定します。

   通常、伝播値を 2 に設定すると、伝播スループットは低下します。これは、依存トランザクションを使用可能なスレーブに割り当て、追加トランザクションを割り当てる前に必要なコミットを待機しているコーディネータに関連したラウンドトリップ遅延によるものです。

   パラレル伝播値を 4 に設定して手順 3 と 4 を繰り返し、さらに、パラレル伝播値を 8 に設定して同じ手順を繰り返します。それでもスループットが改善されない場合は、使用している環境のトランザクションの相互依存性が非常に高いことが考えられます。パラレル伝播値を 1 に下げ、手順 5 に進んでください。

   関連項目： トランザクション依存性を下げるテクニックの詳細は、2-46 ページの「パラレル伝播のチューニング」を参照してください。

   パラレル伝播値を 2、4 または 8 のいずれかに設定することでパフォーマンスが向上した場合は、トランザクションの相互依存性は低いといえます。パラレル伝播値は 9 以上の値に設定してもかまいません。必ず使用している環境を完全にテストしてください。パラレル化が増大すれば、追加のパラレル・スレーブのサポートに必要なリソースも増大することに注意してください。
5. テスト結果に基づき、使用している環境でパフォーマンスが最善になる値にパラレル伝播を設定します。

パラレル伝播のチューニング パラレル伝播のパフォーマンス上の利点を最大限に活用するには、作成する依存トランザクションの数を減らします。トランザクションは、依存関係にあるトランザクションがすべてがコミットされるまで開始できません。

依存トランザクションの数を減らすには、次のようにします。

- 可能であれば、トランザクションを小さくします（つまり、自律性を損なわない範囲で、コミットの回数を増やします）。
- 挿入を受け取る各表の空きリスト数を増やします。
- ホットスポット（頻繁に変更される行。同じ行に対して操作が行われる場合、そのトランザクションはシリアル化されます）を回避します。たとえば、行のカウンタを使用して手動で増分するのではなく、Oracle の順序を使用します。
- 行レベルの依存性の追跡を行うようにします。

  関連項目： 2-51 ページ「パラレル化を改善するための行レベルでの依存性の追跡の使用」

最小限の通信

表のレプリケーション・パフォーマンスを向上させるには、レプリケート表のレプリケーション・サポートを生成するときに、必ず最小限の通信の設定を使用可能にしておきます。

行の更新の競合を検出および解消するには、伝播サイトから受信サイトに新旧の行に関するデータを送信する必要があります。使用している環境によって、更新の競合の検出および解消をサポートするために伝播されるデータの量は異なります。

たとえば、関係するすべてのサイトが Oracle リリース 8.0 以上のデータベースを使用している環境でレプリケート表を作成する場合は、レプリケート表内の変更された各行の競合を検出するために通信する必要があるデータの量を最小化できます。この場合は、次の値が伝播されます。

- 主キー値および変更された各列グループの各列の古い値（変更前の値）
- それぞれの更新された列の新しい値

注意： 挿入された行には古い値はありません。また、削除された行には新しい値はありません。

一般的に、Oracle リリース 8.0 以上のレプリケーション環境では、データ伝播を最小化して、ネットワーク上で送信されるデータの量を減らすことが可能です。それにより、システム全体のパフォーマンスが向上します。

これに対して、レプリケーション環境で Oracle7 のサイトと Oracle リリース 8.0 以上のサイトが混在している場合は、更新の競合を検出するために通信する行データの量を最小化することはできません。この場合、競合を検出するためには、変更された各行の新旧のデータをすべて伝播する必要があります。

関連項目： 競合解消、最小限の通信およびその他の競合解消テクニックの詳細は、5-34 ページの「パフォーマンスのメカニズムと競合の解消」を参照してください。

遅延秒数

パフォーマンスに直接かかわるメカニズムではありませんが、`delay_seconds` パラメータ値を適切に設定すると、遅延トランザクションの伝播のタイミングをより詳細に制御できます。

遅延トランザクションをプッシュする場合は、`SCHEDULE_PUSH` プロシージャまたは `PUSH` ファンクションに `delay_seconds` パラメータを設定します。遅延トランザクションをパージする場合は、`SCHEDULE_PURGE` プロシージャまたは `PURGE` ファンクションに `delay_seconds` パラメータを設定します。これらのプロシージャとファンクションは、`DBMS_DEFER_SYS` パッケージにあります。

delay_seconds パラメータは、ジョブが遅延トランザクション・キューを認識し続ける時間を制御します。delay_seconds パラメータ値を最大限に活用した 2 つの例を次に示します。

delay_seconds = 0（デフォルト）

間隔を 60 分に設定したスケジュール・ジョブが午後 2 時 30 分に起動され、遅延トランザクション・キューをチェックする場合、既存の遅延トランザクションはすべて伝播されます。伝播には 2 分かかるので、ジョブが完了するのは午後 2 時 32 分です。

遅延トランザクションが午後 2 時 34 分にキューに入れられた場合、ジョブが完了しているためこの遅延トランザクションは伝播されません。このシナリオでは、遅延トランザクションは午後 3 時 30 分に伝播されます。

delay_seconds = 300

間隔を 60 分に設定したスケジュール・ジョブが午後 2 時 30 分に起動され、遅延トランザクション・キューをチェックする場合、既存の遅延トランザクションはすべて伝播されます。伝播には 2 分かかるので、ジョブが完了するのは午後 2 時 32 分です。

遅延トランザクションが午後 2 時 34 分にキューに入れられた場合、既存の遅延トランザクションの伝播完了後も 300 秒間（5 分間）はジョブによって遅延トランザクション・キューが認識されるので、この遅延トランザクションは伝播されます。このシナリオでは、この遅延トランザクションは午後 2 時 34 分に伝播されます。

さらに頻繁にジョブが実行されるように設定できないかという疑問が生じます。ジョブを開始および停止すると、ジョブを開始して一定時間認識し続けるよりもはるかに大きいオーバーヘッドが発生します。delay_seconds パラメータ値を使用すると、ジョブの開始および停止に伴うオーバーヘッドを削減できるのみでなく、スケジュール・ジョブのサポートに必要な REDO ロギングの量を減らすことができます。

パフォーマンスにかかわるほとんどの機能と同様に、デメリットもあります。次の理由により、delay_seconds パラメータの長さは常にチェックするようにします。

- **パラレル伝播** : 遅延トランザクション・キューをプッシュするときに使用する各パラレル処理は、その伝播ジョブが完了するまでその他のパラレル・アクティビティでは使用できません。長い delay_seconds 値を定義すると、他の操作がパラレル処理を使用できなくなる可能性があります。パラレル伝播を使用するには、SCHEDULE_PUSH プロシージャまたは PUSH ファンクションの parallelism パラメータを 1 以上に設定します。
- **シリアル伝播** :（パラレル伝播ではなく）シリアル伝播しか使用しない場合、delay_seconds 値を設定するとオープン・セッションが遅延時間中スリープするため、この値を設定しても前述のようなメリットはありません。シリアル伝播を使用するには、SCHEDULE_PUSH プロシージャまたは PUSH ファンクションの parallelism パラメータを 0（ゼロ）に設定します。

- **精度パージ**：DBMS_DEFER_SYS.PURGE プロシージャを使用しているときに purge_method_precise メソッドを指定し、delay_seconds 値に大きい値を定義した場合、指定したパージの実行時にパフォーマンスが低下する可能性があります。purge_method_precise は別の選択肢（purge_method_quick）よりもコストがかかりますが、遅延トランザクションとプロシージャは、正常にプッシュされた後で確実にパージされます。

一般的に、delay_seconds 値を 20 分（パラメータ設定では 1200 秒）を超えて設定してもメリットはありません。

さらに、parallelism パラメータを 0 に設定してシリアル伝播を使用している場合は、delay_seconds 値はあまり大きくは設定しません。パラレル伝播と異なり、シリアル伝播では delay_seconds 値の時間が経過した後にのみキューがチェックされます。シリアル伝播を使用し、delay_seconds を 20 分に設定した場合は、スケジュール・ジョブが 20 分間すべてスリープし、その間に遅延トランザクション・キューに入れられた遅延トランザクションは、20 分経過するまでプッシュされません。したがって、シリアル伝播を使用する場合は、delay_seconds 値を 60 秒以下に設定するようにします。

パラレル伝播値を 20 分に設定する場合、パラレル・プッシュによって 1 分に 1 回チェックされます。リソースを 20 分間ロックしておいてもかまわない場合は、20 分間という比較的長い delay_seconds 値が使用環境で最も効率的と言えます。リソースを 20 分間ロックできない場合は、delay_seconds 値を 10 秒または 20 秒に設定することを考えます。1200 秒に設定した場合よりもジョブの実行回数を多くする必要がありますが、delay_seconds 機能を利用するメリットは残ります（デフォルトの 0 秒に設定した場合と比較した場合）。

レプリケーションの保護メカニズム

マルチマスター・レプリケーション環境では、障害の発生時でも、リモート・サイトに伝播されたトランザクションは消失されず、再伝播されないことが保証されています。トランザクションは次の方法で保護されます。

- 1 つのローカル・トランザクション内で発行された複数のプロシージャ・コールは、1 つのトランザクションとしてリモートで実行されます。
- 伝播の実行中にネットワークまたはリモート・データベースに障害が発生した場合、トランザクションはリモート・サイトでロールバックされ、リモート・データベースが再度アクセス可能になって伝播が正常に完了するまで、レプリケーション元のサイトのローカル・キューに残ります。
- トランザクションがローカル・サイトのキューから削除されるのは、すべての接続先サイトへの伝播および適用が正常に完了してからです。トランザクションがすべての接続先サイトに正常に伝播された後でも、パージ・ジョブによって削除されるまで、トランザクションはキューに残ります。

パラレル伝播の場合は、パラレル操作用に最適化された特殊な分散トランザクション・プロトコルが使用されます。リモート・サイトは、正常に伝播されたトランザクションを記録し、この情報が要求されたときに情報をローカル・サイトに戻します。ローカル・サイトは、この情報を記録し、すべての接続先サイトに伝播されたエントリをローカル・キューからパージします。障害が発生した場合、ローカル・サイトは、適切な時点から伝播を続行できるように、正常に伝播されたトランザクションに関する情報をリモート・サイトに要求します。

注意： 伝播が正常に完了しても、必ずしもトランザクションがリモート・サイトで正常に適用されるとは限りません。解消不能な競合や記憶領域不足などのエラーが発生すると、トランザクションでエラーが発生することがあり、そのエラーはエラー・トランザクションとしてリモート・サイトのログに記録されます。

関連項目：

- 2-44 ページ「パラレル伝播」
- レプリケーション・マネージメント・ツールを使用したエラー・トランザクションの表示および管理の詳細は、レプリケーション・マネージメント・ツールのオンライン・ヘルプを参照してください。

データ伝播の依存性の維持

Oracle は、レプリケート・トランザクションがリモート・サイトに伝播されたときの依存性の順序を維持します。たとえば、次のようなトランザクションがあったとします。

1. トランザクション A が受注を取り消します。
2. トランザクション B が取消しを検出して返金を処理します。

トランザクション B は、ローカル・システムでの受注の取消し（トランザクション A）のコミット済みの更新を参照するので、トランザクション B はトランザクション A に依存しています。

トランザクション A（受注の取消し）が正常に伝播された後で、トランザクション B（返金）が伝播されます。取消しの適用後に返金を処理する更新が適用されます。

パラレル伝播の依存性の追跡 ローカル・システムで新規トランザクションが実行されるときは、次の処理が実行されます。

1. データを更新した最新トランザクションのシステム変更番号（SCN）が記録されます。新規トランザクションはこれを依存 SCN として認識します。SCN はデータ・ブロック・レベルまたは行レベルで記録できます。これについてはこの章で後述します。
2. 依存 SCN 以下の SCN を持つトランザクションが、リモート・システムに正常に伝播されるようにします。

3. 待機中の依存トランザクションが伝播されます。

注意： トランザクション間で依存関係の可能性がないときは、トランザクションはパラレルで伝播されます。

パラレル伝播は、シリアル伝播とは異なる方法でデータ整合性を保持します。シリアル伝播では、依存性を維持するために、ローカル・システム上でコミットするときと同じ順序ですべてのトランザクションが適用されます。パラレル伝播では、依存性を追跡し、依存性があるときはコミットされた順に実行し、依存性がないときはパラレルで実行します。シリアル伝播でもパラレル伝播でも、トランザクション内の実行順序は保持されます。遅延トランザクションは、各サイトで、ローカル・トランザクション内で実行された順序と同じ順序で、リモート・プロシージャ・コールを実行します。

注意： シングル・コーディネータ・プロセスは、リモート・サイトに対する各データベース・リンクに存在しています。同じリモート・サイトに対する各データベース・リンクには、異なる接続修飾子が必要です。

関連項目： 2-22 ページ「接続修飾子」

パラレル化を改善するための行レベルでの依存性の追跡の使用 表を作成するときは、SCN を追跡するために次のオプションを指定できます。

- NOROWDEPENDENCIES。SCN がデータ・ブロック・レベルで追跡されることを指定します（デフォルト）。
- ROWDEPENDENCIES。SCN が表の各行に対して追跡されることを指定します。

表に NOROWDEPENDENCIES 句を使用すると、データ・ブロック SCN はデータ・ブロックに格納されている行の最新の更新を追跡します。その前に更新されている他の行は、同じデータ・ブロックに格納されている可能性がありますが、これらの行がいつ更新されたかを示す情報は、新規 SCN がデータ・ブロック・レベルで適用されたときに失われます。

表に ROWDEPENDENCIES 句を使用すると、1 つのデータ・ブロックに複数の SCN を格納できます。つまり、1 つの SCN がデータ・ブロック内に格納されている行のそれぞれに対する変更を追跡します。同一データ・ブロックに格納されている 2 つの行が別々のトランザクションにより変更された場合、行のそれぞれにはこの変更を追跡する SCN があります。SCN を行レベルで追跡するには、表の各行が 6 バイトの記憶領域を追加使用します。

表に ROWDEPENDENCIES 句を使用するとパラレル伝播が使用可能になり、遅延トランザクション・キューの適用時に、より効率良く依存性を追跡して変更に順序を付けられます。効率が上がることによりパフォーマンスが向上し、レプリケーション環境での拡張性が高まります。

次の問合せを使用すると、現在 ROWDEPENDENCIES 句を使用している表をリストできます。

```
SELECT OWNER, TABLE_NAME FROM DBA_TABLES
  WHERE DEPENDENCIES = 'ENABLED';
```

注意： ROWDEPENDENCIES 句を使用するには、レプリケーション・サイト間の互換性のレベルがリリース 9.0.1 以上である必要があります。互換性のレベルは、COMPATIBLE 初期化パラメータによって制御されます。

関連項目： ROWDEPENDENCIES 句を使用した表の作成方法の詳細は、6-4 ページの「行レベルでの依存性の追跡」を参照してください。

パラレル化を改善するためのトランザクション依存性の最小化 レプリケート表のいくつかで ROWDEPENDENCIES 句を使用していない場合は、トランザクション依存性を最小化することで、これらの表のパラレル伝播のパフォーマンスを改善できます。

この場合、アプリケーション条件によっては、トランザクション間に依存関係を確立して、遅延トランザクションの伝播を強制的にシリアル化できます。複数の非関連トランザクションによってレプリケート表内の同じデータ・ブロックが変更される場合、対応するトランザクションはリモート接続先へシリアル伝播されます。

データ・ブロック・レベルで作成されたトランザクション依存性を最小化するには、データ・ブロックの変更が 1 つまたは少数のデータ・ブロックに集中することを回避します。たとえば、レプリケート表に INSERT アクティビティが頻繁に発生する場合は、表の空きリストを複数作成することで、新しい行の記憶域を複数のデータ・ブロックに分散できます。

多数のトランザクションすべてが同一の小さい表を更新する状態は、できるだけ避けてください。たとえば、複数のトランザクションが 1 つの小さい表に読込みと更新を行い、主キーの順序番号生成をシミュレートするのは、望ましいアプリケーション設計とは言えません。このように設計すると、すべてのトランザクションが同じデータ・ブロックを更新します。より優れたソリューションは、順序を作成し順序番号をキャッシュに格納することで、主キーの生成を最適化し、パラレル伝播のパフォーマンスを改善できます。

競合解消メカニズム

レプリケーション環境の受信マスター・サイトでは、次のような更新の競合、一意性競合および削除の競合が検出されます。

- 主キー列または更新された列グループの列のいずれかで、レプリケート行の古い値（変更前の値）と受信サイトの同じ行の現在の値が異なる場合、受信サイトで更新の競合が検出されます。
- レプリケート行の INSERT または UPDATE の実行中に一意制約違反が発生した場合、受信サイトで一意性競合が検出されます。
- 行の主キーが存在しないために受信サイトで UPDATE 文または DELETE 文の対象の行が見つからない場合、受信サイトで削除の競合が検出されます。

注意： 行の更新の競合を検出および解消するには、伝播サイトから受信サイトに新旧の行に関するデータを送信する必要があります。パフォーマンスを最大にするには、更新の競合の検出および解消サポートに使用するデータの量をチューニングします。詳細は、5-34 ページの「最小限の通信」を参照してください。

競合検出時の行の識別

レプリケーションの競合を正確に検出するには、データ・レプリケーション時に別々のサイトの対応する行を一意に識別し、一致させる必要があります。通常、Oracle のレプリケーション機能では、表の主キーを使用して表内の行を一意に識別します。表に主キーがない場合は、代替キーを指定する必要があります。代替キーとは、データ・レプリケーション時に表内の行を一意に識別するために使用する列または列のセットです。

注意： アプリケーションで表の主キーまたは代替キーを更新しないようにしてください。これにより、行を識別し、レプリケート・データの整合性を保つことができます。

上書きによる競合解消方法

Oracle には、データ競合が検出されたときに競合解消方法を定義するメカニズムがあります。ビルトインの競合解消方法は次のとおりです。

- 最新のタイムスタンプおよび最も古いタイムスタンプ
- 上書きおよび廃棄
- 最大値および最小値
- 加算および平均
- タイムスタンプ
- 優先グループ
- サイト優先順位

ビルトインの競合解消方法ではレプリケーション環境のニーズが満たされない場合、PL/SQL を使用して独自の競合解消方法を作成し、ユーザー定義の競合解消方法として実装できます。競合解消の仕組みの詳細は、第 5 章「競合解消の概念およびアーキテクチャ」を参照してください。

> **関連項目：** レプリケーション・マネージメント・ツールを使用した競合解消の実装方法の詳細は、レプリケーション・マネージメント・ツールのオンラインヘルプを参照してください。レプリケーション・マネージメント API を使用した競合解消の実装方法の詳細は、『Oracle9*i* レプリケーション・マネージメント API リファレンス』を参照してください。

3

マテリアライズド・ビューの概要とアーキテクチャ

この章では、マテリアライズド・ビューの概念およびアーキテクチャについて説明します。この章の内容は、次のとおりです。

- マテリアライズド・ビューの概要
- マテリアライズド・ビューのアーキテクチャ

マテリアライズド・ビューの概要

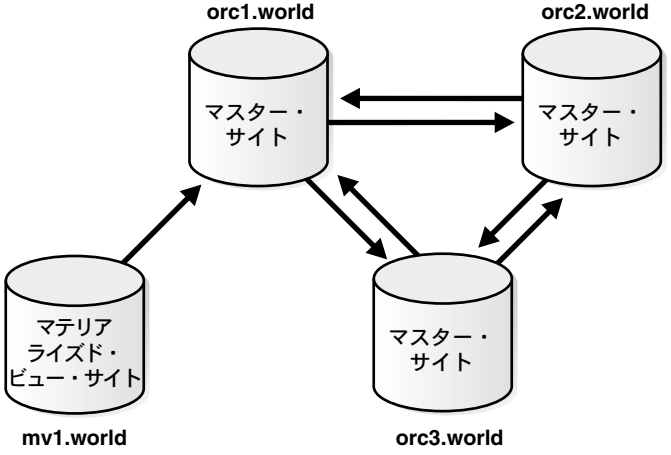
Oracle では、**マテリアライズド・ビュー**（Oracle8*i* ではスナップショットと呼ばれていたもの）を使用して、レプリケーション環境のマスター・サイト以外のサイトにデータをレプリケートし、コストのかかる問合せをデータ・ウェアハウス環境にキャッシュします。この章およびこのマニュアル全体では、レプリケーション環境で使用するマテリアライズド・ビューについて説明します。

関連項目： データ・ウェアハウスでのマテリアライズド・ビューの詳細は、『Oracle9*i* データ・ウェアハウス・ガイド』を参照してください。

マテリアライズド・ビューについて

マテリアライズド・ビューとは、ある一時点におけるターゲット・マスターのレプリカのことです。マスターは、マスター・サイトのマスター表またはマテリアライズド・ビュー・サイトのマスター・マテリアライズド・ビューのいずれかです。マルチマスター・レプリケーションでは、表は他のマスター・サイトによって連続的に更新されますが、マテリアライズド・ビュー・レプリケーションでは、シングル・マスター・サイトまたはマスター・マテリアライズド・ビュー・サイトからバッチ更新（**リフレッシュ**とも呼ばれます）が実行されるたびに、1 つ以上のマスターから更新されます。図 3-1 を参照してください。図 3-1 の矢印は、データベース・リンクを表します。

図 3-1 1 つのマスター・サイトに接続されたマテリアライズド・ビュー



マテリアライズド・ビューが高速リフレッシュされると、Oracle では最後のリフレッシュ以降にマスター表またはマスター・マテリアライズド・ビューに加えられたすべての変更をチェックして、マテリアライズド・ビューに適用される変更があるかどうかを調べる必要があります。したがって、最後のリフレッシュ以降にマスターに対して変更が加えられている場合は、マテリアライズド・ビューのリフレッシュでは変更を適用する時間がかかります。ただし、最後のリフレッシュ以降にマスターに対して何も変更が加えられていない場合は、マテリアライズド・ビューのリフレッシュはすばやく行われます。

マテリアライズド・ビューの使用目的

マテリアライズド・ビューを使用する目的には、次のようなものがあります。

- ネットワーク負荷の軽減
- 大量配布環境の作成
- データのサブセット化の有効化
- モバイル・コンピューティングの有効化

ネットワーク負荷の軽減

ネットワーク負荷の軽減が目的の一つである場合は、マテリアライズド・ビューを使用して会社のデータベースを地域サイトに分散できます。会社全体で 1 つのデータベース・サーバーにアクセスするのではなく、ユーザーの負荷を複数のデータベース・サーバーに分散します。多重化マテリアライズド・ビューを使用すると、他のマテリアライズド・ビューに基づいたマテリアライズド・ビューを作成できます。これにより、クライアントはマスター・サイトのかわりにマテリアライズド・ビューにアクセスできるため、ユーザーの負荷をさらに分散させることができます。レプリケートされるデータ量を減らすために、マテリアライズド・ビューをマスター表またはマスター・マテリアライズド・ビューのサブセットにできます。

マルチマスター・レプリケーションでも会社のデータベースが複数のサイトに分散されますが、マルチマスター・レプリケーションはトランザクションごとに処理を行うため、マテリアライズド・ビューを使用したレプリケーションよりもネットワーク要件が多くなります。また、マルチマスター・レプリケーションではリアルタイムまたはほぼリアルタイムにレプリケーションが提供されるため、ネットワークの通信量が増大する結果となり、専用のネットワーク・リンクが必要となることもあります。

マテリアライズド・ビューは、1 つのマスター・サイトまたはマスター・マテリアライズド・ビュー・サイトから効率的なバッチ処理を使用して更新されます。マテリアライズド・ビュー・レプリケーションでは一定時間おきにデータが更新されるので、マルチマスター・レプリケーションの場合よりもネットワーク要件が少なく、依存性も高くありません。マルチマスター・レプリケーションではネットワーク上で常に通信が行われますが、マテリアライズド・ビュー・レプリケーションでは定期的なリフレッシュのみが必要になります。

マテリアライズド・ビューを使用したデータのレプリケーションでは、専用のネットワーク接続が不要なうえ、ターゲット・データへのローカル・アクセスを実行できるためデータの

可用性が高くなります。これらのメリットに加え、大量配布およびデータのサブセット化（いずれもネットワーク負荷を軽減します）というメリットにより、レプリケート・データベースのパフォーマンスと信頼性が大幅に向上します。

大量配布環境の作成

配置テンプレートを使用すると、マテリアライズド・ビュー環境をローカルで事前作成できます。また、配置テンプレートを使用すれば、マテリアライズド・ビュー環境を迅速かつ簡単に配置でき、営業支援システムやその他の大量配布環境をサポートできます。パラメータを使用すると、配置テンプレートを変更せずに個々のユーザー用のカスタム・データ・セットを作成できます。このテクノロジによって、データベースのインフラストラクチャを数百人または数千人のユーザーに提供できます。

データのサブセット化の有効化

マルチマスター・レプリケーションでは、表全体をレプリケートする必要がありますが、マテリアライズド・ビューでは、列レベルまたは行レベル（あるいはその両方）でのサブセット化に基づいたデータをレプリケートできます。データのサブセット化により、特定のサイトのみに関係する情報をレプリケートできます。たとえば、各地の営業所ではその地域に必要なデータのみをレプリケートし、不要なネットワークの通信量を減らすことができます。

モバイル・コンピューティングの有効化

マテリアライズド・ビューは専用のネットワーク接続を必要としません。マテリアライズド・ビューでは、ジョブをスケジュールしてリフレッシュ処理を自動化するというオプションもありますが、必要に応じて手動でリフレッシュできます。これは、ラップトップ上で実行される販売アプリケーションなどには理想的です。たとえば、開発者は、必要なときにリフレッシュを実行するレプリケーション・マネージメント API を販売アプリケーションに統合できます。1 日の受注業務を終えた営業担当員は、ネットワークにダイアルアップ接続し、統合されたメカニズムを使用してデータベースをリフレッシュすることによって、本社に受注を転送できます。

読取り専用、更新可能および書込み可能のマテリアライズド・ビュー

マテリアライズド・ビューは読取り専用、更新可能または書込み可能のいずれかです。読取り専用マテリアライズド・ビューに対してはデータ操作言語（DML）文を実行できませんが、更新可能および書込み可能のマテリアライズド・ビューに対しては実行できます。

注意： 読取り専用、更新可能および書込み可能のマテリアライズド・ビューでは、マテリアライズド・ビューの定義問合せでマスター内の主キー列をすべて参照する必要があります。

関連項目：

- 読取り専用、更新可能および書込み可能のマテリアライズド・ビューの概要は、1-8 ページの「マテリアライズド・ビュー・レプリケーション」を参照してください。
- マテリアライズド・ビューでのデータ型に関する考慮事項の詳細は、6-2 ページの「データ型に関する考慮事項」を参照してください。

読取り専用マテリアライズド・ビュー

マテリアライズド・ビューの作成中に、FOR UPDATE 句を指定しないか、レプリケーション・マネージメント・ツールでこれと同等のオプションを使用禁止にすると、マテリアライズド・ビューを読取り専用にできます。読取り専用マテリアライズド・ビューでは更新可能マテリアライズド・ビューと同じメカニズムを多数使用しますが、読取り専用の場合は、マテリアライズド・ビュー・グループに属する必要がありません。

また、読取り専用マテリアライズド・ビューを使用すると、マテリアライズド・ビューが原因でマスター・サイトまたはマスター・マテリアライズド・ビュー・サイトでデータ競合が発生することを回避できますが、リモート・マテリアライズド・ビュー・サイトでの更新はできなくなります。読取り専用マテリアライズド・ビューの例を次に示します。

```
CREATE MATERIALIZED VIEW hr.employees AS
  SELECT * FROM hr.employees@orc1.world;
```

更新可能マテリアライズド・ビュー

マテリアライズド・ビューの作成中に、FOR UPDATE 句を指定するか、レプリケーション・マネージメント・ツールでこれと同等のオプションを使用可能にすると、マテリアライズド・ビューを更新可能にできます。更新可能マテリアライズド・ビューへの変更をリフレッシュ中にマスターにプッシュするには、更新可能マテリアライズド・ビューがマテリアライズド・ビュー・グループに属する必要があります。

更新可能マテリアライズド・ビューを使用すると、ユーザーがマテリアライズド・ビュー・サイトでデータを変更できるため、マスター・サイトの負荷が軽減されます。更新可能マテリアライズド・ビューの例を次に示します。

```
CREATE MATERIALIZED VIEW hr.departments FOR UPDATE AS
  SELECT * FROM hr.departments@orc1.world;
```

次の文ではマテリアライズド・ビュー・グループが作成されます。

```
BEGIN
   DBMS_REPCAT.CREATE_MVIEW_REPGROUP (
      gname => 'hr_repg',
      master => 'orc1.world',
      propagation_mode => 'ASYNCHRONOUS');
END;
/
```

次の文では、hr.departments マテリアライズド・ビューがマテリアライズド・ビュー・グループに追加され、このマテリアライズド・ビューが更新可能になります。

```
BEGIN
   DBMS_REPCAT.CREATE_MVIEW_REPOBJECT (
      gname => 'hr_repg',
      sname => 'hr',
      oname => 'departments',
      type => 'SNAPSHOT',
      min_communication => TRUE);
END;
/
```

マテリアライズド・ビュー・グループを作成し、これにマテリアライズド・ビューを追加するには、レプリケーション・マネージメント・ツールを使用することもできます。

注意：

- 更新可能マテリアライズド・ビューを作成するときは、列別名を使用しないでください。CREATE_MVIEW_REPOBJECT プロシージャを使用してマテリアライズド・ビューをマテリアライズド・ビュー・グループに追加しようとしたときに、列の別名が原因でエラーが発生します。
- マスター表またはマスター・マテリアライズド・ビューの列にデフォルト値が定義されていても、これらに基づいた更新可能マテリアライズド・ビューがそのデフォルト値を自動的に使用するわけではありません。
- 更新可能マテリアライズド・ビューは、DELETE CASCADE 制約をサポートしません。

関連項目：

- 詳細は、3-53 ページの「マテリアライズド・ビュー・グループ」を参照してください。
- 列別名の詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

書込み可能マテリアライズド・ビュー

書込み可能マテリアライズド・ビューとは、FOR UPDATE 句を使用して作成されるが、マテリアライズド・ビュー・グループには属さないマテリアライズド・ビューです。ユーザーは書込み可能マテリアライズド・ビューに対して DML 操作を実行できますが、マテリアライズド・ビューをリフレッシュしたときに、この変更はマスターにプッシュされず、マテリアライズド・ビュー自体からも変更が失われます。通常、高速リフレッシュが可能な読取り専用マテリアライズド・ビューが使用できる場合は、書込み可能マテリアライズド・ビューも使用できます。

注意： 書込み可能マテリアライズド・ビューはほとんど使用されないため、マテリアライズド・ビューに関するドキュメントのほとんどは、読取り専用および更新可能マテリアライズド・ビューのみを説明しています。

使用可能なマテリアライズド・ビュー

Oracle では、様々なレプリケーション環境（および非レプリケーション環境）で使用できるいくつかのタイプのマテリアライズド・ビューを用意しています。次の項では、それぞれのタイプのマテリアライズド・ビューと、各マテリアライズド・ビューに最適な環境について説明します。

次の項では、様々なタイプのマテリアライズド・ビューの作成例を示しています。

- 主キー・マテリアライズド・ビュー
- オブジェクト・マテリアライズド・ビュー
- ROWID マテリアライズド・ビュー
- 複合マテリアライズド・ビュー

どのタイプのマテリアライズド・ビューを作成する場合でも、マテリアライズド・ビューの問合せで表所有者のスキーマ名を指定します。たとえば、次の CREATE MATERIALIZED VIEW 文について考えます。

```
CREATE MATERIALIZED VIEW hr.employees
  AS SELECT * FROM hr.employees@orc1.world;
```

この文では、問合せにスキーマ hr が指定されています。

主キー・マテリアライズド・ビュー

主キー・マテリアライズド・ビューがマテリアライズド・ビューのデフォルト・タイプです。マテリアライズド・ビューがマテリアライズド・ビュー・グループの一部として作成され、マテリアライズド・ビューの定義で FOR UPDATE を指定した場合、マテリアライズド・ビューは更新可能です。更新可能マテリアライズド・ビューは、マスター・サイトまたはマスター・マテリアライズド・ビュー・サイトでレプリケーション・グループと同じ名前のマテリアライズド・ビュー・グループに属する必要があります。さらに、更新可能マテリアラ

イズド・ビューは、マスター・レプリケーション・グループと別のデータベースに配置する必要があります。

変更は、（ROWID ではなく）行の主キー値によって識別される、行レベルの変更に従って伝播されます。更新可能な主キー・マテリアライズド・ビューを作成する SQL 文の例を次に示します。

```
CREATE MATERIALIZED VIEW oe.customers FOR UPDATE AS
  SELECT * FROM oe.customers@orc1.world;
```

リモート・マテリアライズド・ビュー・サイトで行のサブセットを作成できるように、主キー・マテリアライズド・ビューには副問合せを含められます。副問合せは主問合せに埋め込まれた問合せで、CREATE MATERIALIZED VIEW 文に複数の SELECT 文が使用されます。この副問合せには、基本的な WHERE 句のように単純なものと、マルチレベルの WHERE EXISTS 句のように複雑なものとがあります。参照される各マスターにマテリアライズド・ビュー・ログがある場合は、選択されたクラスの副問合せを含む主キー・マテリアライズド・ビュー・サイトに高速リフレッシュを実行できます。高速リフレッシュでは、マテリアライズド・ビュー・ログを使用して、最後のリフレッシュ以降に変更が加えられた行のみを更新します。

次のマテリアライズド・ビューは、副問合せを含む WHERE 句を使用して作成されています。

```
CREATE MATERIALIZED VIEW oe.orders REFRESH FAST AS
 SELECT * FROM oe.orders@orc1.world o
 WHERE EXISTS
   (SELECT * FROM oe.customers@orc1.world c
    WHERE o.customer_id = c.customer_id AND c.credit_limit > 10000);
```

このタイプのマテリアライズド・ビューは、副問合せマテリアライズド・ビューと呼ばれます。

注意： この oe.orders マテリアライズド・ビューを作成するには、マスターのマテリアライズド・ビュー・ログに credit_limit をロギングする必要があります。詳細は、6-16 ページの「マテリアライズド・ビュー・ログへの列のロギング」を参照してください。

関連項目：

- マテリアライズド・ビュー・グループの詳細は、3-53 ページの「マテリアライズド・ビュー・グループ」を参照してください。
- 副問合せを含むマテリアライズド・ビューの詳細は、3-19 ページの「副問合せを使用するマテリアライズド・ビュー」を参照してください。
- 高速リフレッシュの詳細は、3-59 ページの「リフレッシュ・タイプ」を参照してください。
- マテリアライズド・ビュー・ログの詳細は、3-47 ページの「マテリアライズド・ビュー・ログ」を参照してください。
- 副問合せの詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

オブジェクト・マテリアライズド・ビュー

マテリアライズド・ビューがオブジェクト表に基づいて `OF` *type* 句を使用して作成されている場合、このマテリアライズド・ビューは**オブジェクト・マテリアライズド・ビュー**と呼ばれます。オブジェクト・マテリアライズド・ビューは、オブジェクト表と同じ構造です。つまり、オブジェクト・マテリアライズド・ビューは行オブジェクトから構成され、各行オブジェクトはオブジェクト識別子（OID）列により識別されます。

関連項目： 3-36 ページ「オブジェクト表に基づいたマテリアライズド・ビュー」

ROWID マテリアライズド・ビュー

Oracle では、下位互換性のために、デフォルトの主キー・マテリアライズド・ビューの他に、`ROWID` マテリアライズド・ビューもサポートしています。`ROWID` マテリアライズド・ビューは、マスター表の行の物理的な行識別子（ROWID）に基づいて作成されます。`ROWID` マテリアライズド・ビューを使用するのは、Oracle7 データベースのマスター表に基づいたマテリアライズド・ビューの場合のみです。Oracle8 以上のデータベースのマスターに基づいて新しいマテリアライズド・ビューを作成するときは使用しないでください。

`ROWID` マテリアライズド・ビューを作成する `CREATE MATERIALIZED VIEW` 文の例を次に示します。

```
CREATE MATERIALIZED VIEW oe.orders REFRESH WITH ROWID AS
 SELECT * FROM oe.orders@orc1.world;
```

関連項目： `ROWID` マテリアライズド・ビューと主キー・マテリアライズド・ビューの相違点の詳細は、3-47 ページの「マテリアライズド・ビュー・ログ」を参照してください。

複合マテリアライズド・ビュー

高速リフレッシュを実行するには、マテリアライズド・ビューの定義問合せで一定の制限事項を守る必要があります。マテリアライズド・ビューの定義問合せが汎用的なものであり、制限事項を守ることができない場合は、そのマテリアライズド・ビューは複合マテリアライズド・ビューであり、高速リフレッシュを実行できません。

具体的には、マテリアライズド・ビューの定義問合せに次のものが含まれる場合、そのマテリアライズド・ビューは複合マテリアライズド・ビューとみなされます。

- CONNECT BY 句

  たとえば、次の文では複合マテリアライズド・ビューが作成されます。

  ```
  CREATE MATERIALIZED VIEW hr.emp_hierarchy AS
    SELECT LPAD(' ', 4*(LEVEL-1))||email USERNAME
      FROM hr.employees@orc1.world START WITH manager_id IS NULL
      CONNECT BY PRIOR employee_id = manager_id;
  ```

- INTERSECT、MINUS または UNION ALL 集合演算

  たとえば、次の文には UNION ALL 集合演算が含まれているため、複合マテリアライズド・ビューが作成されます。

  ```
  CREATE MATERIALIZED VIEW hr.mview_employees AS
    SELECT employees.employee_id, employees.email
    FROM hr.employees@orc1.world
  UNION ALL
    SELECT new_employees.employee_id, new_employees.email
    FROM hr.new_employees@orc1.world;
  ```

- DISTINCT または UNIQUE キーワード（場合による）。ただし、DISTINCT または UNIQUE キーワードを定義問合せに指定して、単純マテリアライズド・ビューを作成することもできます。

  たとえば、次の文では複合マテリアライズド・ビューが作成されます。

  ```
  CREATE MATERIALIZED VIEW hr.employee_depts AS
    SELECT DISTINCT department_id FROM hr.employees@orc1.world
    ORDER BY department_id;
  ```

- 集計関数

  たとえば、次の文では複合マテリアライズド・ビューが作成されます。

  ```
  CREATE MATERIALIZED VIEW hr.average_sal AS
    SELECT AVG(salary) "Average" FROM hr.employees@orc1.world;
  ```

- 副問合せ以外で指定される結合

  たとえば、次の文では複合マテリアライズド・ビューが作成されます。

```
CREATE MATERIALIZED VIEW hr.emp_join_dep AS
  SELECT last_name
  FROM hr.employees@orc1.world e, hr.departments@orc1.world d
  WHERE e.department_id = d.department_id;
```

- UNION 演算（場合による）。具体的には、次の条件のいずれかが満たされた場合に、UNION 演算を指定したマテリアライズド・ビューは複合マテリアライズド・ビューになります。

  – UNION 内の問合せが複合問合せの場合。ここまでの箇条書きの項目は、問合せによりマテリアライズド・ビューが複合マテリアライズド・ビューになる場合を示します。

  – 1 番外側の SELECT リスト列が、UNION 内の問合せに一致しない場合。次の例で、最初の問合せの 1 番外側の SELECT リストには order_total のみが含まれ、2 番目の問合せの 1 番外側の SELECT リストには customer_id が含まれています。したがって、マテリアライズド・ビューは複合マテリアライズド・ビューです。

```
CREATE MATERIALIZED VIEW oe.orders AS
  SELECT order_total
  FROM oe.orders@orc1.world o
  WHERE EXISTS
    (SELECT cust_first_name, cust_last_name
       FROM oe.customers@orc1.world c
       WHERE o.customer_id = c.customer_id
       AND c.credit_limit > 50)
UNION
  SELECT customer_id
  FROM oe.orders@orc1.world o
  WHERE EXISTS
    (SELECT cust_first_name, cust_last_name
       FROM oe.customers@orc1.world c
       WHERE o.customer_id = c.customer_id
       AND c.account_mgr_id = 30);
```

  1 番内側の SELECT リストは、マテリアライズド・ビューが複合マテリアライズド・ビューであるかどうかには関係しません。この例では、UNION 内の両方の問合せで、1 番内側の SELECT リストは cust_first_name と cust_last_name です。

- 3-24 ページの「副問合せを使用するマテリアライズド・ビューでの制限事項」で説明されている要件に適合しない句。

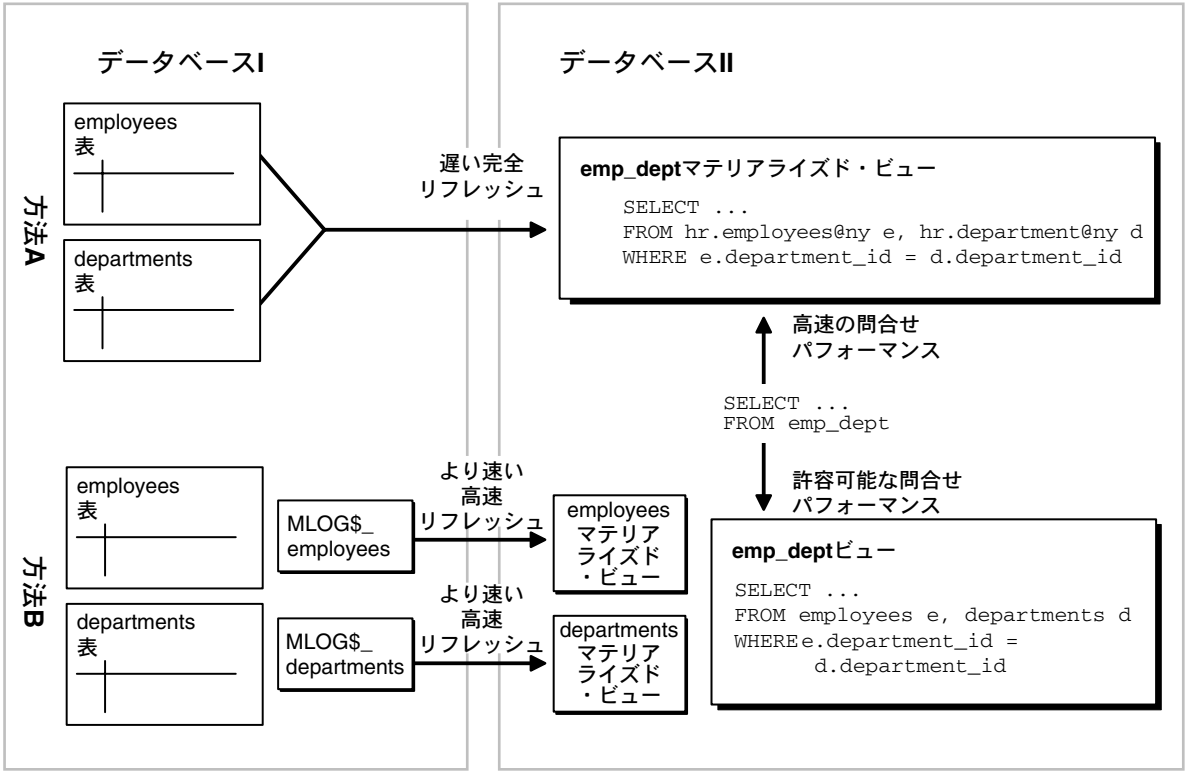
関連項目：

- 集計関数を含むマテリアライズド・ビューおよび複合マテリアライズド・ビューの詳細は、『Oracle9*i* データ・ウェアハウス・ガイド』を参照してください。
- CONNECT BY 句、集合演算、DISTINCT キーワードおよび集計関数の詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

注意： 複合マテリアライズド・ビューでは高速リフレッシュができず、ネットワークのパフォーマンスが低下する場合があるため、できるだけ使用しないでください（詳細は、3-58 ページの「リフレッシュ処理」を参照してください）。

単純マテリアライズド・ビューと複合マテリアライズド・ビューの比較 アプリケーションによっては、複合マテリアライズド・ビューの使用を考慮することが必要な場合があります。図 3-2 とその後に続く本文では、考慮の必要な場合について説明します。

図 3-2 単純マテリアライズド・ビューと複合マテリアライズド・ビューの比較



- **複合マテリアライズド・ビュー**: 図 3-2 の方法 A は、複合マテリアライズド・ビューを示しています。データベース II 内のマテリアライズド・ビューでは、マテリアライズド・ビューのリフレッシュ中に結合操作が完了しているので、非常に効率的に問合せが行われます。ただし、複合マテリアライズド・ビューであるため完全リフレッシュを実行する必要があります。多くの場合、完全リフレッシュは、高速リフレッシュに比べ実行速度が遅くなります。
- **結合ビューを含む単純マテリアライズド・ビュー**: 図 3-2 の方法 B は、データベース II 内の 2 つの単純マテリアライズド・ビューと、マテリアライズド・ビューのデータベース内で結合を実行するビューを示しています。ビューに対する問合せのパフォーマンスは、方法 A の複合マテリアライズド・ビューに対する問合せのパフォーマンスに比べよくありません。ただし、単純マテリアライズド・ビューの方が、高速リフレッシュとマテリアライズド・ビュー・ログを使用してより高速にリフレッシュできます。

要約すると、使用する方法は次のようにして判断できます。

- リフレッシュを実行する頻度が少なく、より高速な問合せのパフォーマンスを優先する場合は、方法 A（複合マテリアライズド・ビュー）を使用します。
- 頻繁にリフレッシュを実行し、問合せのパフォーマンスを優先する必要がない場合は、方法 B（単純マテリアライズド・ビュー）を使用します。

マテリアライズド・ビューの操作に必要な権限

マテリアライズド・ビューに対しては、次の 3 種類のユーザーが操作を実行します。

- **作成者 :** マテリアライズド・ビューを作成するユーザー。
- **リフレッシャ :** マテリアライズド・ビューをリフレッシュするユーザー。
- **所有者 :** マテリアライズド・ビューを所有するユーザー。マテリアライズド・ビューは、このユーザーのスキーマ内に入れられます。

1 人のユーザーが特定のマテリアライズド・ビューに対してこのすべての操作を実行できます。ただし、レプリケーション環境によっては、特定のマテリアライズド・ビューに対して別々のユーザーがこの操作を実行します。これらの操作の実行に必要な権限は、1 人のユーザーが操作を実行するか別々のユーザーが実行するかにより異なります。以降の項で、権限要件を詳しく説明します。

注意 : 以降の項では、クエリー・リライトが使用可能なマテリアライズド・ビューの作成に必要な要件については説明されていません。詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

関連項目 : 以降の項ではデータベース・リンクについて説明しています。データベース・リンクの使用方法の詳細は、『Oracle9*i* データベース管理者ガイド』を参照してください。

作成者が所有者である場合

マテリアライズド・ビューの作成者が所有者でもある場合、このユーザーには、マテリアライズド・ビューを作成する次の権限が必要です。これらの権限は明示的に付与されるか、ロールを介して付与されます。

- `CREATE MATERIALIZED VIEW` または `CREATE ANY MATERIALIZED VIEW`
- `CREATE TABLE` または `CREATE ANY TABLE`
- `CREATE VIEW` または `CREATE ANY VIEW`（データベースの互換性のレベルがリリース 8.1 より低い場合）

- マスターおよびマスターのマテリアライズド・ビュー・ログに対する SELECT オブジェクト権限または SELECT ANY TABLE システム権限。マスター・サイトまたはマスター・マテリアライズド・ビュー・サイトがリモートの場合は、マテリアライズド・ビュー・サイトのユーザーがデータベース・リンクを介して接続する先のマスター・サイトまたはマスター・マテリアライズド・ビュー・サイトのユーザーに、SELECT オブジェクト権限を付与する必要があります。

  マテリアライズド・ビュー・サイトのマテリアライズド・ビューの所有者に、マスター・サイトまたはマスター・マテリアライズド・ビュー・サイトへのプライベート・データベース・リンクがある場合は、このデータベース・リンクがマスター・サイトまたはマスター・マテリアライズド・ビュー・サイトのマスターの所有者に接続します。それ以外の場合は、データベース・リンク経由の接続に関する通常の規則が適用されます。

作成者が所有者でない場合

マテリアライズド・ビューの作成者が所有者でない場合は、マテリアライズド・ビューを作成する権限を作成者と所有者に付与する必要があります。作成者の権限は、明示的にまたはロールを介して付与できますが、所有者の権限は明示的に付与する必要があります。つまり、所有者に付与する権限は、ロールを介して付与できません。

表 3-1 は、マテリアライズド・ビューの作成者が所有者でない場合に必要な権限を示します。

表 3-1 マテリアライズド・ビューの作成に必要な権限（作成者が所有者でない場合）

作成者	所有者
CREATE ANY MATERIALIZED VIEW	CREATE TABLE または CREATE ANY TABLE CREATE VIEW または CREATE ANY VIEW (データベースの互換性のレベルがリリース 8.1 より低い場合) マスターおよびマスターのマテリアライズド・ビュー・ログに対する SELECT オブジェクト権限または SELECT ANY TABLE システム権限。マスター・サイトまたはマスター・マテリアライズド・ビュー・サイトがリモートの場合は、マテリアライズド・ビュー・サイトのユーザーがデータベース・リンクを介して接続する先のマスター・サイトまたはマスター・マテリアライズド・ビュー・サイトのユーザーに、SELECT オブジェクト権限を付与する必要があります。 マテリアライズド・ビュー・サイトのマテリアライズド・ビューの所有者に、マスター・サイトまたはマスター・マテリアライズド・ビュー・サイトへのプライベート・データベース・リンクがある場合は、このデータベース・リンクがマスター・サイトまたはマスター・マテリアライズド・ビュー・サイトのマスターの所有者に接続します。それ以外の場合は、データベース・リンク経由の接続に関する通常の規則が適用されます。 **注意：**所有者のこれらの権限は、ロールを介してではなく、明示的にユーザーに付与する必要があります。

リフレッシャが所有者である場合

マテリアライズド・ビューのリフレッシャが所有者でもある場合、このユーザーには、マスターおよびマスターのマテリアライズド・ビュー・ログに対する SELECT オブジェクト権限または SELECT ANY TABLE システム権限が必要です。マスター・サイトまたはマスター・マテリアライズド・ビュー・サイトがリモートの場合は、マテリアライズド・ビュー・サイトのユーザーがデータベース・リンクを介して接続する先のマスター・サイトまたはマスター・マテリアライズド・ビュー・サイトのユーザーに、SELECT オブジェクト権限を付与する必要があります。この権限は、明示的にまたはロールを介して付与できます。

マテリアライズド・ビュー・サイトのマテリアライズド・ビューの所有者に、マスター・サイトまたはマスター・マテリアライズド・ビュー・サイトへのプライベート・データベース・リンクがある場合は、このデータベース・リンクがマスター・サイトまたはマスター・

マテリアライズド・ビュー・サイトのマスターの所有者に接続します。それ以外の場合は、データベース・リンク経由の接続に関する通常の規則が適用されます。

リフレッシャが所有者でない場合

マテリアライズド・ビューのリフレッシャが所有者でない場合は、リフレッシャと所有者に権限を付与する必要があります。これらの権限は、明示的にまたはロールを介して付与できます。

表 3-2 は、マテリアライズド・ビューのリフレッシャが所有者でない場合に必要な権限を示します。

表 3-2 マテリアライズド・ビューのリフレッシュに必要な権限（リフレッシャが所有者でない場合）

リフレッシャ	所有者
ALTER ANY MATERIALIZED VIEW	マスター・サイトまたはマスター・マテリアライズド・ビュー・サイトがローカルの場合は、マスターおよびマスターのマテリアライズド・ビュー・ログに対する SELECT オブジェクト権限または SELECT ANY TABLE システム権限。 マスター・サイトまたはマテリアライズド・ビュー・サイトがリモートの場合は、マテリアライズド・ビュー・サイトのユーザーがデータベース・リンクを介して接続する先のマスター・サイトまたはマスター・マテリアライズド・ビュー・サイトのユーザーに、SELECT オブジェクト権限を付与する必要があります。マテリアライズド・ビュー・サイトのマテリアライズド・ビューの所有者に、マスター・サイトまたはマスター・マテリアライズド・ビュー・サイトへのプライベート・データベース・リンクがある場合は、このデータベース・リンクがマスター・サイトまたはマスター・マテリアライズド・ビュー・サイトのマスターの所有者に接続します。それ以外の場合は、データベース・リンク経由の接続に関する通常の規則が適用されます。

マテリアライズド・ビューを使用したデータのサブセット化

マスター表またはマスター・マテリアライズド・ビューのデータのサブセットをマテリアライズド・ビューに反映させる場合があります。行のサブセット化を使用すると、マスターからマテリアライズド・ビューで必要な行のみを含めることができます。これには WHERE 句を使用します。列のサブセット化を使用すると、マスターからマテリアライズド・ビューで必要な列のみを含めることができます。これを行うには、マテリアライズド・ビューの作成中に、SELECT 文に該当する列を指定します。配置テンプレートを使用してマテリアライズド・ビューを作成する場合は、更新可能マテリアライズド・ビューに対して列のサブセットを定義できます。

データのサブセット化の使用目的には、次のようなものがあります。

- **ネットワークの通信量の軽減** : 列のサブセットを含むマテリアライズド・ビューでは、マテリアライズド・ビューの定義問合せの WHERE 句を満たす変更のみがマテリアライズド・ビュー・サイトに伝播されます。このため、転送データ量が減少し、ネットワーク通信量が軽減されます。
- **機密データの保護** : マテリアライズド・ビューの定義問合せを満たすデータ以外は表示できません。
- **リソース要件の削減** : マテリアライズド・ビューがラップトップに格納される場合は、一般にハード・ディスクは会社のサーバーよりもはるかに小さくなります。サブセット化したマテリアライズド・ビューにより、使用する記憶領域をかなり小さくすることができます。
- **リフレッシュ時間の短縮** : マテリアライズド・ビュー・サイトに伝播されるデータが少なくなるため、リフレッシュ処理が高速になります。これは、ラップトップからダイアルアップ接続を使用してマテリアライズド・ビューをリフレッシュする必要のあるユーザーにとって不可欠です。

たとえば、次の文では、oe.orders@orc1.world マスター表に基づいたマテリアライズド・ビューが作成され、sales_rep_id 番号が 173 の営業担当員の行のみが含まれます。

```
CREATE MATERIALIZED VIEW oe.orders REFRESH FAST AS
 SELECT * FROM oe.orders@orc1.world
 WHERE sales_rep_id = 173;
```

受注表で sales_rep_id 番号が 173 以外の行は、このマテリアライズド・ビューからは除外されます。

注意 : 以降の項では、副問合せを使用した行のサブセット化について説明します。列のサブセット化の詳細は、4-21 ページの「配置テンプレートを使用した列のサブセット化」を参照してください。

副問合せを使用するマテリアライズド・ビュー

前述の例は、他のマテリアライズド・ビューへの参照制約がない個別のマテリアライズド・ビューでは非常にうまく機能します。しかし、複数の表の情報に基づいたデータをレプリケートする場合は、これらのマテリアライズド・ビューのメンテナンスと定義が難しくなります。次の項では、副問合せが役立つ場合の例を示します。

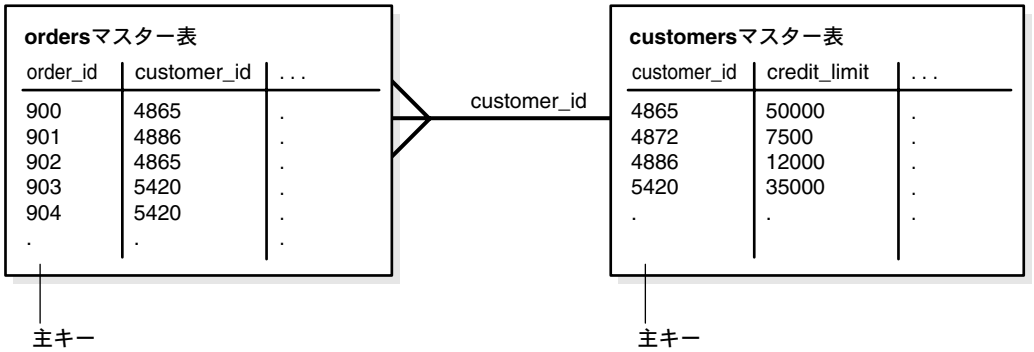
多対 1 副問合せ oe スキーマに customers 表と orders 表が含まれている場合に、orders 表と customers 表の両方のデータに基づいた orders 表のマテリアライズド・ビューを作成するとします。たとえば、営業担当員が、信用限度額が 10,000 ドルを超える顧客の注文をすべて調べる必要があるとすると、orders マテリアライズド・ビューを作成する CREATE MATERIALIZED VIEW 文には、多対 1 関係の副問合せが含まれます。これは、それぞれの顧客が多数の注文をしている可能性があるためです。

図 3-3 では、customers 表と orders 表は customer_id 列を介して関係していることが示されています。営業担当員の目的は、次の文で達成されます。つまり、次の文では、信用限度額が 10,000 ドルより大きい顧客の注文が含まれたマテリアライズド・ビューが作成されます。

```
CREATE MATERIALIZED VIEW oe.orders REFRESH FAST FOR UPDATE AS
  SELECT * FROM oe.orders@orc1.world o
  WHERE EXISTS
    (SELECT * FROM oe.customers@orc1.world c
     WHERE o.customer_id = c.customer_id AND c.credit_limit > 10000);
```

注意： この oe.orders マテリアライズド・ビューを作成するには、マスターのマテリアライズド・ビュー・ログに credit_limit をロギングする必要があります。詳細は、6-16 ページの「マテリアライズド・ビュー・ログへの列のロギング」を参照してください。

図 3-3 多対 1 副問合せを使用した行のサブセット化



この文により作成されるマテリアライズド・ビューは、高速リフレッシュが可能で更新も可能です。信用限度額が 10,000 ドルを超える新規顧客が発生した場合は、後続のリフレッシュ処理中に新規データがこのマテリアライズド・ビュー・サイトに伝播されます。同様に、顧客の信用限度額が 10,000 ドル以下になった場合は、後続のリフレッシュ処理中に、その顧客のデータがこのマテリアライズド・ビューから削除されます。

1 対多副問合せ oe スキーマに customers 表と orders 表が含まれている場合に、customers 表と orders 表の両方のデータに基づいた customers 表のマテリアライズド・ビューを作成するとします。たとえば、営業担当員が、受注合計額が 20,000 ドル以上の全顧客を調べる必要があるとすると、最も効率的な方法は、定義問合せに 1 対多副問合せを指定したマテリアライズド・ビューを作成することです。

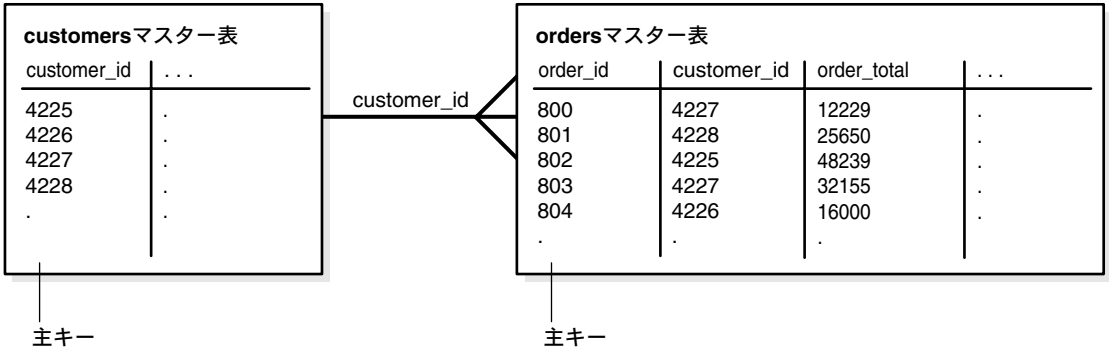
customers 表に対する CREATE MATERIALIZED VIEW 文の定義問合せには、1 対多関係を持つ副問合せが含まれます。つまり、それぞれの顧客が多数の注文をしている可能性があります。

図 3-4 では、orders 表と customers 表は customer_id 列を介して関係していることが示されています。営業担当員の目的は、次の文で達成されます。つまり、次の文では、受注合計額が 20,000 ドルより大きい全顧客が含まれたマテリアライズド・ビューが作成されます。

```
CREATE MATERIALIZED VIEW oe.customers REFRESH FAST FOR UPDATE AS
  SELECT * FROM oe.customers@orc1.world c
  WHERE EXISTS
    (SELECT * FROM oe.orders@orc1.world o
     WHERE c.customer_id = o.customer_id AND o.order_total > 20000);
```

注意： この oe.customers マテリアライズド・ビューを作成するには、orders 表のマテリアライズド・ビュー・ログに customer_id と order_total をロギングする必要があります。詳細は、6-16 ページの「マテリアライズド・ビュー・ログへの列のロギング」を参照してください。

図 3-4 1 対多副問合せを使用した行のサブセット化



この文により作成されるマテリアライズド・ビューは、高速リフレッシュが可能で、更新も可能です。受注合計額が 20,000 ドルを超える新規顧客が発生した場合は、後続のリフレッシュ処理中に新規データがこのマテリアライズド・ビュー・サイトに伝播されます。同様に、20,000 ドルを超える受注合計額に含まれている注文を顧客が取り消して、受注合計額が 20,000 ドルより低くなった場合は、後続のリフレッシュ処理中に、その顧客のデータがこのマテリアライズド・ビューから削除されます。

注意： リリース 1（9.0.1）より前の Oracle では、1 対多の副問合せを使用するマテリアライズド・ビューの高速リフレッシュがサポートされていないため、このマテリアライズド・ビュー・サイトの互換性のレベルはリリース 9.0.1 以上である必要があります。互換性のレベルは、COMPATIBLE 初期化パラメータによって制御されます。

多対多副問合せ oe スキーマに order_items 表と inventories 表が含まれている場合に、inventories 表と order_items 表の両方のデータに基づいた inventories 表のマテリアライズド・ビューを作成するとします。たとえば、営業担当員が、現在のインベントリ数が 1 以上の各製品で、製品の product_id が order_items 表に含まれている全製品のインベントリを調べる必要があるとします。つまり、営業担当員は、顧客が注文した全製品について、インベントリが 1 以上の製品を調べる必要があります。ここで、インベントリとは、特定の倉庫にある製品の数です。このため、特定の製品は多数の受注項目と多数のインベントリに含まれる可能性があります。

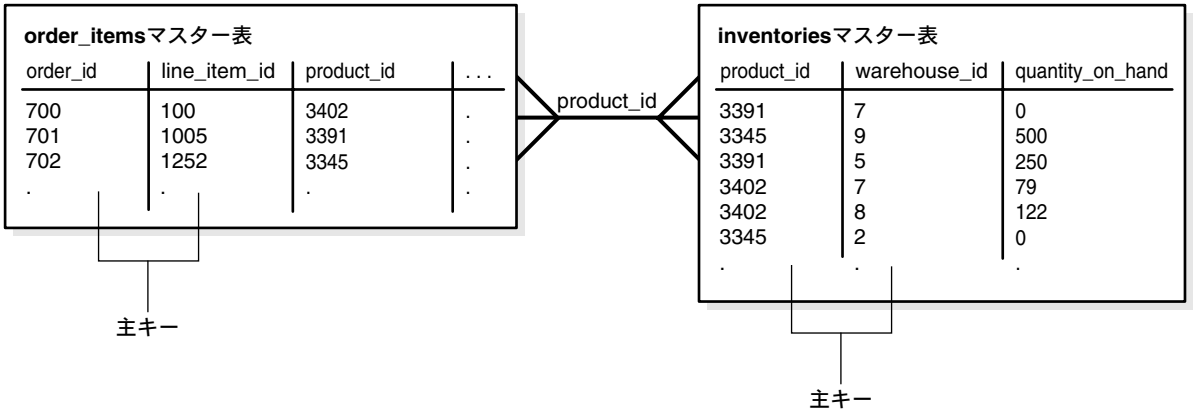
この営業担当員の目的を達成するには、order_items 表と inventories 表との間の多対多関係に対する副問合せを使用したマテリアライズド・ビューを作成します。

inventories マテリアライズド・ビューを作成する場合は、order_items 表に含まれている製品に関して、現在の数量が 1 以上のインベントリを取り出します。図 3-5 では、inventories 表と order_items 表は product_id 列を介して関係していることが示されています。次の文によりこのマテリアライズド・ビューが作成されます。

```
CREATE MATERIALIZED VIEW oe.inventories REFRESH FAST FOR UPDATE AS
  SELECT * FROM oe.inventories@orc1.world i
  WHERE i.quantity_on_hand > 0 AND EXISTS
    (SELECT * FROM oe.order_items@orc1.world o
     WHERE i.product_id = o.product_id);
```

注意： この oe.inventories マテリアライズド・ビューを作成するには、マスターのマテリアライズド・ビュー・ログに order_items 表の product_id 列をロギングする必要があります。詳細は、6-16 ページの「マテリアライズド・ビュー・ログへの列のロギング」を参照してください。

図 3-5 多対多副問合せを使用した行のサブセット化



この文により作成されるマテリアライズド・ビューは、高速リフレッシュが可能で、更新も可能です。製品に対して 1 以上のインベントリが `order_items` 表で見つかった場合は、後続のリフレッシュ処理中に新規データがこのマテリアライズド・ビュー・サイトに伝播されます。同様に、顧客が製品の注文を取り消して、その製品に対する他の注文が `order_items` 表にない場合は、後続のリフレッシュ処理中に、その製品のインベントリがこのマテリアライズド・ビューから削除されます。

注意： リリース 1（9.0.1）より前の Oracle では、多対多の副問合せを使用するマテリアライズド・ビューの高速リフレッシュがサポートされていないため、このマテリアライズド・ビュー・サイトの互換性のレベルはリリース 9.0.1 以上である必要があります。互換性のレベルは、`COMPATIBLE` 初期化パラメータによって制御されます。

副問合せを使用するマテリアライズド・ビューでの制限事項

副問合せを使用するマテリアライズド・ビューの定義問合せには、マテリアライズド・ビューの高速リフレッシュ機能を保つためのいくつかの制限事項があります。

副問合せを使用する高速リフレッシュ・マテリアライズド・ビューの制限事項は、次のとおりです。

- マテリアライズド・ビューは、主キー・マテリアライズド・ビューである必要があります。
- マスターのマテリアライズド・ビュー・ログに、副問合せで参照されている列が含まれている必要があります。含める列の詳細は、6-16 ページの「マテリアライズド・ビュー・ログへの列のロギング」を参照してください。
- 副問合せが多対多または 1 対多の場合は、主キーの一部ではない結合列をマスターのマテリアライズド・ビュー・ログに含める必要があります。この制限は、多対 1 の副問合せには適用されません。
- 副問合せは、肯定副問合せであることが必要です。たとえば、EXISTS は使用できますが、NOT EXISTS は使用できません。
- 副問合せは、EXISTS を使用して各ネスト・レベルを接続する必要があります（IN は使用できません）。
- 各表は 1 つの EXISTS 式にのみ含められます。
- 結合式では、完全一致または等価比較（等価結合）を使用する必要があります。
- 各表は副問合せ内で 1 回のみ結合できます。
- 各ネスト・レベルの各表に、主キーが存在している必要があります。
- 各ネスト・レベルでは、上位のレベルの表のみを参照できます。
- 副問合せには AND 演算子を含められますが、各 OR 演算子は 1 行に含まれている列のみを参照できます。1 つの副問合せ内の複数の OR 演算子は、AND 演算子を使用して接続できます。
- 副問合せ内で参照されている表はすべて、同一のマスター・サイトまたはマスター・マテリアライズド・ビュー・サイト内にある必要があります。

注意： CREATE MATERIALIZED VIEW 文に ON PREBUILT TABLE 句および副問合せが含まれている場合、この副問合せは多対多として扱われます。したがって、この場合は、結合列をマテリアライズド・ビュー・ログに記録する必要があります。CREATE MATERIALIZED VIEW 文の ON PREBUILT TABLE 句の詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

関連項目： 主キー・マテリアライズド・ビューの詳細は、3-7 ページの「主キー・マテリアライズド・ビュー」を参照してください。

マテリアライズド・ビューの高速リフレッシュ機能に関する判断

マテリアライズド・ビューの副問合せが前述の制限事項を満たすかどうかを判断するには、高速リフレッシュを指定してマテリアライズド・ビューを作成します。副問合せマテリアライズド・ビューに関する制限事項に違反している場合は、エラーが戻されます。強制リフレッシュを指定した場合、Oracle では高速リフレッシュが実行できなければ完全リフレッシュが自動的に実行されるため、エラーは戻されません。

また、DBMS_MVIEW パッケージ内の EXPLAIN_MVIEW プロシージャを使用して、既存のマテリアライズド・ビュー、あるいはまだ作成されていないマテリアライズド・ビューに関して次の情報を判断できます。

- マテリアライズド・ビューの機能
- 各機能を使用できるかどうか
- 機能を使用できない理由（機能を使用できない場合）

この情報は、VARRAY または MV_CAPABILITIES_TABLE に格納できます。この表に情報を格納する場合は、EXPLAIN_MVIEW プロシージャを実行する前に、*$ORACLE_HOME*/rdbms/admin ディレクトリにある utlxmv.sql スクリプトを実行してこの表を作成する必要があります。

たとえば、oe.orders マテリアライズド・ビューの機能を判断するには、次のように入力します。

```
EXECUTE DBMS_MVIEW.EXPLAIN_MVIEW ('oe.orders');
```

または、マテリアライズド・ビューがまだ作成されていない場合は、その作成に使用する問合せを指定できます。

```
BEGIN
  DBMS_MVIEW.EXPLAIN_MVIEW ('SELECT * FROM oe.orders@orc1.world o
    WHERE EXISTS (SELECT * FROM oe.customers@orc1.world c
    WHERE o.customer_id = c.customer_id AND c.credit_limit > 500)');
END;
/
```

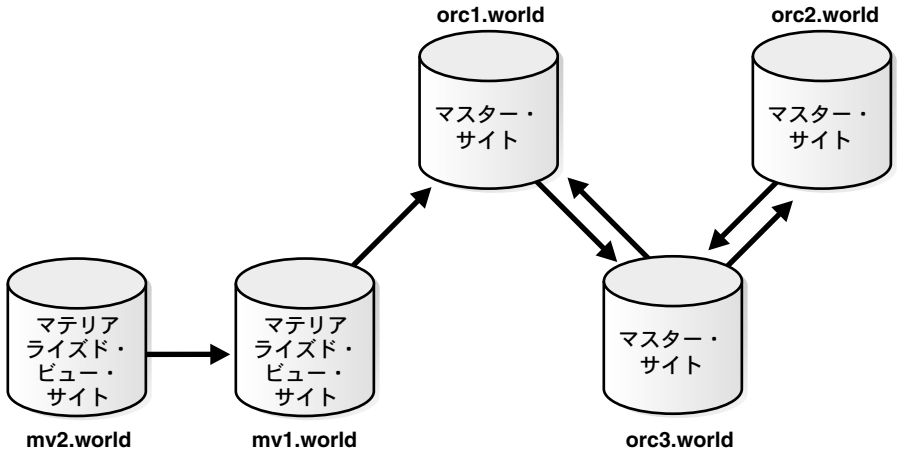
MV_CAPABILITIES_TABLE を問い合せて結果を表示します。

関連項目： EXPLAIN_MVIEW プロシージャの詳細は、『Oracle9*i* データ・ウェアハウス・ガイド』を参照してください。

多重化マテリアライズド・ビュー

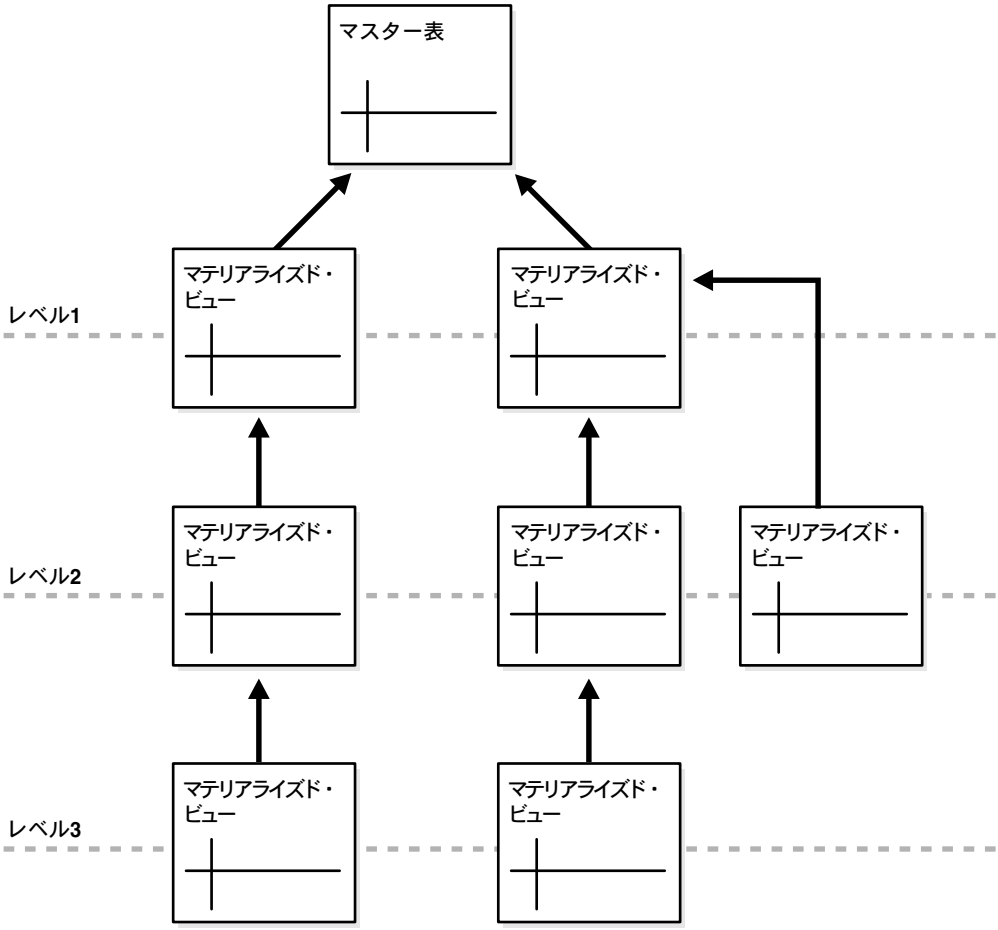
他のマテリアライズド・ビューに基づいたマテリアライズド・ビューを作成する機能により、**多重化マテリアライズド・ビュー**を作成できます。他のマテリアライズド・ビューに基づいたマテリアライズド・ビューは、読取り専用または更新可能です。図 3-6 の矢印は、データベース・リンクを表します。

図 3-6 多重化マテリアライズド・ビュー



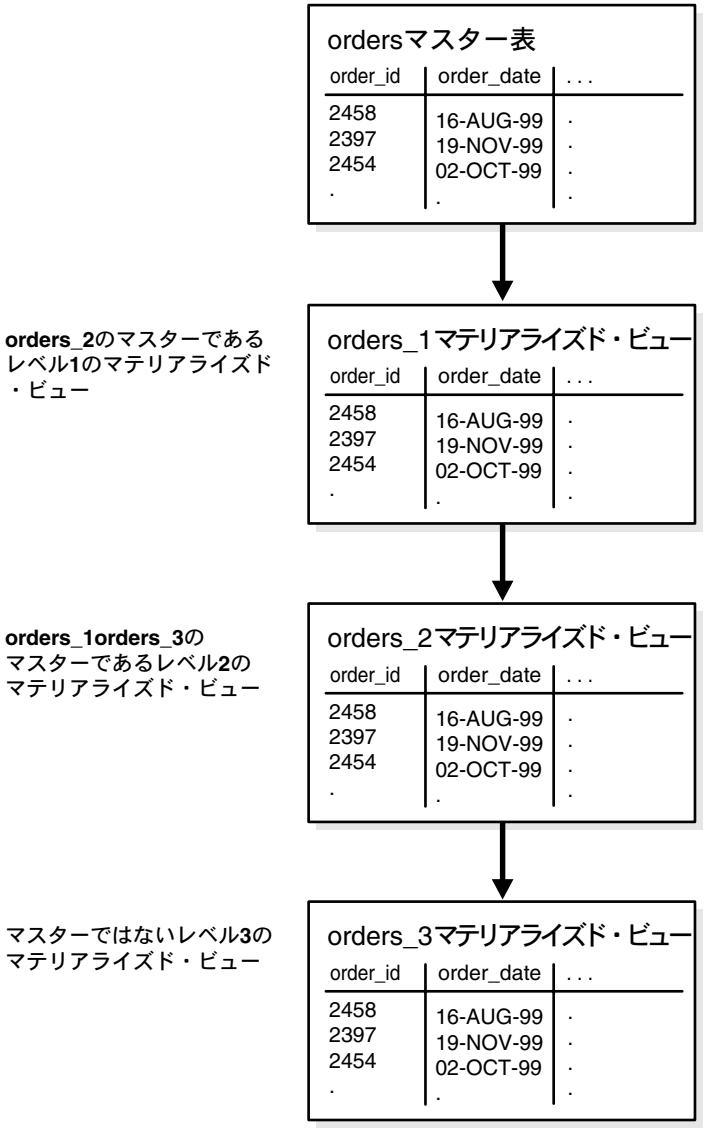
多重化マテリアライズド・ビューを使用する場合、マスター表に基づいたマテリアライズド・ビューはレベル 1 のマテリアライズド・ビューと呼ばれます。次に、レベル 1 のマテリアライズド・ビューに基づいたマテリアライズド・ビューは、レベル 2 のマテリアライズド・ビューと呼ばれます。その次はレベル 3 というようになります。図 3-7 は、これらのレベルを示します。

図 3-7 マテリアライズド・ビューのレベル



他のマテリアライズド・ビューのマスターとして機能するマテリアライズド・ビューは、**マスター・マテリアライズド・ビュー**と呼ばれます。どのレベルのマテリアライズド・ビューでもマスター・マテリアライズド・ビューにできます。図 3-7 に示したように、マスター・マテリアライズド・ビューに基づいたマテリアライズド・ビューは複数あります。図 3-7 では、レベル 2 の 2 つのマテリアライズド・ビューがレベル 1 の 1 つのマテリアライズド・ビューに基づいています。図 3-8 は、レベル 1（orders_1）とレベル 2（orders_2）にあるマスター・マテリアライズド・ビューを示します。

図 3-8 マスター・マテリアライズド・ビュー



レベル 1 のマテリアライズド・ビュー orders_1 のマスターは、マスター・サイトにあるマスター表 orders ですが、レベル 2 から開始すると、各マテリアライズド・ビューにはその上のレベルにマスター・マテリアライズド・ビューがあります。たとえば、レベル 2 のマテリアライズド・ビュー orders_2 のマスターは、レベル 1 のマテリアライズド・ビュー orders_1 です。

マスター・マテリアライズド・ビューは、マスター・サイトのマスター表と同じように機能します。つまり、レベル 2 のマテリアライズド・ビューからレベル 1 のマテリアライズド・ビューにプッシュされる変更は、レベル 1 のマテリアライズド・ビューからマスター表にプッシュされる変更とまったく同じ方法で処理されます。

マスター・マテリアライズド・ビュー・サイトには受信者を登録する必要があります。受信者は、マスター・マテリアライズド・ビューに基づいた多重化マテリアライズド・ビュー・サイトのプロパゲータから遅延トランザクションを受信し、それを適用する必要があります。

関連項目： 2-19 ページ「受信者」

多重化マテリアライズド・ビューを使用すると、レプリケーション環境を柔軟に設計できます。マテリアライズド・ビュー・サイトによっては、マスター表のデータをすべてレプリケートする必要のないサイトもあり、実際、このようなサイトにはすべてのデータを格納する記憶域容量がない場合があります。さらに、レプリケートするデータが少ないことにより、ネットワーク上のアクティビティも減少します。

多重化マテリアライズド・ビューは、3 レベル以上の構造を持つ組織や、ネットワーク・リソースの制約がある組織に理想的です。たとえば、統合本社、各国本社、各国支店を持つ会社があるとします。この会社では、各国本社レベルと支店レベルにデータをレプリケートするコンピュータが多数あります。この場合、レプリケーション環境は、統合本社にマスター・サイトを、各国本社レベルにマテリアライズド・ビューを配置して構成できます。各国本社レベルのマテリアライズド・ビューは、それぞれの国にあてはまるデータ・サブセットのみをマスター表からレプリケートします。ここで多重化マテリアライズド・ビューを使用すると、各国本社レベルのマテリアライズド・ビューに基づいて支店レベルのマテリアライズド・ビューを配置できます。支店レベルのマテリアライズド・ビューには、レベル 1 のマテリアライズド・ビューからのデータで、ローカルな顧客にあてはまるデータ・サブセットのみが含まれます。

多重化マテリアライズド・ビューの使用例

全社員の情報を本社（米国）でメンテナンスする多国籍企業があったとします。この会社では、hr スキーマ内の表を使用して社員情報をメンテナンスします。この会社には 14 か国に本店がそれぞれ 1 つと、その 14 か国の主要都市に地区オフィスが多数あります。

たとえば、英国全体で本店が 1 つありますが、ロンドンにもオフィスが 1 つあります。英国本店では英国全土の全社員の情報をメンテナンスし、ロンドン・オフィスではこのオフィスの社員のみの社員情報をメンテナンスします。この例では、hr.employees マスター表が米国の本社にあり、各地区オフィスには必要な社員情報のみを含む hr.employees マテリアライズド・ビューがあります。

次の文では、英国本店の hr.employees マテリアライズド・ビューを作成します。この文は、本社のデータベース内のマスター表 orc1.world に問い合せます。この文では、country_id が UK の社員のみがマテリアライズド・ビューに含まれるように、副問合せを使用しています。

```
CREATE MATERIALIZED VIEW hr.employees REFRESH FAST FOR UPDATE AS
  SELECT * FROM hr.employees@orc1.world e
    WHERE EXISTS
      (SELECT * FROM hr.departments@orc1.world d
       WHERE e.department_id = d.department_id
       AND EXISTS
         (SELECT * FROM hr.locations@orc1.world l
          WHERE l.country_id = 'UK'
          AND d.location_id = l.location_id));
```

注意： この hr.employees マテリアライズド・ビューを作成するには、次の列をロギングする必要があります。

- department_id 列を、orc1.world にある hr.employees マスター表のマテリアライズド・ビュー・ログにロギングする必要があります。
- country_id 列を、orc1.world にある hr.locations マスター表のマテリアライズド・ビュー・ログにロギングする必要があります。

詳細は、6-16 ページの「マテリアライズド・ビュー・ログへの列のロギング」を参照してください。

次の文では、英国本店のレベル 1 のマテリアライズド・ビューに基づいてロンドン・オフィスの hr.employees マテリアライズド・ビューを作成します。この文は、英国本店のデータベースにあるマテリアライズド・ビュー reg_uk.world に問い合せます。この文では、city_id が London の社員のみがマテリアライズド・ビューに含まれるように、副問合せを使用しています。

```
CREATE MATERIALIZED VIEW hr.employees REFRESH FAST FOR UPDATE AS
  SELECT * FROM hr.employees@reg_uk.world e
    WHERE EXISTS
      (SELECT * FROM hr.departments@reg_uk.world d
       WHERE e.department_id = d.department_id
       AND EXISTS
         (SELECT * FROM hr.locations@reg_uk.world l
          WHERE l.city = 'London'
          AND d.location_id = l.location_id));
```

注意： この hr.employees マテリアライズド・ビューを作成するには、次の列をロギングする必要があります。

- department_id 列を、reg_uk.world にある hr.employees マスター表のマテリアライズド・ビュー・ログにロギングする必要があります。
- country_id 列を、reg_uk.world にある hr.locations マスター表のマテリアライズド・ビュー・ログにロギングする必要があります。

詳細は、6-16 ページの「マテリアライズド・ビュー・ログへの列のロギング」を参照してください。

多重化マテリアライズド・ビューの使用上の制限事項

マスター・マテリアライズド・ビューおよびマテリアライズド・ビューに基づいたマテリアライズド・ビューには、次の制限事項があります。

- どちらも主キー・マテリアライズド・ビューである必要があります。
- どちらもリリース 9.0.1 以上の互換性のレベルのデータベースに入れる必要があります。

注意： COMPATIBLE 初期化パラメータにより、データベースの互換性のレベルを制御されます。

多重化マテリアライズド・ビューに対するその他の制限事項 次のタイプのマテリアライズド・ビューは、更新可能マテリアライズド・ビューのマスターとしては指定できません。

- ROWID マテリアライズド・ビュー
- 複合マテリアライズド・ビュー
- 読取り専用マテリアライズド・ビュー

ただし、これらのマテリアライズド・ビューは、読取り専用マテリアライズド・ビューのマスターとして指定できます。

マテリアライズド・ビューに基づいた更新可能マテリアライズド・ビューに対するその他の制限事項 マテリアライズド・ビューに基づいた更新可能マテリアライズド・ビューには、次の制限事項があります。

- マスター・マテリアライズド・ビュー・サイトのマテリアライズド・ビュー・グループと同じ名前のマテリアライズド・ビュー・グループに属する必要があります。
- マスター・マテリアライズド・ビュー・サイトのマテリアライズド・ビュー・グループとは別のデータベースに入れる必要があります。

- 読取り専用マテリアライズド・ビューではなく、別の更新可能マテリアライズド・ビュー（複数も可）に基づいている必要があります。
- マスター・マテリアライズド・ビュー・サイトの PUBLIC により所有されるマテリアライズド・ビュー・グループ内のマテリアライズド・ビューに基づいている必要があります。

ユーザー定義型を含むマテリアライズド・ビュー

Oracle の**オブジェクト型**はユーザー定義データ型で、これを使用すると現実の複雑なエンティティ（顧客や注文など）をデータベース内に単一エンティティ（**オブジェクト**）としてモデル化できます。オブジェクト型は、CREATE TYPE . . .AS OBJECT 文を使用して作成します。オブジェクト型とオブジェクトは、レプリケーション環境のマスター・サイトとマテリアライズド・ビュー・サイト間でレプリケートできます。

表の 1 列を占有する Oracle オブジェクトを、**列オブジェクト**と呼びます。一般に、列オブジェクトが含まれている表にはその他の列も含まれており、このような列のデータ型は組込みデータ型（VARCHAR2 や NUMBER など）の場合があります。**オブジェクト表**とは、各行がオブジェクトを表す特殊な表です。オブジェクト表の各行は**行オブジェクト**です。

コレクションをレプリケートすることもできます。コレクションとは、VARRAY データ型およびネストした表のデータ型に基づいたユーザー定義データ型です。VARRAY は CREATE TYPE . . .AS VARRAY 文を、ネストした表は CREATE TYPE . . .AS TABLE 文を使用して作成します。

注意：

- ユーザー定義型とユーザー定義型に基づいたオブジェクトをレプリケートするには、マスター・サイトとマテリアライズド・ビュー・サイトの互換性のレベルが、どちらもリリース 9.0.1 以上である必要があります。互換性のレベルは、COMPATIBLE 初期化パラメータによって制御されます。
- ON COMMIT オプションが指定されたマテリアライズド・ビューを、ユーザー定義型のマスターに基づいて作成することはできません。ON COMMIT オプションが指定されたマテリアライズド・ビューとは、CREATE MATERIALIZED VIEW 文で ON COMMIT REFRESH 句を使用して作成するマテリアライズド・ビューです。
- Advanced Replication では、型の継承をサポートしていません。

関連項目： ユーザー定義型、Oracle オブジェクトおよびコレクションの詳細は、『Oracle9*i* アプリケーション開発者ガイド - オブジェクト・リレーショナル機能 』を参照してください。この項では、このマニュアルに記載されている基本的な情報を理解していることを前提としています。

レプリケーション・サイトでの型の一致

ユーザー定義型には、オブジェクト、ネストした表、VARRAY および索引タイプなど、CREATE TYPE 文で作成された型がすべて含まれます。ユーザー定義型に基づいたスキーマ・オブジェクトをレプリケートするには、全レプリケーション・サイトに同一のユーザー定義型自体が存在する必要があります。さらに、ユーザー定義型が使用されるレプリケーション・グループにユーザー定義型を追加することをお薦めします。ただし、これは必須ではありません。

ユーザー定義型とそれに基づいたスキーマ・オブジェクトをレプリケートする場合は、次の条件が適用されます。

- マスター・サイトとマテリアライズド・ビュー・サイトでレプリケートされるユーザー定義型は、これらの型に依存するマテリアライズド・ビューを作成する前にマテリアライズド・ビュー・サイトで作成する必要があります。
- ユーザー定義型を持つマテリアライズド・ビューを作成するには、マテリアライズド・ビューの基になるマスターがすべて同一マスター・サイト上にある必要があります。
- ユーザー定義型は全レプリケーション・サイトで同一にする必要があります。
  - レプリケートされる各ユーザー定義型について、オブジェクト識別子（OID）、スキーマ所有者および型の名前を全レプリケーション・サイトで同一にする必要があります。
  - ユーザー定義型がオブジェクト型の場合は、全レプリケーション・サイトで、オブジェクト型の属性の順序とデータ型が一致する必要があります。属性の順序とデータ型は、オブジェクト型を作成するときに設定します。たとえば、次のようなオブジェクト型の場合を考えます。

    ```
    CREATE TYPE cust_address_typ AS OBJECT
         (street_address     VARCHAR2(40),
          postal_code        VARCHAR2(10),
          city               VARCHAR2(30),
          state_province     VARCHAR2(10),
          country_id         CHAR(2));
    /
    ```

    全レプリケーション・サイトで、street_address はこの型の最初の属性で VARCHAR2(40)、postal_code は 2 番目の属性で VARCHAR2(10)、city は 3 番目の属性で VARCHAR2(30)、というようになっている必要があります。
  - 全レプリケーション・サイトで、ユーザー定義型のハッシュコードが一致する必要があります。Oracle では、ユーザー定義型が検査され、ハッシュコードが割り当てられます。この検査では、型の属性、属性の順序および型の名前が調べられます。複数の型についてこれらの項目がすべて同じである場合、型のハッシュコードは同じです。型のハッシュコードは、DBA_TYPE_VERSIONS データ・ディクショナリ・ビューを問い合せて表示できます。

全レプリケーション・サイトでユーザー定義型を同一にするには、次のいずれかの方法でマテリアライズド・ビュー・サイトにユーザー定義型を作成する必要があります。

- レプリケーション・マネージメント API の使用
- CREATE TYPE 文の使用

レプリケーション・マネージメント API の使用 マテリアライズド・ビュー・サイトでユーザー定義型を含むレプリケート・オブジェクトを作成、変更または削除する場合は、レプリケーション・マネージメント API の使用をお薦めします。レプリケーション・マネージメント API を使用しない場合、レプリケーション・エラーが発生することがあります。

マスター・サイトとマテリアライズド・ビュー・サイトで同一のユーザー定義型を作成するには、DBMS_REPCAT パッケージ内の CREATE_MVIEW_REPOBJECT プロシージャを使用します。このプロシージャは型を作成してマテリアライズド・ビュー・グループに追加します。マテリアライズド・ビュー・サイトからユーザー定義型を削除するには、DBMS_REPCAT パッケージ内の DROP_MVIEW_REPOBJECT プロシージャを使用します。

> **関連項目：**『Oracle9*i* レプリケーション・マネージメント API リファレンス』

CREATE TYPE 文の使用 型を作成するには、マテリアライズド・ビュー・サイトで CREATE TYPE 文を使用できます。型を使用する読取り専用マテリアライズド・ビューを作成し、この読取り専用マテリアライズド・ビューをマテリアライズド・ビュー・グループに追加しない場合は、この方法が必要になることがあります。

この方法を選択する場合は、次のことを確認する必要があります。

- マテリアライズド・ビュー・サイトとマスター・サイトで型が同一スキーマ内にあること。
- マテリアライズド・ビュー・サイトとマスター・サイトで型の属性および順序が同一であること。
- マテリアライズド・ビュー・サイトとマスター・サイトで属性の型が同一のデータ型であること。
- マテリアライズド・ビュー・サイトとマスター・サイトで型のオブジェクト識別子が同一であること。

型のオブジェクト識別子は、DBA_TYPES データ・ディクショナリ・ビューを問い合せると検出できます。たとえば、cust_address_typ のオブジェクト識別子（OID）を検出するには、次の問合せを入力します。

```
SELECT TYPE_OID FROM DBA_TYPES WHERE TYPE_NAME = 'CUST_ADDRESS_TYP';

TYPE_OID
--------------------------------
6F9BC33653681B7CE03400400B40A607
```

これで、マスター・サイトでの型の OID が認識できたため、次の手順を実行して、マテリアライズド・ビュー・サイトでこの型を作成します。

1. マスター・サイトでこの型を所有するユーザーとして、マテリアライズド・ビュー・サイトにログインします。マテリアライズド・ビュー・サイトにこのユーザーがない場合はユーザーを作成します。
2. CREATE TYPE 文を発行し、OID を指定します。

```
CREATE TYPE oe.cust_address_typ OID '6F9BC33653681B7CE03400400B40A607'
     AS OBJECT (
     street_address     VARCHAR2(40),
     postal_code        VARCHAR2(10),
     city               VARCHAR2(30),
     state_province     VARCHAR2(10),
     country_id         CHAR(2));
/
```

これで、マテリアライズド・ビュー・サイトでこの型を使用できます。

列オブジェクトを使用したマスターの列のサブセット化

読取り専用マテリアライズド・ビューでは、その他の属性はレプリケートせずに列オブジェクトの特定の属性のみをレプリケートできます。たとえば、前述の cust_address_typ ユーザー定義データ型を使用して、customers マスター表がマスター・サイト orc1.world に作成されているとします。

```
CREATE TABLE oe.customers (
      customer_id        NUMBER(6),
      cust_first_name    VARCHAR2(20),
      cust_last_name     VARCHAR2(20),
      cust_address       cust_address_typ);
```

次の読取り専用マテリアライズド・ビューをリモート・マテリアライズド・ビュー・サイトに作成できます。

```
CREATE MATERIALIZED VIEW oe.customers_mv1 AS
   SELECT customer_id, cust_last_name, c.cust_address.postal_code
   FROM oe.customers@orc1.world c;
```

ここでは、postal_code 属性が cust_address 列オブジェクト内に指定されています。

更新可能マテリアライズド・ビューでは列オブジェクト全体をレプリケートします。列オブジェクトの属性を部分的にレプリケートすることはできません。次の文は cust_address 列オブジェクト全体を指定しているため有効です。

```
CREATE MATERIALIZED VIEW oe.customers_mv1 FOR UPDATE AS
   SELECT customer_id, cust_last_name, cust_address
   FROM oe.customers@orc1.world;
```

関連項目： 配置テンプレートを使用した列のサブセット化の詳細は、4-21 ページの「配置テンプレートを使用した列のサブセット化」を参照してください。列のサブセット化は、配置テンプレートを使用した場合にのみサポートされます。

オブジェクト表に基づいたマテリアライズド・ビュー

マテリアライズド・ビューがオブジェクト表に基づいて OF *type* 句を使用して作成されている場合、このマテリアライズド・ビューは**オブジェクト・マテリアライズド・ビュー**と呼ばれます。オブジェクト・マテリアライズド・ビューは、オブジェクト表と同じ構造です。つまり、オブジェクト・マテリアライズド・ビューは行オブジェクトで構成されます。オブジェクト表に基づいたマテリアライズド・ビューが OF *type* 句を使用せずに作成されている場合、このマテリアライズド・ビューは読取り専用で、オブジェクト・マテリアライズド・ビューではありません。つまり、このようなマテリアライズド・ビューには、行オブジェクトではなく通常の行が含まれています。

オブジェクト表に基づいたマテリアライズド・ビューを作成するには、マテリアライズド・ビューが依存する型がマテリアライズド・ビュー・サイトに必要で、それぞれの型にはマスター・サイトと同じオブジェクト識別子が必要です。

OF *type* 句を使用したオブジェクト・マテリアライズド・ビューの作成 必要な型をマテリアライズド・ビュー・サイトに作成した後、OF *type* 句を指定してオブジェクト・マテリアライズド・ビューを作成できます。

たとえば、次の SQL 文で orc1.world マスター・サイトに categories_tab オブジェクト表を作成するとします。

```
CREATE TYPE category_typ AS OBJECT
   (category_name           VARCHAR2(50),
    category_description    VARCHAR2(1000),
    category_id             NUMBER(2))
NOT FINAL;
/

CREATE TABLE categories_tab OF category_typ
    (category_id    PRIMARY KEY);
```

categories_tab マスター表に基づいて高速リフレッシュが可能なマテリアライズド・ビューを作成するには、この表のマテリアライズド・ビュー・ログを作成します。

```
CREATE MATERIALIZED VIEW LOG ON categories_tab WITH OBJECT ID;
```

オブジェクト表に対するマテリアライズド・ビュー・ログを作成する場合には、WITH OBJECT ID 句が必要です。

category_typ タイプをマテリアライズド・ビュー・サイトに作成した後、次の SQL 文のように OF *type* 句を使用して、categories_tab オブジェクト表に基づいたオブジェクト・マテリアライズド・ビューを作成できます。

```
CREATE MATERIALIZED VIEW categories_objmv OF category_typ
   REFRESH FAST FOR UPDATE
   AS SELECT * FROM categories_tab@orc1.world;
```

ここで、*type* は category_typ です。

注意： 型は、マテリアライズド・ビュー・サイトとマスター・サイトで同一にする必要があります。詳細は、3-33 ページの「レプリケーション・サイトでの型の一致」を参照してください。

オブジェクト表に基づいて OF *type* 句を使用せずに作成したマテリアライズド・ビュー オブジェクト表に基づいて OF *type* 句を使用せずにマテリアライズド・ビューを作成した場合、このマテリアライズド・ビューは読取り専用となり、基になるオブジェクト表のオブジェクト・プロパティは失われます。つまり、作成された読取り専用マテリアライズド・ビューにはマスターの列が 1 つ以上含まれますが、それぞれの行はリレーショナル表の行として機能します。行は行オブジェクトではありません。

たとえば、次の SQL 文を使用して、categories_tab マスター表に基づいたマテリアライズド・ビューを作成できます。

```
CREATE MATERIALIZED VIEW categories_relmv
   AS SELECT * FROM categories_tab@orc1.world;
```

この場合、categories_relmv マテリアライズド・ビューは読取り専用で、このマテリアライズド・ビューの行は、リレーショナル表の行と同じように機能します。

オブジェクト・マテリアライズド・ビューでの OID の保持 オブジェクト・マテリアライズド・ビューは、マスターのオブジェクト識別子（OID）指定を継承します。マスターに主キー・ベースの OID がある場合、マテリアライズド・ビューの行オブジェクトの OID は主キー・ベースになります。マスターにシステム生成された OID がある場合、マテリアライズド・ビューの行オブジェクトの OID はシステム生成された OID です。また、オブジェクト・マテリアライズド・ビューの各行の OID は、マスター内の同じ行の OID と一致し、マテリアライズド・ビューのリフレッシュ中、OID は保持されます。この結果、オブジェクト表の行に対する REF は、マテリアライズド・ビュー・サイトでも有効になります。

コレクション列を含むマテリアライズド・ビュー

コレクション列とは、VARRAY データ型およびネストした表のデータ型に基づいた列です。Oracle ではコレクション列を含むマテリアライズド・ビューの作成をサポートしています。

コレクション列がネストした表の場合は、マテリアライズド・ビューの作成中にオプションとして *nested_table_storage_clause* を指定できます。*nested_table_storage_clause* を使用すると、ネストした表用の記憶表の名前をマテリアライズド・ビューに指定できます。たとえば、マスター・サイト orc1.world にマスター表 people_reltab を作成し、この表にネストした表 phones_ntab が含まれているとします。

```
CREATE TYPE phone_typ AS OBJECT (
   location    VARCHAR2(15),
   num         VARCHAR2(14));
/

CREATE TYPE phone_ntabtyp AS TABLE OF phone_typ;
/

CREATE TABLE people_reltab (
   id               NUMBER(4) CONSTRAINT pk_people_reltab PRIMARY KEY,
   first_name       VARCHAR2(20),
   last_name        VARCHAR2(20),
   phones_ntab      phone_ntabtyp)
   NESTED TABLE phones_ntab STORE AS phone_store_ntab
   ((PRIMARY KEY (NESTED_TABLE_ID, location)));
```

この SQL 文の最後の行の PRIMARY KEY 指定に注意してください。親表に基づいてマテリアライズド・ビューを作成する場合は、記憶表に主キーを指定する必要があります。この例では、記憶表は phone_store_ntab で、親表は people_reltab です。

高速リフレッシュ可能なマテリアライズド・ビューを作成する場合は、親表と記憶表の両方にマテリアライズド・ビュー・ログを作成し、ネストした表の列を親表のマテリアライズド・ビュー・ログのフィルタ列として指定します。

```
CREATE MATERIALIZED VIEW LOG ON people_reltab;

ALTER MATERIALIZED VIEW LOG ON people_reltab ADD(phones_ntab);

CREATE MATERIALIZED VIEW LOG ON phone_store_ntab WITH PRIMARY KEY;
```

マテリアライズド・ビュー・サイトで必要な型を作成し、それぞれの型のオブジェクト識別子がマスター・サイトのオブジェクト識別子と同じになるようにします。この後、people_reltab に基づいてマテリアライズド・ビューを作成し、次の文を使用してその記憶表を指定できます。

```
CREATE MATERIALIZED VIEW people_reltab_mv
   NESTED TABLE phones_ntab STORE AS phone_store_ntab_mv
   REFRESH FAST AS SELECT * FROM people_reltab@orc1.world;
```

この例では、*nested_table_store_clause* は前の例で「NESTED TABLE」で始まる行で、記憶表の名前が phone_store_ntab_mv であることを指定します。*nested_table_storage_clause* はオプションです。この句を指定しない場合、記憶表には自動的に名前が付けられます。記憶表の名前を表示するには、DBA_NESTED_TABLES データ・ディクショナリ・ビューを問い合せます。

記憶表には次の特性があります。

- 個別のセカンダリ・マテリアライズド・ビューです。
- ネストした表を含むマテリアライズド・ビューをリフレッシュすると、自動的にリフレッシュされます。
- ネストした表を含むマテリアライズド・ビューを削除すると、自動的に削除されます。
- マスターの記憶表の主キー制約を継承します。

記憶表はマスターの記憶表の主キー制約を継承するため、STORE AS 句には PRIMARY KEY を指定しないでください。

マテリアライズド・ビュー内のネストした表の記憶表に対して次のアクションを直接実行することはできません。

- 記憶表のリフレッシュ
- レプリケーション・グループへの記憶表の追加
- 記憶表の変更
- 記憶表の削除
- 記憶表に対するレプリケーション・サポートの生成

これらのアクションは、ネストした表を含むマテリアライズド・ビューに対する実行時に間接的に発生します。さらに、記憶表内の列のサブセットはレプリケートできません。

> **関連項目：** *nested_table_storage_clause* の詳細は、『Oracle9*i* SQL リファレンス』を参照してください。CREATE TABLE 文に詳しく説明されています。

コレクション列を含むマテリアライズド・ビューに関する制限事項 コレクション列を含むマテリアライズド・ビューには、次の制限事項があります。

- コレクション列の行のサブセット化はできません。ただし、ネストした表の親表に対しては行のサブセット化を使用でき、その結果、マスター・マテリアライズド・ビューのネストした表がサブセット化されます。
- コレクション列の列のサブセット化はできません。
- ネストした表の記憶表には主キーが必要です。
- ネストした表の親表で高速リフレッシュ可能にするには、親表とネストした表の両方の記憶表にマテリアライズド・ビュー・ログが必要です。

REF 列を含むマテリアライズド・ビュー

REF 列を含むマテリアライズド・ビューを作成できます。REF とは Oracle の組込みデータ型で、オブジェクト表内の行オブジェクトを指す論理的なポインタです。有効範囲付 REF は、指定された表のみへの参照を含む REF で、有効範囲なし REF は、データベース内で対応するオブジェクト型に基づいたすべてのオブジェクト表に対する参照を含められます。有効範囲付 REF は、有効範囲なし REF と比べて必要とする記憶領域が少なく、より効率的なアクセスを提供します。

次の項からわかるように、マテリアライズド・ビューの作成中に REF 列の有効範囲を変更して、マテリアライズド・ビュー・サイトのローカルなマテリアライズド・ビューまたは表を指すように変更できます。REF 列の有効範囲を変更しないと、リモートのマスターを指したままになります。有効範囲なし REF 列は、常にマスターを指します。マテリアライズド・ビューの REF 列がリモート・マスターを指す場合、REF には参照先がないとみなされます。SQL では、参照先がない REF を参照解除すると、NULL が返されます。また、PL/SQL では UTL_OBJECT パッケージを使用した REF の参照解除のみがサポートされ、参照先がない REF に対しては例外が発生します。

有効範囲付 REF 列 有効範囲付 REF 列を含むマスターに基づいてマテリアライズド・ビューを作成する場合は、REF の有効範囲をマテリアライズド・ビュー・サイトの別のオブジェクト表またはオブジェクト・マテリアライズド・ビューに変更できます。通常、REF 列の有効範囲は、元のリモート・オブジェクト表ではなくローカル・オブジェクト・マテリアライズド・ビューに変更します。マテリアライズド・ビューの有効範囲を変更するには、CREATE MATERIALIZED VIEW 文に SCOPE FOR 句を使用するか、マテリアライズド・ビューを作成した後で ALTER MATERIALIZED VIEW 文を使用します。REF 列の有効範囲を変更しないと、マテリアライズド・ビューはマスターの REF 列をそのまま保持します。

たとえば、次の文を使用して orc1.world マスター・サイトに customers_with_ref マスター表を作成するとします。

```
-- Create the user-defined datatype cust_address_typ.
CREATE TYPE cust_address_typ AS OBJECT
   (street_address     VARCHAR2(40),
    postal_code        VARCHAR2(10),
    city               VARCHAR2(30),
    state_province     VARCHAR2(10),
    country_id         CHAR(2));
/

-- Create the object table cust_address_objtab.
CREATE TABLE cust_address_objtab OF cust_address_typ;

-- Create table with REF to cust_address_typ.
CREATE TABLE customers_with_ref (
     customer_id        NUMBER(6) PRIMARY KEY,
     cust_first_name    VARCHAR2(20),
     cust_last_name     VARCHAR2(20),
     cust_address       REF cust_address_typ SCOPE IS cust_address_objtab);
```

cust_address_typ がマテリアライズド・ビューにあると仮定した場合、次の文を使用すると cust_address_objtab_mv オブジェクト・マテリアライズド・ビューを作成できます。

```
CREATE MATERIALIZED VIEW cust_address_objtab_mv OF cust_address_typ AS
   SELECT * FROM oe.cust_address_objtab@orc1.world;
```

次に、次の文を使用して customers_with_ref マスター表のマテリアライズド・ビューを作成し、REF の有効範囲を cust_address_objtab_mv マテリアライズド・ビューに変更できます。

```
CREATE MATERIALIZED VIEW customers_with_ref_mv
   (SCOPE FOR (cust_address) IS oe.cust_address_objtab_mv)
   AS SELECT * FROM oe.customers_with_ref@orc1.world;
```

マテリアライズド・ビューを作成するときに SCOPE FOR 句を使用する場合は、SCOPE FOR 句に指定するマテリアライズド・ビューまたは表を先に作成する必要があります。そうしないと、マテリアライズド・ビューの作成中に SCOPE FOR 句を指定できません。たとえば、cust_address_objtab_mv マテリアライズド・ビューを作成する前に customers_with_ref_mv を作成した場合は、customers_with_ref_mv マテリアライズド・ビューを作成するときに SCOPE FOR 句を使用できません。この場合、REF はリモート・マスター・サイトのオブジェクト表を指すため、参照先がないとみなされます。

ただし、マテリアライズド・ビューを作成するときに SCOPE FOR 句を使用しない場合でも、SCOPE FOR 句を指定するようにマテリアライズド・ビューを変更できます。たとえば、次の文で customers_with_ref_mv マテリアライズド・ビューを変更できます。

```
ALTER MATERIALIZED VIEW customers_with_ref_mv
   MODIFY SCOPE FOR (cust_address) IS oe.cust_address_objtab_mv;
```

有効範囲なし REF 列 有効範囲なし REF 列を含むリモート・マスターに基づいてマテリアライズド・ビューを作成した場合、マテリアライズド・ビューに REF 列は作成されますが、REF はリモート・データベースを指すため参照先がないとみなされます。

マテリアライズド・ビュー・ログへの REF 列のロギング 必要であれば、REF 列をマテリアライズド・ビュー・ログにロギングできます。

> **関連項目：** 詳細は、6-16 ページの「マテリアライズド・ビュー・ログへの列のロギング」を参照してください。

WITH ROWID 句を使用して作成された REF REF 列に対して WITH ROWID 句を指定すると、REF で参照されているオブジェクトの行 ID がメンテナンスされます。REF に含まれている行 ID を使用して、参照されているオブジェクトを直接検出できます。OID 索引から行 ID をフェッチする必要はありません。したがって、WITH ROWID 句は行 ID ヒントを指定するために使用します。WITH ROWID 句は、有効範囲付 REF に対してはサポートされていません。

WITH ROWID 句を使用して作成された REF をレプリケートすると、REF が最初に作成または変更されたサイトを除き、各レプリケーション・サイトでは行 ID ヒントが正しくなくなります。他のサイトでは REF に含まれている ROWID 情報が無意味になります。行 ID ヒントは自動的には修正されません。無効な行 ID ヒントは、パフォーマンス上の問題の原因になります。この場合は、ANALYZE 文を使用して、行 ID ヒントが正しくないレプリケーション・サイトで行 ID ヒントを修正する必要があります。

> **関連項目：** ANALYZE 文の詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

マスター・サイトまたはマスター・マテリアライズド・ビュー・サイトでのマテリアライズド・ビューの登録

マスター・サイトとマスター・マテリアライズド・ビュー・サイトでは、マテリアライズド・ビューに関する情報がマスター表またはマスター・マテリアライズド・ビューに基づいて自動的に登録されます。次の項では、マテリアライズド・ビューの登録メカニズムについて説明します。

登録されたマテリアライズド・ビューに関する情報の表示

レベル 1 のマテリアライズド・ビューまたはマテリアライズド・ビュー・グループは、マスター・サイトに登録されます。レベル 2 以上の多重化マテリアライズド・ビューまたはマテリアライズド・ビュー・グループは、マスター・サイトではなくマスター・マテリアライズド・ビュー・サイトに登録されます。マスター・サイトまたはマスター・マテリアライズド・ビュー・サイトで DBA_REGISTERED_MVIEWS データ・ディクショナリ・ビューを問い合せると、リモート・マテリアライズド・ビューに関する次の情報を表示できます。

- 所有者、名前およびマテリアライズド・ビューを含むデータベース
- マテリアライズド・ビューの定義問合せ
- マテリアライズド・ビューのその他の特性（リフレッシュ方式など）

さらに、マスター・サイトまたはマスター・マテリアライズド・ビュー・サイトで DBA_MVIEW_REFRESH_TIMES ビューを問い合せると、各リモート・マテリアライズド・ビューの最新リフレッシュ時刻を取得できます。管理者は、この情報を使用してマテリアライズド・ビューのアクティビティを監視し、マスター表またはマスター・マテリアライズド・ビューの削除、変更または再配置が必要な場合にマテリアライズド・ビュー・サイトへの変更を調整できます。

内部メカニズム

Oracle では、マテリアライズド・ビューが作成されると、それをマスター・サイトまたはマスター・マテリアライズド・ビュー・サイトに登録し、マテリアライズド・ビューが削除されるとその登録を解除します。同じことがマテリアライズド・ビュー・グループにも当てはまります。

マスター・マテリアライズド・ビューを削除した場合、これに基づいたマテリアライズド・ビューは自動的には削除されません。これらのマテリアライズド・ビューは手動で削除する必要があります。このようなマテリアライズド・ビューを削除せずに、すでに削除されているマスター・マテリアライズド・ビューに対してそのマテリアライズド・ビューからリフレッシュしようとすると、エラーが戻されます。

たとえば、orders_lev1 というマテリアライズド・ビューが oe.orders マスター表に基づいており、orders_lev2 というマテリアライズド・ビューが orders_lev1 に基づいているとします。orders_lev1 を削除しても、orders_lev2 はそのまま残ります。ここで、orders_lev2 をリフレッシュしようとすると、orders_lev1 はすでに存在しないため、エラーが戻されます。

注意： マテリアライズド・ビューを作成（または削除）するとき、このマテリアライズド・ビューがマスター・サイトまたはマスター・マテリアライズド・ビュー・サイトに登録（または登録解除）されるとはかぎりません。マテリアライズド・ビューを作成中に Oracle で正常に登録できない場合は、DBMS_MVIEW パッケージ内の REGISTER_MVIEW プロシージャを使用して手動で登録を完了する必要があります。マテリアライズド・ビューを削除するときに Oracle で正常にその登録を解除できない場合は、手動で登録を解除するまで、そのマテリアライズド・ビューの登録情報がマスター・サイトまたはマスター・マテリアライズド・ビュー・サイトに残ります。複合マテリアライズド・ビューは登録されない場合があります。

注意： Oracle7 のマスター・サイトでは、マテリアライズド・ビューは登録できません。

マテリアライズド・ビューの手動による登録

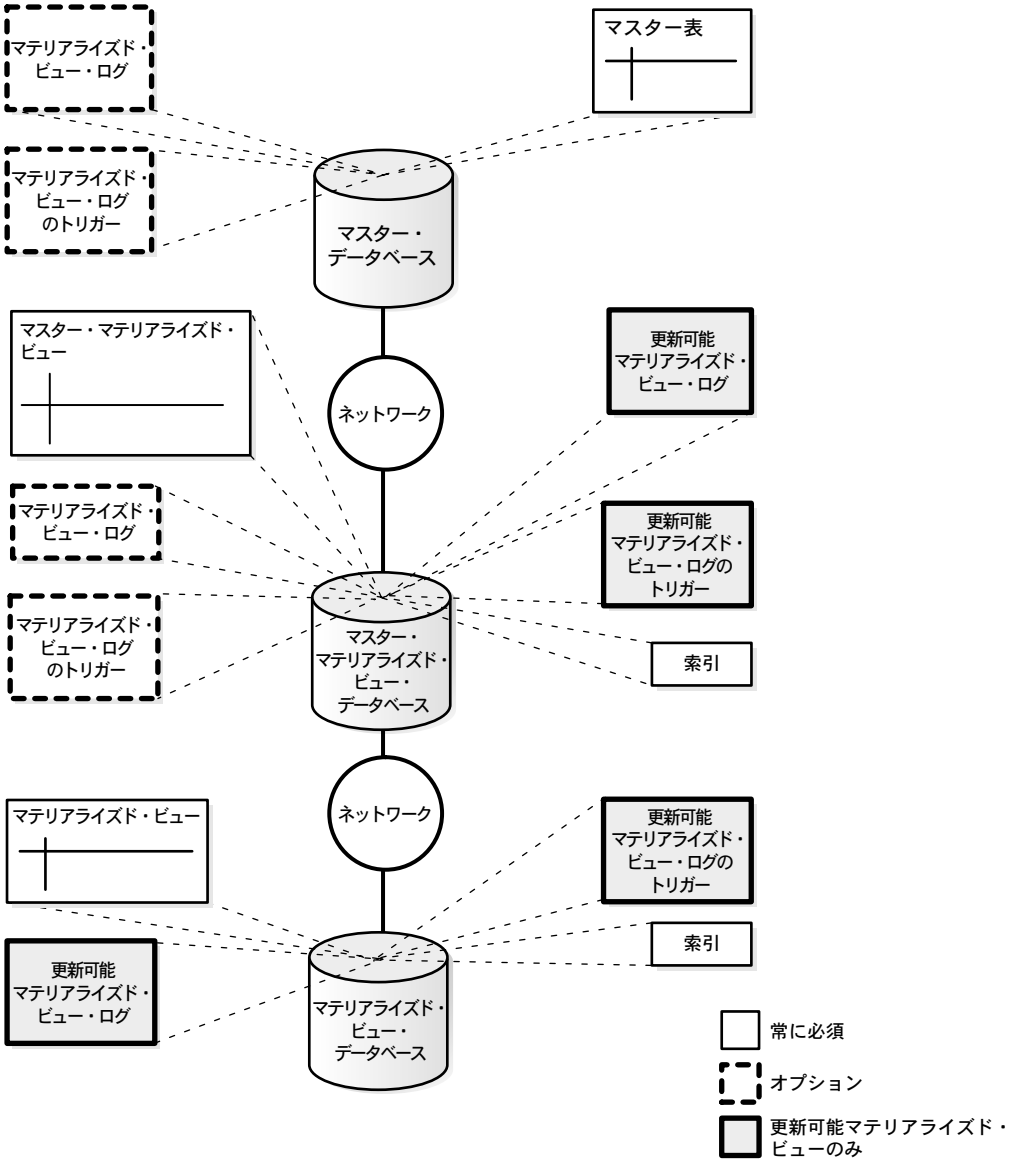
必要な場合は、登録を手動でメンテナンスできます。マテリアライズド・ビューの登録情報を追加、変更または削除するには、DBMS_MVIEW パッケージにある REGISTER_MVIEW および UNREGISTER_MVIEW プロシージャを使用します。

関連項目： REGISTER_MVIEW プロシージャおよび UNREGISTER_MVIEW プロシージャの詳細は、『Oracle9*i* レプリケーション・マネージメント API リファレンス』を参照してください。

マテリアライズド・ビューのアーキテクチャ

図 3-9 は、マテリアライズド・ビューのレプリケーションに使用されるオブジェクトを示します。これらのオブジェクトの一部はオプションで、作成したマテリアライズド・ビュー環境のサポートに必要な場合にのみ使用します。たとえば、読取り専用マテリアライズド・ビューがある場合は、マテリアライズド・ビュー・サイトには更新可能マテリアライズド・ビュー・ログや内部トリガーはありません。また、高速リフレッシュができない複合マテリアライズド・ビューがある場合は、マスター・サイトにマテリアライズド・ビュー・ログがない可能性があります。

図 3-9 マテリアライズド・ビューのレプリケーション・オブジェクト



マスター・マテリアライズド・ビューには、マテリアライズド・ビュー・ログと更新可能マテリアライズド・ビュー・ログの両方が含まれている場合があります。マスター・マテリアライズド・ビュー・サイトを計画する場合は、この 2 つのログに必要な追加領域を考慮してください。

マスター・サイトおよびマスター・マテリアライズド・ビュー・サイトのメカニズム

マテリアライズド・ビューの高速リフレッシュをサポートするために、図 3-10 に示されている 3 つのメカニズムがマスター・サイトとマテリアライズド・ビュー・サイトの両方で必要です。

注意： マスター・マテリアライズド・ビューには、この項で説明されているメカニズムの他に、3-50 ページの「マテリアライズド・ビュー・サイトのメカニズム」で説明されているメカニズムも含まれています。

図 3-10 マスター・サイトおよびマスター・マテリアライズド・ビュー・サイトのオブジェクト

マスター表およびマスター・マテリアライズド・ビュー

マスター表またはマスター・マテリアライズド・ビューは、マテリアライズド・ビューの基礎です。マスター表はターゲット・マスター・サイトにあり、マスター・マテリアライズド・ビューはマスター・マテリアライズド・ビュー・サイトにあります。マスターがマスター表の場合、この表はマテリアライズド・ビュー・レプリケーションとマルチマスター・レプリケーションの両方に関与している可能性があります。マテリアライズド・ビューが指すマスター・サイトまたはマスター・マテリアライズド・ビュー・サイトは 1 つのみです。

マスター表またはマスター・マテリアライズド・ビューに加えられた変更はマテリアライズド・ビュー・ログに記録され、リフレッシュ処理中にマテリアライズド・ビューに伝播されます。

注意： 高速リフレッシュ可能マテリアライズド・ビューは、マスター表とマスター・マテリアライズド・ビューのみに基づいて作成できます。シノニムまたはビューに基づいたマテリアライズド・ビューは、完全リフレッシュを行う必要があります。

マテリアライズド・ビュー・ログの内部トリガー

DML を使用してマスター表またはマスター・マテリアライズド・ビューに変更が加えられると、内部トリガーにより、影響を受けた行に関する情報がマテリアライズド・ビュー・ログに記録されます。この情報には、主キー、行 ID またはオブジェクト識別子（あるいはこれらすべて）の値のみでなく、マテリアライズド・ビュー・ログにロギングされる他の列の値も含まれます。これは、ターゲットのマスター表またはマスター・マテリアライズド・ビューのマテリアライズド・ビュー・ログを作成するときに自動的にアクティブになる AFTER ROW 内部トリガーです。このトリガーは、INSERT 文、UPDATE 文または DELETE 文により表のデータが変更されるたびに、マテリアライズド・ビュー・ログに行を挿入します。このトリガーは、常に最後に起動されるトリガーです。

注意： マテリアライズド・ビューに副問合せが含まれている場合は、副問合せで参照される列のログが必要になることがあります。副問合せマテリアライズド・ビューの詳細は、3-18 ページの「マテリアライズド・ビューを使用したデータのサブセット化」、ログが必要になる列の詳細は、6-16 ページの「マテリアライズド・ビュー・ログへの列のロギング」を参照してください。

マテリアライズド・ビュー・ログ

マスターに基づいてマテリアライズド・ビューを高速リフレッシュする場合は、マスター上にマテリアライズド・ビュー・ログが必要です。マスター表またはマスター・マテリアライズド・ビューにマテリアライズド・ビュー・ログを作成すると、基になる表がマテリアライズド・ビュー・ログとして作成されます。マテリアライズド・ビュー・ログには、マスター表またはマスター・マテリアライズド・ビューで更新された行の主キー、行 ID またはオブジェクト識別子（あるいはこれらすべて）を含められます。マテリアライズド・ビュー・ログには、副問合せを含むマテリアライズド・ビューの高速リフレッシュをサポートする他の列を含めることもできます。

マテリアライズド・ビュー・ログの表の名前は、MLOG$_*master_name* です。マテリアライズド・ビュー・ログは、ターゲット・マスターと同じスキーマ内に作成されます。1 つのマテリアライズド・ビュー・ログは、マスター表またはマスター・マテリアライズド・ビューに対する複数のマテリアライズド・ビューをサポートできます。前の項で説明したように、ターゲット・マスターに対して DML トランザクションが発生するたびに、内部トリガーによってマテリアライズド・ビュー・ログに変更情報が追加されます。

マテリアライズド・ビュー・ログには次のタイプがあります。

- **主キー** : マテリアライズド・ビューは、マスター表またはマスター・マテリアライズド・ビューが変更されると、影響を受けた行の主キーに基づいて変更内容を記録します。
- **ROWID**: マテリアライズド・ビューは、マスター表またはマスター・マテリアライズド・ビューが変更されると、影響を受けた行の行 ID に基づいて変更内容を記録します。
- **オブジェクト ID**: マテリアライズド・ビューは、マスター・オブジェクト表またはマスター・オブジェクト・マテリアライズド・ビューが変更されると、影響を受けた行オブジェクトのオブジェクト識別子に基づいて変更内容を記録します。
- **組合せ** : マテリアライズド・ビューは、マスター表またはマスター・マテリアライズド・ビューが変更されると、前述の 3 つのオプションの組合せに基づいて変更内容を記録します。影響を受けた行の主キー、`ROWID` およびオブジェクト識別子に基づいて変更内容が記録される可能性もあります。このようなマテリアライズド・ビュー・ログは、主キー、`ROWID` およびオブジェクトの各マテリアライズド・ビューをサポートします。これは、1 つのマスターに基づいた 3 種類のマテリアライズド・ビューがすべて含まれている環境に役立ちます。

組合せマテリアライズド・ビュー・ログは、1 種類のタイプの値を追跡するマテリアライズド・ビュー・ログと同じように機能しますが、複数のタイプの値が記録されるという点のみが異なります。たとえば、組合せマテリアライズド・ビュー・ログは、影響を受けた行の主キーと行 ID の両方を追跡および記録できます。

主キーに基づいたマテリアライズド・ビュー・ログと行 ID に基づいたマテリアライズド・ビュー・ログの間に大きな違いはありませんが（影響を受けた行を記録する際に、主キーと物理行 ID のどちらを使用するかという点のみが異なります）、実際に与える影響は非常に大きくなります。ROWID マテリアライズド・ビューとマテリアライズド・ビュー・ログを使用すると、マスター表の再編成および切捨てが難しくなります。これは、`ROWID` マテリアライズド・ビューを高速リフレッシュできなくなるためです。マスター表の再編成または切捨てを実行した場合はマスター表の行 ID が変更されるため、ROWID マテリアライズド・ビューは完全リフレッシュする必要があります。

注意：

- マスター表の再編成には、DBMS_MVIEW パッケージの BEGIN_TABLE_REORGANIZATION および END_TABLE_REORGANIZATION プロシージャを使用します。詳細は、『Oracle9*i* レプリケーション・マネージメント API リファレンス』を参照してください。
- マスター表を再編成するもう 1 つの方法としては、表のオンライン再定義がありますが、オンライン再定義は、マテリアライズド・ビュー・ログ、マスター・マテリアライズド・ビューおよびマテリアライズド・ビューを持つマスター表に対しては使用できません。オンライン再定義は、マテリアライズド・ビュー・ログを持たないマスター表に対して実行できます。表のオンライン再定義の詳細は、『Oracle9*i* データベース管理者ガイド』を参照してください。

オブジェクト表に基づいたマテリアライズド・ビュー・ログ オブジェクト表に対してマテリアライズド・ビュー・ログを作成できます。たとえば、次の SQL 文は categories_type ユーザー定義型を作成します。

```
CREATE TYPE category_typ AS OBJECT
   (category_name           VARCHAR2(50),
    category_description    VARCHAR2(1000),
    category_id             NUMBER(2))
NOT FINAL;
/
```

この型に基づいてオブジェクト表を作成するとき、オブジェクト識別子がシステム生成のものか主キーに基づいたものかを指定できます。

```
CREATE TABLE categories_tab_sys OF category_typ
    (category_id    PRIMARY KEY)
    OBJECT ID SYSTEM GENERATED;

CREATE TABLE categories_tab_pkbased OF category_typ
    (category_id    PRIMARY KEY)
    OBJECT ID PRIMARY KEY;
```

オブジェクト表に基づいてマテリアライズド・ビューを作成するときは、WITH OBJECT ID 句を指定してオブジェクト識別子をロギングする必要がありますが、オブジェクト識別子が主キー・ベースの場合は、主キーもロギングされるように指定できます。

たとえば、次の文では categories_tab_sys オブジェクト表のマテリアライズド・ビュー・ログを作成し、オブジェクト識別子列をロギングするように指定します。

```
CREATE MATERIALIZED VIEW LOG ON categories_tab_sys
   WITH OBJECT ID;
```

次の文では、categories_tab_pkbased オブジェクト表のマテリアライズド・ビュー・ログを作成し、オブジェクト識別子列とともに主キー列をロギングするように指定します。

```
CREATE MATERIALIZED VIEW LOG ON categories_tab_pkbased
   WITH OBJECT ID, PRIMARY KEY;
```

マテリアライズド・ビューおよびマテリアライズド・ビュー・ログの異なるスキーマへのインポートに関する制限事項 マテリアライズド・ビューおよびマテリアライズド・ビュー・ログは、DDL 文で明示的に指定されたスキーマ名でエクスポートされます。したがって、マテリアライズド・ビューおよびマテリアライズド・ビュー・ログをエクスポート元のスキーマと異なるスキーマにインポートすることはできません。FROM USER および TO USER インポート・オプションを使用してマテリアライズド・ビューまたはマテリアライズド・ビュー・ログを含むエクスポート・ダンプ・ファイルをインポートしようとすると、インポート・ログ・ファイルにエラーが書き込まれ、インポートは実行されません。

マテリアライズド・ビュー・サイトのメカニズム

マテリアライズド・ビューを作成すると、マテリアライズド・ビューをサポートするために、いくつかのメカニズムがマテリアライズド・ビュー・サイトで作成されます。具体的には、1 つの実表、1 つ以上の索引および場合によっては 1 つのビューが作成されます。更新可能なマテリアライズド・ビューを作成する場合は、マテリアライズド・ビュー・サイトに内部トリガーとローカル・ログ（更新可能なマテリアライズド・ビュー・ログ）も作成されます。

注意： マテリアライズド・ビュー・サイトがマスター・マテリアライズド・ビュー・サイトの場合は、この項で説明されているメカニズムに加えて、前項で説明されているメカニズムも含まれています。3-46 ページの「マスター・サイトおよびマスター・マテリアライズド・ビュー・サイトのメカニズム」を参照してください。

関連項目： 3-45 ページ図 3-9「マテリアライズド・ビューのレプリケーション・オブジェクト」

実表

マテリアライズド・ビューの実表の機能は、マテリアライズド・ビュー・データベースの互換性のレベルに依存します。互換性設定は、初期化パラメータ・ファイルの COMPATIBLE 初期化パラメータによって定義されます。

リリース 8.1 以上の互換性を持つ実表 互換性設定がリリース 8.1 以上の場合は、次の条件が適用されます。

- 実表が実際のマテリアライズド・ビューです（追加のビューは不要です）。
- マテリアライズド・ビュー名のサイズの上限は 30 バイトです。31 バイト以上の名前でマテリアライズド・ビューを作成しようとすると、エラーが戻されます。
- マテリアライズド・ビューには、マテリアライズド・ビューの作成時に指定した名前が付きます。

リリース 8.1 より低い互換性を持つ実表 互換性設定がリリース 8.1 より低い場合は、次の条件が適用されます。

- 実表はビューの基になるサポート・オブジェクトで、ビューがマテリアライズド・ビューです。
- マテリアライズド・ビューの作成時に生成される索引は、実表に作成されます。
- マテリアライズド・ビュー名のサイズの上限は 30 バイトです。31 バイト以上の名前でマテリアライズド・ビューを作成しようとすると、エラーが戻されます。
- マテリアライズド・ビューには、マテリアライズド・ビューの作成時に指定した名前が付きます。
- マテリアライズド・ビューの名前が 19 バイト以下の場合は、実表の名前は `SNAP$_` *materialized_view_name* になります。
- マテリアライズド・ビューの名前が 20 バイトから 30 バイトまでの場合は、実表の名前は 20 バイトに切り捨てられ、`SNAP$_` という接頭辞が付きます。数字の接尾辞が付く場合もあります。マテリアライズド・ビュー・サイトで、切り捨てられた名前と同じ名前の実表が他にない場合、切り捨てられた名前には何も追加されません。マテリアライズド・ビュー・サイトで、切り捨てられた名前と同じ名前の実表がある場合は、作成される実表には接尾辞が 1、2 というように最高 9999 まで付けられます。したがって、実表の名前は、`SNAP$_`、マテリアライズド・ビュー名の最初の 20 バイト、さらに何も付けられないか、あるいは 1 桁から 4 桁の数字になります。

  たとえば、`abcdefghijklmnopqrstuvwxyz` という名前のマテリアライズド・ビューに `SNAP$_abcdefghijklmnopqrst` という実表があり、これと同じ名前の実表が他にないとします。このマテリアライズド・ビュー・サイトでこの後作成される `abcdefghijklmnopqrstuvwxy` という名前のマテリアライズド・ビューの実表の名前は、`SNAP$_abcdefghijklmnopqrst1` になります。

注意： Oracle8 リリース 8.0 のデータベースの互換性設定は、Oracle8*i* リリース 8.1 より低くする必要があります。

関連項目： COMPATIBLE パラメータの詳細は、6-4 ページの「初期化パラメータ」および『Oracle9*i* データベース移行ガイド』を参照してください。8.1.0 より前のバージョンと互換性を持つマテリアライズド・ビューをサポートして作成されるビューの詳細は、3-52 ページの「ビュー」を参照してください。

ビュー

ビューが作成されるのは、Oracle8 リリース 8.0 以下のデータベースとのマテリアライズド・ビューのレプリケーションをサポートする必要がある場合、あるいは Oracle8*i* リリース 8.1 以上のマテリアライズド・ビュー・サイトの互換性設定が Oracle8*i* リリース 8.1 より低くなっている場合のみです。ビューが作成される場合、ビューの名前は CREATE MATERIALIZED VIEW 文で指定された名前と同じになります。たとえば、CREATE MATERIALIZED VIEW sales.mview_customer AS .... 文は、mview_customer という名前のビューを作成します。

索引

リモート・マテリアライズド・ビュー・サイトでは、各主キー・マテリアライズド・ビューに少なくとも 1 つの索引が作成されます。この索引は、ターゲット・マスター表またはマスター・マテリアライズド・ビューの主キーと対応しており、名前は I_SNAP$_*materialized view_name* になります。副問合せを使用するマテリアライズド・ビューの高速リフレッシュをサポートするために、リモート・マテリアライズド・ビュー・サイトに追加の索引が作成されることもあります。

注意： I_SNAP$_*materialized view_name* が 32 文字を超える場合、表名が切り捨てられ、順序番号が追加されます。

更新可能マテリアライズド・ビュー・ログ

更新可能マテリアライズド・ビュー・ログ（USLOG$_*materialized_view_name*）は、高速リフレッシュ中にマテリアライズド・ビューで上書きまたは削除する行を判断するために使用されます。このログは読取り専用マテリアライズド・ビューでは作成されず、また完全リフレッシュ中にはこのログは使用されません。完全リフレッシュの場合はマテリアライズド・ビュー全体が置き換えられるためです。

更新可能マテリアライズド・ビューとマスターとの間に競合がある場合は、マスター・サイトまたはマスター・マテリアライズド・ビュー・サイトのマテリアライズド・ビュー・ログには存在しないエントリが、リフレッシュ中に更新可能マテリアライズド・ビュー・ログに残ることがあります。この場合は、更新可能マテリアライズド・ビュー・ログを使用してマテリアライズド・ビューの行が削除または上書きされます。

更新可能マテリアライズド・ビュー・ログは、次のように、書込み可能マテリアライズド・ビューの高速リフレッシュにも使用されます。

1. ユーザーが、リモート・マスターを持つ書込み可能マテリアライズド・ビューに行を挿入します。このビューは書込み可能であり更新可能ではないため、このトランザクションはマテリアライズド・ビュー・サイトの遅延トランザクション・キューには格納されません。
2. この挿入に関する情報が、更新可能マテリアライズド・ビュー・ログにロギングされます。
3. ユーザーがマテリアライズド・ビューを高速リフレッシュします。
4. 更新可能マテリアライズド・ビュー・ログの情報を使用して、挿入された行が削除されます。高速リフレッシュの完了後、マテリアライズド・ビューはマスターと同一である必要があります。このため、挿入された行を削除する必要があります。

更新可能マテリアライズド・ビュー・ログの内部トリガー

マスター・サイトまたはマスター・マテリアライズド・ビュー・サイトでの内部トリガーと同じように、マテリアライズド・ビュー・サイトの内部トリガーも更新可能マテリアライズド・ビューに適用される DML による変更を `USLOG$_`*materialized_view_name* ログに記録します。読取り専用マテリアライズド・ビューではこのトリガーは作成されません。

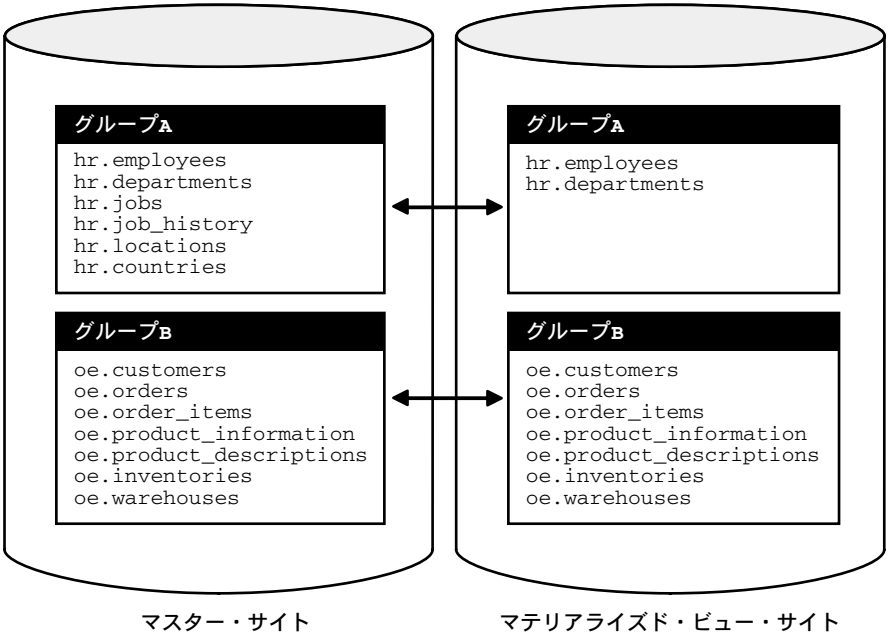
編成メカニズム

前項で説明されているメカニズムに加えて、マテリアライズド・ビュー・サイトのマテリアライズド・ビューを編成するメカニズムが他にもいくつかあります。このメカニズムによって、マテリアライズド・ビュー・サイトとマスター・サイトまたはマスター・マテリアライズド・ビュー・サイト間の編成上の整合性、およびターゲット・レプリケーション・グループとのトランザクション（読込み）一貫性が保たれます。これらのメカニズムは、マテリアライズド・ビュー・グループとリフレッシュ・グループです。

マテリアライズド・ビュー・グループ

レプリケーション・システム内の**マテリアライズド・ビュー・グループ**は、ターゲット・レプリケーション・グループにあるオブジェクトの一部または完全なコピーをマスター・サイトまたはマスター・マテリアライズド・ビュー・サイトでメンテナンスします。マテリアライズド・ビュー・グループは、マスター・サイトまたはマスター・マテリアライズド・ビュー・サイトのレプリケーション・グループの境界を超えられません。図 3-11 は、マスター・サイトのグループ A およびグループ B と、マテリアライズド・ビュー・サイトのグループ A およびグループ B との相関関係を示します。

図 3-11 マテリアライズド・ビュー・グループとマスター・グループの対応



マテリアライズド・ビュー・サイトのグループ A（図 3-11 を参照）には、マスター・サイトの対応するグループ A にあるオブジェクトの一部しか含まれていません。マテリアライズド・ビュー・サイトのグループ B には、マスター・サイトのグループ B のオブジェクトがすべて含まれています。しかし、マテリアライズド・ビュー・サイトのグループ B にマスター・サイトのグループ A のオブジェクトが含まれることはありません。図 3-11 に示されているように、マテリアライズド・ビュー・グループの名前は、その基になるマスター・グループと同じ名前になります。たとえば、`personnel` マスター・グループに基づいたマテリアライズド・ビュー・グループは、`personnel` という名前です。

マテリアライズド・ビュー・グループは、マテリアライズド・ビュー・サイトとマスター・サイトまたはマスター・マテリアライズド・ビュー・サイト間の編成上の整合性を維持するという目的のみでなく、更新可能マテリアライズド・ビューのサポートでも必要になります。マテリアライズド・ビューがマテリアライズド・ビュー・グループに属していない場合は、読取り専用または書込み可能マテリアライズド・ビューです。

マテリアライズド・ビューのグループ所有者 マテリアライズド・ビューのグループ所有者により、マスター・サイトまたはマスター・マテリアライズド・ビュー・サイトの単一のレプリケーション・グループに基づいた複数のマテリアライズド・ビュー・グループが使用可能になります。たとえば、マテリアライズド・ビュー・サイトと同じデータベース内で複数のユーザーをサポートする必要がある場合は、1 つのターゲット・マスター・グループに複数のマテリアライズド・ビュー・グループを作成できます。複数のマテリアライズド・ビュー・グループを作成すると、各マテリアライズド・ビュー・グループのマテリアライズド・ビュー定義に異なる副問合せを定義でき、各ユーザーは自分用のデータのサブセットのみにアクセスできます。

複数のマテリアライズド・ビュー・グループを定義すると、グループ・レベルでデータ・セットを制御できます。たとえば、`hr`、`personnel` および `manufacturing` という名前の異なるマテリアライズド・ビュー・グループをそれぞれの部門用に作成した場合、各部門を個々のオブジェクトではなくグループとして管理でき、複数のオブジェクトをグループとして削除できます。

マスター・サイトまたはマスター・マテリアライズド・ビュー・サイトの単一のレプリケーション・グループに基づいた複数のマテリアライズド・ビュー・グループを同一マテリアライズド・ビュー・サイトで使用できるようにするには、マテリアライズド・ビュー・グループを定義するときにグループ所有者を追加識別子として指定します。

グループ所有者を指定してマテリアライズド・ビュー・グループを定義した後、同じグループ所有者を定義してマテリアライズド・ビュー・オブジェクトをターゲット・マテリアライズド・ビュー・グループに追加します。グループ所有者を使用する場合は、各マテリアライズド・ビュー・オブジェクトが一意の名前を持つ必要があります。1 つのマテリアライズド・ビュー・サイトに、マスター・サイトまたはマスター・マテリアライズド・ビュー・サイトの同一レプリケーション・グループに基づいたマテリアライズド・ビュー・グループが複数ある場合は、マテリアライズド・ビュー・グループのオブジェクト名に、別のマテリアライズド・ビュー・グループのマテリアライズド・ビュー・オブジェクトと同じ名前を使用することはできません。名前の競合を避けるには、オブジェクト名の最後にグループ所有者を追加します。たとえば、`hr` と `ac` というグループ所有者がいる場合は、`employees` マテリアライズド・ビュー・オブジェクトには、それぞれ `employees_hr` および `employees_ac` という名前を付けます。

また、1 つのマテリアライズド・ビュー・サイトの同じレプリケーション・グループに基づいたマテリアライズド・ビュー・グループはすべて、同じマスター・サイトまたはマスター・マテリアライズド・ビュー・サイトを指す必要があります。たとえば、`hr` と `personnel` により所有されるグループが同一マテリアライズド・ビュー・サイトにあると想定して、`hr` により所有されている `hr_repg` マテリアライズド・ビュー・グループが `orc1.world` マスター・サイトの対応するマスター・グループに基づいている場合、`personnel` により所有されている `hr_repg` マテリアライズド・ビュー・グループも、`orc1.world` の対応するマスター・グループに基づいている必要があります。

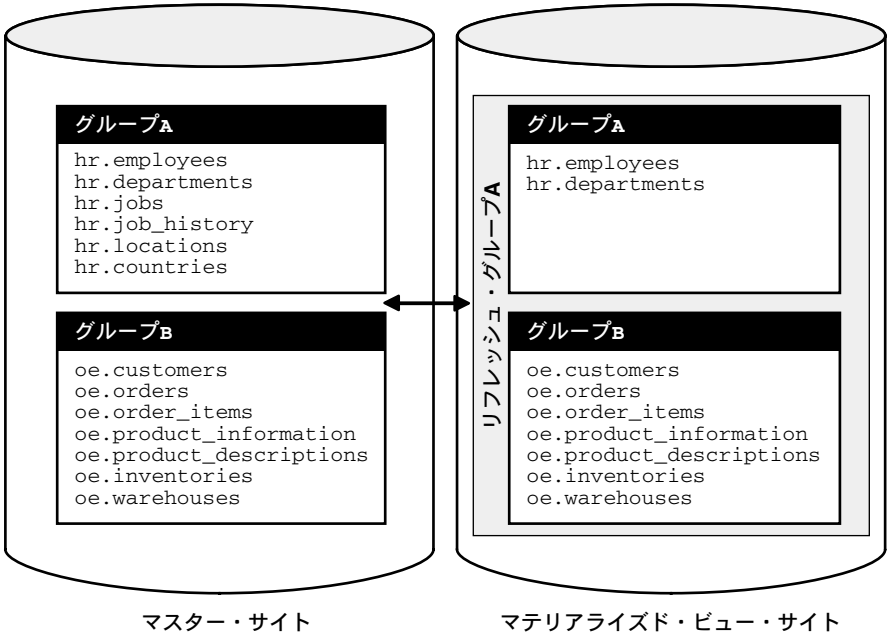
> **関連項目：** レプリケーション・マネージメント API を使用したグループ所有者の定義方法の詳細は、『Oracle9*i* レプリケーション・マネージメント API リファレンス』を参照してください。

リフレッシュ・グループ

複数のマテリアライズド・ビュー間の参照整合性およびトランザクション（読込み）一貫性を保つために、Oracle では、個々のマテリアライズド・ビューをリフレッシュ・グループの一部としてリフレッシュできます。リフレッシュ・グループ内のすべてのマテリアライズド・ビューのリフレッシュが終了すると、グループ内のすべてのマテリアライズド・ビューのデータはトランザクションの一貫性を保つ特定の時点のデータに対応します。

図 3-12 に示すように、リフレッシュ・グループには複数のマテリアライズド・ビュー・グループのマテリアライズド・ビューを含めることができるため、レプリケーション・グループ境界にまたがってトランザクション（読込み）一貫性を保つことができます。

図 3-12 複数のマテリアライズド・ビュー・グループのオブジェクトを含むリフレッシュグループ



マテリアライズド・ビュー・グループごとに 1 つのリフレッシュ・グループを定義するよりも、複数のマテリアライズド・ビュー・グループのオブジェクトを含む 1 つの大きなリフレッシュ・グループを使用するほうが効率的な場合があります。このように構成すると、マテリアライズド・ビューのリフレッシュに必要なオーバーヘッドの量が削減されます。リフレッシュ・グループには、マテリアライズド・ビューを最大 400 個まで含められます。

1 つのマテリアライズド・ビュー・グループの内容をリフレッシュするために複数のリフレッシュ・グループを使用する構成は避けてください。複数のリフレッシュ・グループを使用して 1 つのマテリアライズド・ビュー・グループの内容をリフレッシュすると、マテリアライズド・ビュー・データの整合性が失われる原因となり、マテリアライズド・ビュー・サイトで参照整合性の問題が発生することがあります。このような構成は、データベース環境について十分な知識があり、参照整合性の問題の発生を回避できる場合にのみ使用してください。

リフレッシュ・グループのサイズ

リフレッシュ・グループのサイズを決定するときは、サイズの違いによるメリットとデメリットの両方を考慮する必要があります。Oracle は、大規模リフレッシュ・グループに対応できるように最適化されています。したがって、大規模リフレッシュ・グループと小規模リフレッシュ・グループを比較した場合、グループ内のマテリアライズド・ビューが類似していると想定すると、大規模リフレッシュ・グループのほうが同じ数のマテリアライズド・ビューを高速にリフレッシュできます。たとえば、マテリアライズド・ビューを 20 ずつ含む 5 つのリフレッシュ・グループよりも、100 のマテリアライズド・ビューを含む 1 つのリフレッシュ・グループのほうが高速にリフレッシュできます。また、大規模リフレッシュ・グループを使用すると、1 回のレプリケーション・マネージメント API コールで、より多くのマテリアライズド・ビューをリフレッシュできます。

リフレッシュ・グループのリフレッシュ時に、グループ内の各マテリアライズド・ビューは、マテリアライズド・ビュー・サイトでグループ内のすべてのマテリアライズド・ビューのリフレッシュが終了するまでロックされます。リフレッシュ操作中にユーザーがマテリアライズド・ビューを更新するとデータの整合性が損なわれる可能性があるので、これを回避するためにマテリアライズド・ビューのロックは必須です。したがって、リフレッシュ・グループの規模が小さいということは、リフレッシュ実行時のマテリアライズド・ビューのロック時間が短くなることを意味します。

リフレッシュの実行時は、ネットワーク接続を維持する必要があります。リフレッシュ実行時にネットワーク接続が切断または妨害されると、データベースの整合性を保つためにすべての変更がロールバックされます。したがって、ネットワーク接続を維持することが困難な場合は、リフレッシュ・グループの規模を小さくします。

Advanced Replication では、NULL リフレッシュの最適化が行われます。つまり、特定のマテリアライズド・ビューの最後のリフレッシュ以降にマスター表またはマスター・マテリアライズド・ビューが変更されなかった場合、このマテリアライズド・ビューに対してマテリアライズド・ビュー・グループのリフレッシュ時に余分な時間はほとんどかかりません。ただし、リリース 8.1 より前のデータベースにあるマテリアライズド・ビューでは、更新頻度が少ないマスター表のマテリアライズド・ビューを、別の独立したリフレッシュ・グループに分割することを検討してください。それによって、更新頻度が高いマスター表のマテリアライズド・ビューを含む他のマテリアライズド・ビュー・グループのリフレッシュ時間を短縮できます。

表 3-3 に、大規模リフレッシュ・グループと小規模リフレッシュグループの利点をまとめます。

表 3-3 大規模リフレッシュ・グループおよび小規模リフレッシュ・グループ

大規模リフレッシュ・グループの利点	小規模リフレッシュ・グループの利点
■ 同数のマテリアライズド・ビューを複数のリフレッシュ・グループに分けてリフレッシュするよりも高速にリフレッシュできます。	■ マテリアライズド・ビューのロック時間が短くなります。
■ 1 回のレプリケーション・マネージメント API コールでリフレッシュできます。	■ ネットワーク接続が失われたことが原因でリフレッシュ実行時に変更がロールバックされる可能性が低くなります。

リフレッシュ処理

マテリアライズド・ビューのデータは、そのマスター表またはマスター・マテリアライズド・ビューの現行のデータと常に一致しているとはかぎりません。マテリアライズド・ビューは、ある一時点（作成時またはリフレッシュ時）に存在していたデータとして、トランザクション（読込み）一貫性を保ってマスターを反映したものです。マテリアライズド・ビューのデータをマスターのデータとほぼ一致させるには、マテリアライズド・ビューを定期的にリフレッシュする必要があります。**マテリアライズド・ビュー・リフレッシュ**は、より新しい状態のマスター表またはマスター・マテリアライズド・ビューをマテリアライズド・ビューに反映させる効果的なバッチ操作です。

更新可能マテリアライズド・ビューのリフレッシュでは、まず、マテリアライズド・ビュー・サイトの遅延トランザクションがマスター・サイトまたはマスター・マテリアライズド・ビュー・サイトにプッシュされます。次に、マスター・サイトまたはマスター・マテリアライズド・ビュー・サイトのデータがプルされ、マテリアライズド・ビューに適用されます。

マスター表の 1 行は、マテリアライズド・ビューのリフレッシュとリフレッシュの間に何度も更新される場合がありますが、リフレッシュでは、その行はマテリアライズド・ビュー内で現行のデータを使用して 1 回のみ更新されます。たとえば、マテリアライズド・ビューの前回のリフレッシュ以降に、マスター表の 1 行が 10 回更新されても、次回のリフレッシュではマテリアライズド・ビュー内の対応する行は 1 度のみ更新されます。

より新しいマスター・データを反映するために、各マテリアライズド・ビューのリフレッシュ間隔およびリフレッシュ方法を決定します。たとえば、アプリケーションで頻繁に更新されるマスター表に基づいたマテリアライズド・ビューは、頻繁にリフレッシュする必要があります。これに対して、比較的静的なマスターに基づいたマテリアライズド・ビューは、あまり頻繁にリフレッシュする必要はありません。つまり、アプリケーションの特性および要件を分析して、マテリアライズド・ビューの適切なリフレッシュ間隔を判断します。

マテリアライズド・ビューのリフレッシュには、いくつかのリフレッシュ・タイプとリフレッシュ開始方法がサポートされています。

リフレッシュ・タイプ

マテリアライズド・ビューをリフレッシュするには、高速リフレッシュ、完全リフレッシュまたは強制リフレッシュのいずれかを使用できます。

完全リフレッシュ マテリアライズド・ビューの**完全リフレッシュ**を実行するには、マテリアライズド・ビューを管理するサーバーがマテリアライズド・ビューの定義問合せを実行します。これは基本的に、マテリアライズド・ビューを再作成する処理です。マテリアライズド・ビューをリフレッシュするために、問合せの結果セットが現在のマテリアライズド・ビュー・データを置き換えます。完全リフレッシュはあらゆるマテリアライズド・ビューで実行できます。定義問合せを満たすデータの量によっては、完全リフレッシュの実行に高速リフレッシュの場合よりもかなり長い時間がかかる場合があります。

マスター・マテリアライズド・ビューの完全リフレッシュを実行した場合、それ以降にこのマスター・マテリアライズド・ビューに基づいたマテリアライズド・ビューに対して実行するリフレッシュは、完全リフレッシュにする必要があります。マスター・マテリアライズド・ビューが完全リフレッシュを実行した後でこのようなマテリアライズド・ビューに高速リフレッシュを実行しようとすると、次のエラーが戻されます。

```
ORA-12034 マテリアライズド・ビュー・ログは最新のリフレッシュよりも新しいものです。
```

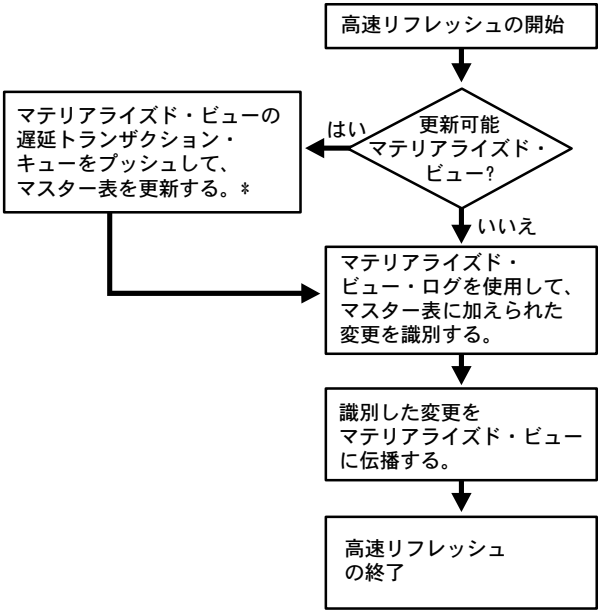
注意： マテリアライズド・ビューが完全リフレッシュされる場合は、効率を最大化するために、PCTFREE を 0 に、PCTUSED を 99 に設定します。

高速リフレッシュ **高速リフレッシュ**を実行するには、マテリアライズド・ビューを管理しているマスターが、マテリアライズド・ビューの最新リフレッシュ後にマスターに加えられた変更を識別し、その変更内容をマテリアライズド・ビューに適用します。高速リフレッシュは、関係する各サーバーおよびネットワークでレプリケートするデータが少なくて済むため、マスターに加えられた変更が少ないときは完全リフレッシュよりも効率的です。

マテリアライズド・ビューの高速リフレッシュは、マスター表またはマスター・マテリアライズド・ビューにマテリアライズド・ビュー・ログがある場合にのみ可能です。また、高速リフレッシュを完全リフレッシュより高速にするには、CREATE MATERIALIZED VIEW 文の各結合列に対して索引が必要です。

SQL*Loader を使用してマスター表またはマスター・マテリアライズド・ビューでダイレクト・パス・ロードを行うと、それ以後は高速リフレッシュを実行してもダイレクト・パス・ロード中に発生した変更は適用されません。また、マスターに対してその他のバルク・ロード操作を行った結果加えられた変更も適用されません。このような操作の例としては、APPEND ヒントを指定した INSERT 文と、いくつかの INSERT ...SELECT * FROM 文があります。

図 3-13 マテリアライズド・ビューの高速リフレッシュ



* このステップは、**DBMS_DEFER_SYS.PUSH**関数を使用して別に実行することもできます。

更新可能な多重化マテリアライズド・ビューがある場合、多重化マテリアライズド・ビューに対して加えられた DML 変更はこのマテリアライズド・ビューに複数回取り出され、マテリアライズド・ビューがリフレッシュされるたびにデータ整合性が保たれます。次に、この動作をわかりやすく説明します。次のような特性を持つレプリケーション環境の例について考えます。

- マスター・サイト `orc1.world` に `oe.customers` 表があります。
- レベル 1 のマテリアライズド・ビュー・サイト `ca.us` に、`orc1.world` の `oe.customers` 表に基づいた `oe.customers_region` 更新可能マテリアライズド・ビューがあります。
- レベル 2 の更新可能マテリアライズド・ビュー・サイト `sf.ca` に、`ca.us` の `oe.customers_region` マテリアライズド・ビューに基づいた `oe.customers_sf` 更新可能マテリアライズド・ビューがあります。

前述の内容を前提として、次のような使用例が考えられます。

1. sf.ca の oe.customers_sf 更新可能マテリアライズド・ビューで、顧客の credit_limit が 3000 から 5000 に変更されました。
2. この変更が、sf.ca の遅延トランザクション・キューに入れられます。
3. レベル 2 のマテリアライズド・ビュー oe.customers_sf の高速リフレッシュにより、credit_limit の新しい値が ca.us の oe.customers_region マテリアライズド・ビューにプッシュされます。
4. この変更が、ca.us の oe.customers_region マテリアライズド・ビューに適用されます。
5. ca.us サイトでの credit_limit に対する更新が、レベル 1 のマテリアライズド・ビュー・サイトの遅延トランザクション・キューとマテリアライズド・ビュー・ログの両方に記録されます。
6. レベル 2 のマテリアライズド・ビュー oe.customers_sf の高速リフレッシュにより、この sf.ca のマテリアライズド・ビューに credit_limit の 5000 という値がプルされます。
7. レベル 1 のマテリアライズド・ビュー oe.customers_region の高速リフレッシュにより、credit_limit の新しい値が orc1.world の oe.customers マスター表にプッシュされます。
8. この変更が、orc1.world の oe.customers マスター表に適用されます。
9. orc1.world サイトでの credit_limit に対する更新が、このマスター・サイトの遅延トランザクション・キューとマテリアライズド・ビュー・ログの両方に記録されます。
10. レベル 1 のマテリアライズド・ビュー oe.customers_region が新しく高速リフレッシュされ、ca.us のこのマテリアライズド・ビューに credit_limit の 5000 という値がプルされます。
11. ca.us サイトでの credit_limit に対する更新が、このレベル 1 のマテリアライズド・ビュー・サイトのマテリアライズド・ビュー・ログに記録されます。
12. レベル 2 のマテリアライズド・ビュー oe.customers_sf の高速リフレッシュにより、sf.ca のこのマテリアライズド・ビューに credit_limit の 5000 という値が再度プルされます。

強制リフレッシュ マテリアライズド・ビューの**強制リフレッシュ**を実行するために、マテリアライズド・ビューを管理するサーバーは高速リフレッシュの実行を試みます。高速リフレッシュが実行できない場合は、完全リフレッシュが実行されます。高速リフレッシュが実行できない場合にマテリアライズド・ビューのリフレッシュを実行するには、強制リフレッシュの設定を使用します。

リフレッシュの開始

管理者は、リフレッシュ・グループを作成するときに、スケジュールした間隔でマテリアライズド・ビューを自動的にリフレッシュするようにグループを構成できます。逆に、リフレッシュ・グループを手動または必要に応じてリフレッシュするように、スケジュール情報を省略することもできます。ダイアルアップ・ネットワーク接続を使用してリフレッシュを実行する場合は、手動リフレッシュが理想的なソリューションになります。

スケジュールしたリフレッシュ 自動リフレッシュのためのリフレッシュ・グループを作成するには、作成プロセスで、スケジュールしたリフレッシュ間隔をグループに指定する必要があります。グループのリフレッシュ間隔を設定するときは、次の特性を考慮してください。

- リフレッシュ間隔を指定する日付または日付式の値は、将来の特定の時点である必要があります。
- リフレッシュ間隔は、リフレッシュの実行に必要な時間よりも長くする必要があります。
- 相対日付式の値は、最新のリフレッシュ日付から計算される特定の時点である必要があります。ネットワークまたはシステムの障害により、グループのリフレッシュがスケジュールどおりに実行されない場合、相対日付式の結果は状況に応じて変更されることがあります。
- 明示的日付式の値は、最新のリフレッシュ日付に関係なく特定の時点である必要があります。
- 古いデータに対する環境の許容度を考慮してください。許容度が低い場合はリフレッシュ間隔を小さくし、許容度が高い場合はリフレッシュ間隔を大きくします。

リフレッシュ間隔の指定に使用できる単純な日付式の例を、次に示します。

- 1 時間間隔の指定 :

  ```
  SYSDATE + 1/24
  ```

- 7 日間隔の指定 :

  ```
  SYSDATE + 7
  ```

> **関連項目：** 日付算術の詳細は、『Oracle9*i* データベース管理者ガイド』および『Oracle9*i* SQL リファレンス』を参照してください。

必要に応じたリフレッシュ 環境や状況によっては、スケジュールしたマテリアライズド・ビュー・リフレッシュの実行が適切でない場合もあります。たとえば、マスター表にバルク・データをロードした直後は、依存マテリアライズド・ビューにマスター表のデータは反映されていません。次に実行予定の自動グループ・リフレッシュを待たずに、対応付けられたマテリアライズド・ビューにマスター表の新規行をすぐに伝播するために、依存マテリアライズド・ビュー・グループを手動でリフレッシュすることもできます。

マテリアライズド・ビューが非接続環境のラップトップの営業支援システムと統合されている場合は、マテリアライズド・ビューを必要に応じてリフレッシュすることもできます。営業支援ソフトウェアを設計する開発者がアプリケーション・コントロール（ボタンなど）を作成します。営業担当員は、1 日の受注をサーバーに転送するときに、ダイアルアップ・ネットワーク接続を確立し、アプリケーション・コントロールを使用してマテリアライズド・ビューをリフレッシュできます。

`hr_refg` リフレッシュ・グループを必要に応じてリフレッシュする例を次に示します。

```
EXECUTE DBMS_REFRESH.REFRESH('hr_refg');
```

4

配置テンプレートの概要およびアーキテクチャ

この章では、配置テンプレートの概要と、配置テンプレートを使用して簡単かつ効率的にマテリアライズド・ビュー・サイトを配布する方法について説明します。

この章の内容は、次のとおりです。

- 大量配布の課題
- Oracle 配置テンプレートの概念
- 配置テンプレートのアーキテクチャ
- 配置テンプレートの設計
- マテリアライズド・ビュー作成のローカル制御

注意： 配置テンプレートを作成する前に、第 3 章「マテリアライズド・ビューの概要とアーキテクチャ」を読んでください。マテリアライズド・ビューについての理解は、配置テンプレートを作成する際に役立ちます。

大量配布の課題

Oracle 配置テンプレートは、広範囲に分散したマテリアライズド・ビュー環境を効率的に配置し、管理するためのツールです。配置テンプレートの概念、アーキテクチャおよび使用方法について説明する前に、大量配布の環境の課題について検討します。

時と場所を問わず正確な情報を入手する必要性は、ますます強くなっています。また、情報は分散し、ユーザーもネットワークに接続されていない環境にいることが多く、実際に利用する場所に情報を配布する必要があります。

たとえば、モバイル端末を使用する営業担当員の場合を考えます。正確な顧客情報をラップトップ上で自由自在に参照したいと考えている営業担当員の数は、数百名あるいは数千名にも及ぶ可能性があります。これに応えるため、DBA は、データおよびデータベース・インフラストラクチャ（表、索引、制約、トリガーなど）を適切なタイミングで効率よくすべてのサイトに提供する必要があります。

これまで DBA は、独自の配置方法を開発する必要がありました。一般に、DBA は、リモート・マテリアライズド・ビュー・サイトにマテリアライズド・ビュー環境を作成するための非常に複雑なスクリプトを作成する必要がありました。さらに、それに加えて、DBA はデータ・セットをマテリアライズド・ビュー・サイトでカスタマイズすることが必要な場合もありました。スクリプトを作成した後も、スクリプトを配置するために手動でパッケージ化し、実装する必要があり、多くの場合その両方に多大なトラブルシューティングが必要でした。

このような問題点をもとに、効率的な大量配布を行うための新しいテクノロジとリソースが生まれました。大量配布とは、データベース・インフラストラクチャ、データおよびフロントエンド・アプリケーションを多数のユーザーに配布するプロセスのことです。Advanced Replication では、大量配布という用語はデータおよびデータ・インフラストラクチャの配布のみを指します。

配置テンプレートおよび大量配布の目標

大量配布のツールおよびテクノロジは、データおよびデータベース・インフラストラクチャを配布する DBA を支援する必要があります。配置テンプレートの目標は、一度定義した環境をもとに必要な数の配置テンプレートのインスタンスを作成し、個々のサイトでカスタマイズを行えるようにすることです。

この目標を実現するために、Oracle の配置テンプレートでは次のことをサポートしています。

1 回のマテリアライズド・ビュー環境の定義	配置テンプレートを使用してマテリアライズド・ビュー環境の構造を 1 回定義することにより、各ユーザー（サイト）がフロントエンド・アプリケーションをサポートするデータベース・インフラストラクチャを利用できるようにします。
個別のマテリアライズド・ビュー・サイトのカスタマイズ	配置テンプレートのパラメータを使用して各マテリアライズド・ビュー環境をカスタマイズし、各ユーザーが必要な特定のデータ・サブセットを受信できるようにします。

大量配布には、モバイル端末を使用する営業担当員やサービス技術員、小売店およびリモートの在庫保管サイトなどへの情報の配布といった様々な用途があります。このような環境では、配置テンプレートを使用してリモート・サイトにデータベース・インフラストラクチャを作成します。これは主に、配置テンプレートがデータのサブセット化、非接続環境でのレプリケーション、および少ないリソース要件をサポートしているからです（これはラップトップ・ユーザーに最適です）。

Oracle 配置テンプレートの概念

DBA は、Oracle から提供されている配置テンプレートを使用して、マテリアライズド・ビュー環境を**パッケージ化**し、安全性の高いカスタム配置を簡単に実行できます。配置テンプレートのパッケージ化とは、配置テンプレートにより作成されるマテリアライズド・ビュー環境を定義するプロセスです。配置テンプレートのパッケージ化により、テンプレートをリモート・マテリアライズド・ビュー・サイトで**インスタンス化**する準備が整います。インスタンス化によって、マテリアライズド・ビュー・サイトのオブジェクトが作成され、マテリアライズド・ビューにデータが移入されます。

配置テンプレートには、固定データ・セットを使用する 1 つのマテリアライズド・ビューのみを含む単純なものも、1 つ以上の変数に基づく動的データ・セットを使用する数百のマテリアライズド・ビューを含む複雑なものもあります。配置テンプレートには、次の機能があります。

- 集中制御
- マテリアライズド・ビュー環境を繰り返し配置する機能
- リモート・サイトでのデータのサブセット化およびカスタマイズを可能にする、テンプレート・パラメータ
- テンプレートのインスタンス化およびデータ・アクセスを制御する、認証ユーザー・リスト

マテリアライズド・ビュー環境を配置するには、その前にマスター・サイトに配置テンプレートを作成します。このテンプレートには、リモート・サイトのオブジェクトやターゲット・リフレッシュ・グループを作成するデータ定義言語（DDL）など、マテリアライズド・ビュー環境を配置するために必要なすべての情報を格納します。また、このテンプレートには、ユーザーのセキュリティ情報へのリンクや、カスタム・マテリアライズド・ビュー作成のためのテンプレート・パラメータも格納します。

配置テンプレートの要素

各配置テンプレートには、マテリアライズド・ビュー・サイトに必要なマテリアライズド・ビューおよびそれに関連するオブジェクトを作成するための設計図を含めます。具体的には、配置テンプレートをマスター・サイトに作成し、マテリアライズド・ビュー環境を作成するために必要なマテリアライズド・ビュー、トリガー、ビューなどをこの配置テンプレートに追加します。また、オプションでテンプレート・パラメータや認証ユーザーを定義して、インスタンス化プロセス実行時の柔軟性や安全性を大幅に向上させることもできます。

配置テンプレートの要素は、次の 4 つのカテゴリに分類できます。

- テンプレート一般情報
- テンプレートのオブジェクト定義
- テンプレート・パラメータ
- ユーザー認証

テンプレート一般情報

テンプレート名、ターゲット・リフレッシュ・グループ、およびプライベートまたはパブリックの状態から構成されるテンプレート一般情報は、Oracle 配置テンプレートの基礎になる情報です。図 4-1 に示すように、`REFRESH_TEMPLATE_NAME` は、配置テンプレートのデータ・ディクショナリ・ビューのどの場面でも使用します。DBA は、マテリアライズド・ビュー環境のオブジェクトをテンプレートに追加し、その後、指定されたテンプレートの識別に応じて配布するテンプレートをリリースします（図 4-2 を参照してください）。

配置テンプレートは、1 つのマスター・サイトで定義します。マスター・サイトに同じ名前の配置テンプレートを 2 つ定義することはできませんが、配置テンプレートを同じ名前で他のサイトにコピーすることはできます。

図 4-1 配置テンプレートのビューの関係

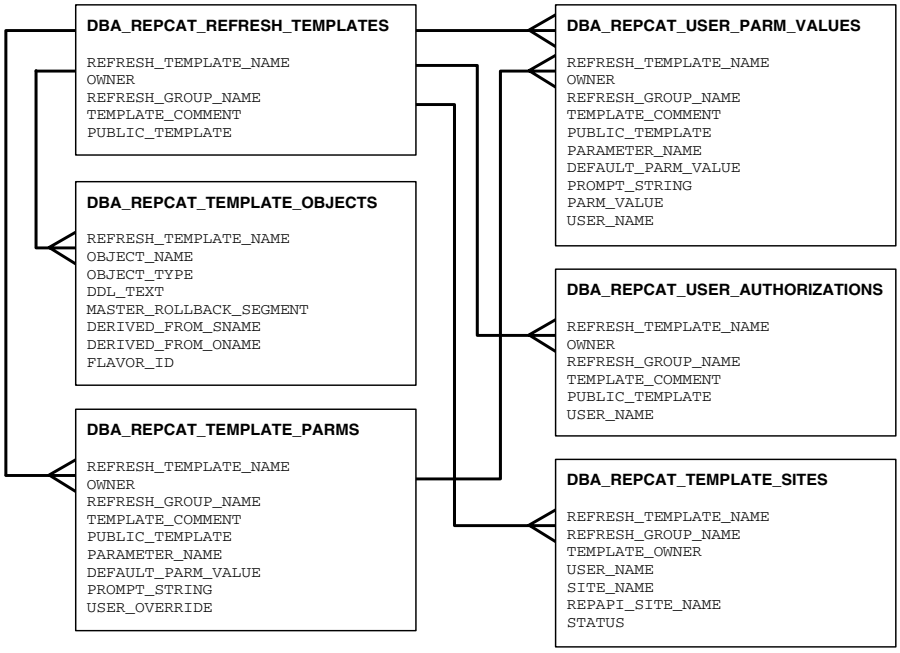
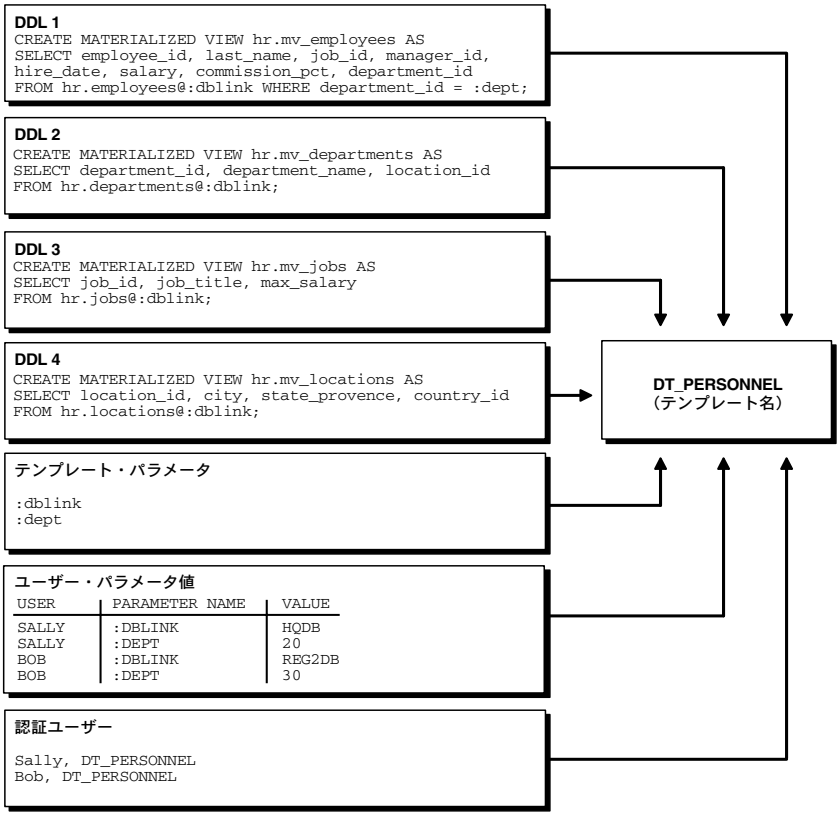


図 4-2 配置テンプレートに追加するテンプレートの要素



テンプレートのオブジェクト定義

配置テンプレートを定義した後で、テンプレートにオブジェクトを追加します。テンプレートがマテリアライズド・ビュー・サイトでインスタンス化されると、オブジェクト DDL（CREATE MATERIALIZED VIEW、CREATE TABLE など）が実行され、適切なオブジェクトがマテリアライズド・ビュー・サイトに作成されます。

配置テンプレートには既存のマスター・オブジェクトに基づいたオブジェクトを追加できますが、必要な場合は、DDL を定義し、新しいテンプレート・オブジェクトを作成してオブジェクトを作成することもできます。新しいオブジェクト DDL を作成した場合は、実行エラーを防ぐために DDL が字句規則に適合しているかどうかのチェックが行われます。配置テンプレートに追加する更新可能マテリアライズド・ビューはマスター・グループ内の表に基づいている必要がありますが、読取り専用マテリアライズド・ビューなどのその他のオブ

ジェクトは、マスター・グループに属していないオブジェクトに基づいていてもかまいません。

通常、テンプレートにはマテリアライズド・ビューを追加しますが、必要な場合はその他のオブジェクトも追加できます。たとえば、マテリアライズド・ビュー・サイトでのデータ整合性を規定するための制約、データ表示のためのビュー、あるいはローカルなデータ記憶域に格納するための表などを追加できます。場合によっては、配置テンプレートにアプリケーションのすべてのオブジェクトを含めることもできます。配置テンプレートを使用して作成したマテリアライズド・ビューは、テンプレートに定義されたリフレッシュ・グループに自動的に追加されます。

配置テンプレートを使用してインスタンス化できないオブジェクト型は、次のとおりです。

- ユーザー定義型
- ユーザー定義型の本体
- ユーザー定義演算子
- 索引タイプ

配置テンプレートを使用して、これらのオブジェクト型に基づいたオブジェクトをインスタンス化することもできません。

関連項目： リフレッシュ・グループの詳細は、4-4 ページの「テンプレート一般情報」を参照してください。

テンプレート・パラメータ

各ターゲット・マテリアライズド・ビュー・サイトで一意のデータ・セットが必要な場合は、オブジェクト DDL で変数を定義できます。変数を定義してパラメータ付きテンプレートを作成すると、テンプレートをインスタンス化するときにカスタム・データ・セットを使用できるため、マテリアライズド・ビュー・サイトごとに異なるデータ・セットを使用できます。これらのパラメータは、オブジェクト DDL に組み込まれます。テンプレートのインスタンス化時に、個別のユーザー値がパラメータに代入されます。

Oracle では、テンプレートにデフォルト値およびユーザー固有のパラメータ値を指定できます。パラメータ値は、配置テンプレートの作成時または作成後に指定できますが、テンプレートをインスタンス化する前に指定する必要があります。テンプレートのインスタンス化時にパラメータの値を指定することはできません。

ユーザー固有のパラメータ値を使用した場合は、指定されたユーザーがテンプレートをインスタンス化するときにそのパラメータ値が自動的に使用されます。たとえば、変数 `region` について考えます。テンプレート `sales_temp` に対して次のようなユーザー固有のパラメータ値を指定したとします。

ユーザー	地域
fay	east
baer	west

この場合、マテリアライズド・ビューの定義 SELECT 文は次のようになります。

```
SELECT cust_id, sales_to_date, status FROM table_x WHERE region_id=:region;
```

ユーザー fay および baer がテンプレート sales_temp をインスタンス化すると、作成されるマテリアライズド・ビュー・データ・セットは次のようになります。

ユーザー fay		-	ユーザー baer	
cust_id	region	-	cust_id	region
a123	east	-	b123	west
a234	east	-	b234	west
a345	east	-	b345	west
a456	east	-	b456	west

WHERE 句のテンプレート・パラメータとセキュリティ

カスタマイズしたデータ・サブセットを作成する他に、CREATE MATERIALIZED VIEW 文の WHERE 句でテンプレート・パラメータを指定して、マテリアライズド・ビュー・サイトでは WHERE 句の条件を満たすデータのみを表示および変更するようにできます。たとえば、ユーザー固有のパラメータ・リストの region パラメータに次のように指定したとします。

ユーザー	地域
fay	east
baer	west

ユーザー fay がインスタンス化したマテリアライズド・ビューにアクセスするユーザーは、地域 east のデータのみを参照でき、この WHERE 句に適合するデータのみを表示、更新または削除できます。つまり、このマテリアライズド・ビューには地域 east のデータしか含まれていないため、このマテリアライズド・ビューのユーザーは地域 west のデータを表示、更新または削除できません。

ユーザー認証

配置テンプレートは、パブリックまたはプライベートのいずれかとして定義できます。これは、テンプレートの作成時に設定します。パブリック・テンプレートは、マスター・サイトへのアクセス権を持つすべてのユーザーがインスタンス化できます。

テンプレートがプライベート用として作成されている場合は、認証ユーザーのみがターゲット・テンプレートをインスタンス化できます。プライベート用として規定する場合は、マスター・サイトに認証されたユーザーのリストを作成します。認証されていないユーザーがターゲット・テンプレートをインスタンス化しようとすると、そのインスタンス化プロセスは失敗します。

配置サイト

集中制御の目的で、リモート・マテリアライズド・ビュー・サイトでインスタンス化された環境の特性の一部を監視および管理できます。具体的には、配置テンプレートをインスタンス化したサイトを監視できます（配置テンプレート名、認証ユーザーおよびインスタンス化された環境の状態を含みます）。

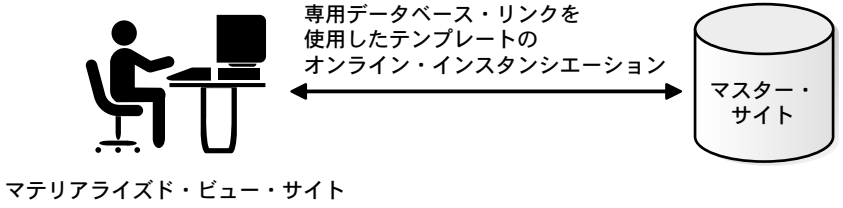
配置テンプレートのパッケージ化とインスタンス化

配置テンプレートの定義が完了した後で、テンプレートをパッケージ化して、リモート・マテリアライズド・ビュー・サイトでインスタンス化できます。パッケージ化した配置テンプレートがマテリアライズド・ビュー・サイトでインスタンス化されると、マテリアライズド・ビュー・サイト・オブジェクトが作成され、マテリアライズド・ビューにデータが移入されます。リモート・マテリアライズド・ビュー・サイトは、オンライン・インスタンシエーションまたはオフライン・インスタンシエーションのいずれかを使用して作成できます。

オンライン・インスタンシエーション

オンライン・インスタンシエーションでは、マテリアライズド・ビュー・サイトはターゲット・マスター・サイトに接続した状態で配置テンプレートをインスタンス化できます。オンライン・インスタンシエーション・プロセスでは、マテリアライズド・ビュー・サイトの構造が作成され、指定したデータ・サブセットがマスター・サイトから取り出されて適切なマテリアライズド・ビューに格納されます。

図 4-3 オンライン・インスタンシエーション



Oracle Enterprise Edition、Oracle Standard Edition または Oracle Personal Edition を使用するマテリアライズド・ビュー・クライアントの場合、オンライン・インスタンシエーション用の配置テンプレートのパッケージ化は、スクリプト・ファイルの生成を意味します。このスクリプト・ファイルがマテリアライズド・ビュー・サイトで実行されると、マテリアライズド・ビュー・オブジェクトが作成された後にマスター・サイトに接続され、マテリアライズド・ビューにデータが移入されます。マスター・サイトからネットワーク経由でマテリアライズド・ビューにデータを移入するには、`CREATE MATERIALIZED VIEW ... AS SELECT` などの SQL 文を使用します。

オンライン・インスタンシエーションのメリットの 1 つは、データ・サブセットがインスタンス化プロセス実行時点の最新の状態である点です。ただし、これにはデメリットもあります。オンライン・インスタンシエーションの実行中はマテリアライズド・ビュー・サイトとマスター・サイトが接続されている必要があるため、作成するマテリアライズド・ビュー環境のサイズによっては、ネットワークの通信量が増大する可能性があります。

また、モデム接続を使用しているラップトップ・ユーザーは、接続時間が長くなることがあります。接続時間は、作成されるオブジェクトの数、マテリアライズド・ビューの副問合せの複雑さ、送信されるデータの量によって変わります。低帯域幅のモデムで接続している場合、これらの要素による影響は特に大きくなります。

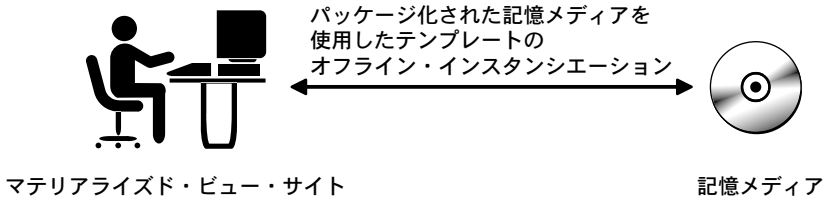
オフライン・インスタンシエーション

ピーク使用期間中のサーバー負荷を小さくし、リモートの接続時間を短縮するには、使用環境のテンプレートの**オフライン・インスタンシエーション**を選択します。オフライン・インスタンシエーション用のテンプレートのパッケージ化は、DDL およびデータ操作言語（DML）を含むスクリプトまたはバイナリ・ファイルの生成を意味します。このスクリプトまたはバイナリ・ファイルによって、配置テンプレートに定義したマテリアライズド・ビュー環境が作成され、データが移入されます。DBA は、スクリプトまたはバイナリ・ファイルをパッケージ化して記憶メディア（テープ、CD-ROM など）に保存し、マテリアライズド・ビュー・サイトに転送する方法を提供します。各マテリアライズド・ビュー・サイトで、それぞれのオフライン・インスタンシエーション・スクリプトが必要です。

インスタンス化用のテンプレートをパッケージ化すると、マテリアライズド・ビューの基になっているテンプレートの各マスター表のマテリアライズド・ビュー・ログに対して、変更のロギングが開始されます。特定のマスター表のマテリアライズド・ビュー・ログに記録された変更内容は、マスター表に基づくすべてのマテリアライズド・ビューがインスタンス化後にリフレッシュされるまで消去されません。したがって、テンプレートをパッケージ化した後で、テンプレートのインスタンス化とマテリアライズド・ビューのリフレッシュをできるだけ早く実行して、マテリアライズド・ビュー・ログが大きくなりすぎないようにする必要があります。

インスタンス化時は、テンプレートとデータがマスター・サイトではなく記憶メディアからプルされます。この操作により、ネットワークの通信量を軽減でき、ネットワークに常時接続する必要がなくなります。ただし、インスタンス化後は、マテリアライズド・ビュー・サイトにはパッケージ化した時点のマスター・サイトのデータが反映されているので、リフレッシュによって更新する必要があります。

図 4-4 オフライン・インスタンシエーション



オフライン・インスタンシエーションは、多数のラップトップやその他の非接続環境のコンピュータでターゲット・テンプレートをインスタンス化する大量配布に最適なソリューションです。

多重化マテリアライズド・ビューのオフライン・インスタンシエーション 配置テンプレートのオフライン・インスタンシエーションを使用してマテリアライズド・ビュー・サイトを作成すると、マスター表に定義されている競合解消方法がマテリアライズド・ビュー・サイトにプルされません。このマテリアライズド・ビュー・サイトに基づいてマテリアライズド・ビュー（多重化マテリアライズド・ビュー）を作成する場合は、データ整合性を保つためにこれらの競合解消方法が必要になることがあります。オンライン・インスタンシエーションを使用すると、インスタンス化中に競合解消方法がプルされます。

配置テンプレートのインスタンス化の使用例

インスタンス化のターゲット・サイトで、次のいずれかのデータベース・クライアントが稼働している必要があります。

- Oracle Enterprise Edition
- Oracle Standard Edition
- Oracle Personal Edition

表 4-1 に、配置テンプレートのインスタンス化の使用例の概要を示します。

表 4-1 配置テンプレートのインスタンス化の使用例

インスタンス化のタイプ	マテリアライズド・ビュー・クライアントのタイプ	説明
オフライン	Oracle Enterprise Edition Oracle Standard Edition Oracle Personal Edition	ユーザーは、SQL*Plus を使用してオフライン・インスタンシエーション・スクリプトを実行します。オフライン・インスタンシエーション・スクリプトには、マテリアライズド・ビュー・サイトのオブジェクトを作成する CREATE 文と、マテリアライズド・ビューにデータを移入する INSERT 文が含まれます。
オンライン	Oracle Enterprise Edition Oracle Standard Edition Oracle Personal Edition	ユーザーは、SQL*Plus を使用してオンライン・インスタンシエーション・スクリプトを実行します。オンライン・インスタンシエーション・スクリプトには、マテリアライズド・ビュー・サイトのオブジェクトを作成する CREATE 文が含まれます。マテリアライズド・ビュー・オブジェクトが作成されると、オンライン・インスタンシエーション・スクリプトがマスター・サイトに接続し、CREATE MATERIALIZED VIEW ...AS SELECT 文を使用してマテリアライズド・ビューを作成し、これにデータを移入します。

DBA またはターゲット・ユーザーのいずれかが、配置テンプレートをパッケージ化できます。配置テンプレートをパッケージ化するには、レプリケーション・マネージメント・ツールの「Template Script Generation Wizard」（オフライン・インスタンシエーションの場合）またはレプリケーション・マネージメント API（オフラインまたはオンライン・インスタンシエーションの場合）を使用します。配置テンプレートをパッケージ化するには、エンド・ユーザーはパブリック API を使用し、DBA は一般的にプライベート API を使用します。

通常、配置テンプレートをオフラインでインスタンス化する場合は DBA がパッケージ化を実行しますが、配置テンプレートをオンラインでインスタンス化する場合はユーザーがパッケージ化を実行する可能性があります。ただし、ユーザーまたは DBA にとっては使用する API コールが異なること以外に制約はなく、どちらもオンラインまたはオフラインによるパッケージ化を実行できます。

配置テンプレートをパッケージ化するには、次のレプリケーション・マネージメント API のファンクションを使用できます。

プライベート・ファンクション（DBA のみ使用）：

- DBMS_REPCAT_RGT.INSTANTIATE_OFFLINE
- DBMS_REPCAT_RGT.INSTANTIATE_ONLINE

パブリック・ファンクション：

- DBMS_REPCAT_INSTANTIATE.INSTANTIATE_OFFLINE
- DBMS_REPCAT_INSTANTIATE.INSTANTIATE_ONLINE

関連項目： ファンクションの詳細は、6-28 ページの「配置テンプレートをインスタンス化するマテリアライズド・ビュー・サイトの準備」および『Oracle9*i* レプリケーション・マネージメント API リファレンス』を参照してください。

注意： オフライン・インスタンシエーション用の配置テンプレートをパッケージ化すると、関連するマテリアライズド・ビュー・ログにより、テンプレートにパッケージ化したマテリアライズド・ビューのロギングが開始されます。この即時ロギングによって、マテリアライズド・ビュー・サイトでオフライン・インスタンシエーション完了後に高速リフレッシュを実行できます。オフライン・インスタンシエーションの実行後、マテリアライズド・ビュー・ログを監視して、リモート・マテリアライズド・ビュー・サイトのリフレッシュが適切なときに実行されていることを確認してください。テンプレートをパッケージ化するときにロギングが開始されるため、リモート・マテリアライズド・ビュー・サイトがリフレッシュされないとマテリアライズド・ビュー・ログが大きくなりすぎることがあります。

配置テンプレートのアーキテクチャ

Oracle では、マテリアライズド・ビュー環境を迅速かつ効率的に配布するために、配置テンプレートに標準マテリアライズド・ビュー・アーキテクチャを使用しています。配置テンプレートでは、マテリアライズド・ビュー定義の作成、リフレッシュ特性、競合解消およびグループ化に、マテリアライズド・ビュー環境を手動で作成する場合と同じ方法を使用します。ただし、配置テンプレートでは DDL を実行してオブジェクトを即時に作成するのではなく、テンプレートをインスタンス化するときに、配置テンプレートに含まれるオブジェクト DDL が実行される点が異なります。

システム表に格納されたテンプレート定義

配置テンプレートでは、マテリアライズド・ビュー・サイトで DDL を実行してマテリアライズド・ビュー環境を即時に作成するのではなく、マテリアライズド・ビューおよび関連するオブジェクト定義が配置テンプレートに格納されます。すべてのオブジェクト定義を配置テンプレートに追加した後、テンプレートをインスタンス化すると、格納されているすべての DDL がリモート・マテリアライズド・ビュー・サイトで実行され、必要なマテリアライズド・ビュー環境が作成されます。

これらのオブジェクト定義はすべて、配置テンプレートの定義サイトにあるシステム表に、配置テンプレート名をキーにして格納されます。配置テンプレートをパッケージ化すると、格納されているオブジェクト DDL がシステム表からプルされ、バイナリ・ファイルのインスタンス化スクリプトが作成されます。

標準 DDL の使用

テンプレートのオブジェクト定義は、マテリアライズド・ビュー・サイトでローカルにオブジェクトを作成する場合と同じ DDL を使用して作成されます。たとえば、次の文を発行するとマテリアライズド・ビューが作成されます。

```
CREATE MATERIALIZED VIEW hr.departments_mv
      REFRESH FAST WITH PRIMARY KEY FOR UPDATE AS SELECT
      department_id, department_name, manager_id, location_id
    FROM hr.departments@orc1.world;
```

これと同じマテリアライズド・ビューを配置テンプレートに追加するには、レプリケーション・マネージメント・ツールの「Deployment Template Wizard」を使用するか、次の例に示すように CREATE_TEMPLATE_OBJECT ファンクションを使用します。

```
DECLARE
   tempstring VARCHAR2(3000);
   a NUMBER;
BEGIN
   tempstring := 'CREATE MATERIALIZED VIEW hr.departments_mv
      REFRESH FAST WITH PRIMARY KEY FOR UPDATE AS SELECT
      department_id, department_name, manager_id, location_id
      FROM hr.departments@orc1.world';
   a := DBMS_REPCAT_RGT.CREATE_TEMPLATE_OBJECT (
           refresh_template_name => 'hr_refg_dt',
           object_name => 'departments_mv',
           object_type => 'SNAPSHOT',
           ddl_text => tempstring,
           master_rollback_seg => 'rbs');
END;
/
```

注意： ddl_text パラメータの一重引用符で囲まれた DDL 文の末尾には、セミコロンを付けないでください。

このファンクションを実行すると、dt_mviewenv という名前の配置テンプレートにマテリアライズド・ビュー定義が追加されます。このマテリアライズド・ビューがインスタンス化されると、マテリアライズド・ビュー mview_test が作成されます。マテリアライズド・ビューを作成するのみでなく、表、トリガー、プロシージャ、索引およびその他のオブジェクト定義を配置テンプレートに追加できます。

マテリアライズド・ビューを作成するときは、マテリアライズド・ビューの問合せで表所有者のスキーマ名を必ず指定してください。前述の例では、employees 表の所有者に hr が指定されています。

関連項目： CREATE_TEMPLATE_OBJECT ファンクションの使用方法の詳細は、『Oracle9*i* レプリケーション・マネージメント API リファレンス』の「DBMS_REPCAT_RGT パッケージ」の「CREATE_TEMPLATE_OBJECT ファンクション」を参照してください。

パッケージ化およびインスタンス化のプロセス

リモート・マテリアライズド・ビュー・サイトでのインスタンス化用に配置テンプレートをパッケージ化するときは、オンライン・インスタンシエーション用とオフライン・インスタンシエーション用のテンプレートを用意します。インスタンス化プロシージャによって、リモート・マテリアライズド・ビュー環境が作成され、その環境にデータが移入されます。

オンライン・インスタンシエーション用の配置テンプレートのパッケージ化

オンライン・インスタンシエーション用に配置テンプレートをパッケージ化すると、リモート・マテリアライズド・ビュー環境の作成に必要な DDL が生成され、すべてのテンプレート・パラメータが代入されます。生成された DDL が格納される場所は、マテリアライズド・ビュー・クライアントのタイプによって異なります。

オンライン・インスタンシエーション・スクリプトは、テンプレートをパッケージ化するレプリケーション・マネージメント API が実行されるコンピュータのハード・ドライブ上にローカルに格納されます。このコンピュータがマテリアライズド・ビュー・サイトのコンピュータでない場合は、オンライン・インスタンシエーション・ファイルをマテリアライズド・ビュー・サイトに転送する必要があります。

オフライン・インスタンシエーション用の配置テンプレートのパッケージ化

オフライン・インスタンシエーション用に配置テンプレートをパッケージ化するときに、リモート・マテリアライズド・ビュー環境の作成に必要な DDL とマテリアライズド・ビュー環境へのデータの移入に必要な DML を、生成したファイルに格納します。また、パッケージ化の最中に、すべてのテンプレート・パラメータの代入が実行されます。

テンプレートをパッケージ化するときに、オフライン・インスタンシエーション用のスクリプトまたはバイナリ・ファイルを作成し、ハード・ディスク、CD-ROM またはテープなどの記憶デバイスに保存します。オフライン・インスタンシエーション用の配置テンプレートをパッケージ化するには、レプリケーション・マネージメント・ツールの「Template Script Generation Wizard」またはレプリケーション・マネージメント API を使用できます。

オフライン・インスタンシエーション・スクリプトは、テンプレートをパッケージ化する要求が発行されるコンピュータのハード・ドライブ上にローカルに格納されます。このコンピュータがマテリアライズド・ビュー・サイトのコンピュータでない場合は、オフライン・インスタンシエーション・ファイルをマテリアライズド・ビュー・サイトに転送する必要があります。

リモート・マテリアライズド・ビュー・サイトでテンプレートをインスタンス化するときは、記憶メディアまたはローカルのハード・ドライブからスクリプトまたはバイナリ・ファイルが実行されます。これにより、マテリアライズド・ビュー環境が作成され、パッケージ作成プロセスで定義したデータ・セットが環境に移入されます。前述のように、個々のサイト用のデータ・セットを定義するテンプレート・パラメータはすべて、パッケージ作成プロセスで定義します。

オンライン・インスタンシエーション

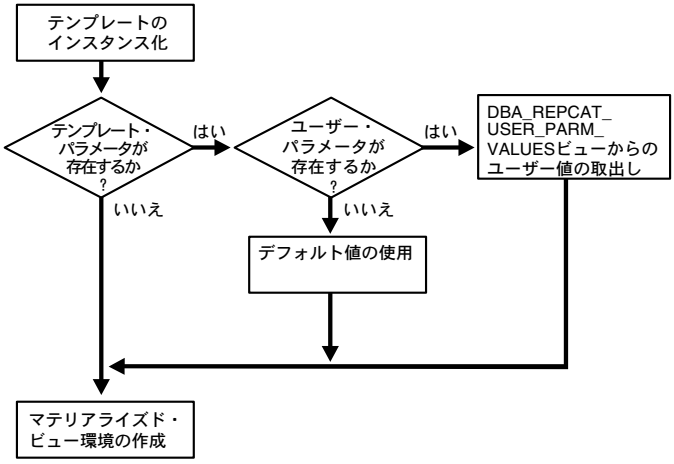
オンライン・インスタンシエーション・プロセスでは、マテリアライズド・ビュー・サイトの構造が作成され、指定したデータ・サブセットがマスター・サイトから取り出されて適切なマテリアライズド・ビューに格納されます。リモート・マテリアライズド・ビュー・サイトでオンライン・インスタンシエーション・プロセスを開始すると、配置テンプレート用に定義したパラメータが評価されます。パラメータに定義した値は、テンプレートのオブジェクト DDL を実行するときに、リモート・マテリアライズド・ビュー・サイトにカスタム・データ・セットをインストールするために使用されます。同時に、マテリアライズド・ビューがマスター・サイトに登録され、マテリアライズド・ビュー・ログにマスター表への変更のロギングが開始されます。

テンプレート・パラメータの値を定義するには、デフォルト・パラメータ値とユーザー・パラメータ値という 2 つの方法を使用できます。Oracle はこれらのパラメータ値が存在するかどうかをチェックし、次の順に使用します。

1. ユーザー・パラメータ値
2. デフォルト・パラメータ値

ユーザー・パラメータ値が定義してあり、リストに含まれているユーザーがテンプレートをインスタンス化する場合、テンプレートをインスタンス化するときにユーザー・パラメータ値が使用されます。ユーザー・パラメータ値を定義していない場合は、デフォルト・パラメータ値が使用されます。図 4-5 は、パラメータのチェック・プロセスを示します。

図 4-5 オンライン・インスタンシエーション時のパラメータのチェック



パラメータのチェックが終わると、テンプレートによって作成されたオブジェクトが、テンプレートの作成時に指定したリフレッシュ・グループに追加されます。

オフライン・インスタンシエーション

大量配布の環境では、ほとんどのマテリアライズド・ビュー環境でオフライン・インスタンシエーションを使用して、必要なマテリアライズド・ビュー環境を作成します。配置テンプレートをパッケージ化すると、スクリプトまたはバイナリ・ファイルが作成され、マテリアライズド・ビュー環境の作成に必要な DDL、インスタンス化プロセスで使用されるパラメータ値およびマテリアライズド・ビュー環境へのデータの移入に必要な DML が格納されます。

スクリプトまたはバイナリ・ファイルは、CD-ROM やフロッピィ・ディスクなどの記憶メディアにコピーしたり、Web サイトや FTP サイトに保存してリモート・マテリアライズド・ビュー・サイトにダウンロードしたりできます。このような柔軟な配布メカニズムによって、管理者とユーザーは最も効率的な方法で配置テンプレートをインスタンス化できます。

パッケージ化およびインスタンス化のオプション

配置テンプレートのパッケージ化とインスタンス化には、多数のオプションがあります。表 4-2 ではそれらのオプションを示し、パッケージ化とインスタンス化のメカニズムについて説明し、操作を実行するときに使用するドキュメントをリストしています。

表 4-2 パッケージ化およびインスタンス化のオプション

インスタンス化のタイプ	インスタンス化を実行するクライアントのタイプ	使用するテンプレートのパッケージ化	パッケージ化ドキュメント	使用するテンプレートのインスタンス化	インスタンス化のドキュメント
オフライン	Oracle Enterprise Edition Oracle Standard Edition Oracle Personal Edition	レプリケーション・マネージメント・ツールの「Template Script Generation Wizard」	レプリケーション・マネージメント・ツールのオンライン・ヘルプの「Package for Offline Instantiation: Overview」を参照（「Help Contents」で「Deployment Templates」→「Packaging and Instantiation」の順に選択）。	オフライン・インスタンシエーション・スクリプトおよび SQL*Plus	レプリケーション・マネージメント・ツールのオンライン・ヘルプの「Instantiate at Remote Materialized View Site」の項目を参照（「Help Contents」で「Deployment Templates」→「Packaging and Instantiation」の順に選択）。
オフライン	Oracle Enterprise Edition Oracle Standard Edition Oracle Personal Edition	レプリケーション・マネージメント API（PL/SQL パッケージおよび SQL*Plus）	『Oracle9*i* レプリケーション・マネージメント API リファレンス』にあるパッケージ化に関する説明を参照。	オフライン・インスタンシエーション・スクリプトおよび SQL*Plus	『Oracle9*i* レプリケーション・マネージメント API リファレンス』にある配置テンプレートのインスタンス化に関する説明を参照。
オンライン	Oracle Enterprise Edition Oracle Standard Edition Oracle Personal Edition	レプリケーション・マネージメント API（PL/SQL パッケージおよび SQL*Plus）	『Oracle9*i* レプリケーション・マネージメント API リファレンス』にあるパッケージ化に関する説明を参照。	オンライン・インスタンシエーション・スクリプトおよび SQL*Plus	『Oracle9*i* レプリケーション・マネージメント API リファレンス』にある配置テンプレートのインスタンス化に関する説明を参照。

インスタンス化の結果

リモート・マテリアライズド・ビュー・サイトで配置テンプレートのインスタンス化を実行すると、マテリアライズド・ビュー・サイトでローカルにマテリアライズド・ビュー環境を作成した場合とまったく同じ構造が作成されます。具体的には、指定した名前でマテリアライズド・ビューが作成され、制約整合性を保つために主キーに基づいて索引が作成されます。また、テンプレートのその他のオブジェクトも、マテリアライズド・ビューで手動で作成した場合と同じように作成されます。

オフライン・インスタンシエーションの場合は、サーバーでパッケージ化してからリモート・マテリアライズド・ビュー・サイトでインスタンス化するまでの時間が長くなると、インスタンス化後の最初のリフレッシュに必要な時間も長くなります。マテリアライズド・ビュー・サイトでは、マスター・サイトのマテリアライズド・ビュー・ログを使用して、テンプレートがパッケージ化された時点以降の高速リフレッシュを実行します。前述のように、マスター表に対して行った変更は、配置テンプレートをパッケージ化するとただちにマテリアライズド・ビュー・ログに記録されます。

> **関連項目：** 詳細は、3-44 ページの「マテリアライズド・ビューのアーキテクチャ」を参照してください。

マテリアライズド・ビュー・グループ

インスタンス化した配置テンプレートによって作成されたオブジェクトは、オブジェクトのマスター・グループ名と同じ名前のマテリアライズド・ビュー・グループに自動的に追加されます。たとえば、マスター・グループ `personnel` および `technical` のオブジェクトを含む `dt_mviewenv` 配置テンプレートをインスタンス化した場合、テンプレート・オブジェクトはマテリアライズド・ビュー・グループ `personnel` および `technical` にそれぞれ追加されます。前述のように、マテリアライズド・ビュー・グループはターゲット・マスター・グループと編成上の整合性を保つのみでなく、更新可能マテリアライズド・ビューを使用するために必要です。

> **関連項目：** 詳細は、3-53 ページの「マテリアライズド・ビュー・グループ」を参照してください。

リフレッシュ・グループ

最初に配置テンプレートを作成するときは、テンプレートのマテリアライズド・ビュー・オブジェクトを追加するリフレッシュ・グループの名前を定義する必要があります。インスタンス化プロセスが完了した後で、マスター・サイトに常時ネットワーク接続している場合は、リフレッシュ・グループ内のマテリアライズド・ビューを設定した間隔で自動的にリフレッシュするように指定できます。

Oracle Enterprise Edition、Oracle Standard Edition および Oracle Personal Edition の場合は、レプリケーション・マネージメント・ツールまたは `DBMS_REFRESH.CHANGE` プロシージャを使用して、リフレッシュ・グループのリフレッシュ間隔と次回リフレッシュするデータを変更できます。レプリケーション・マネージメント・ツールでこれらの設定を変更するには、リフレッシュ・グループを選択して、「Next Date」フィールドと「Interval」フィー

ルドを編集します。DBMS_REFRESH.CHANGE プロシージャを使用してこれらの設定を変更する場合は、interval パラメータと next_date パラメータを適切に設定します。マテリアライズド・ビュー・サイトからマスター・サイトに常時ネットワーク接続していない場合は、必要に応じてリフレッシュ・グループをリフレッシュできます。

next_date と interval の指定に使用できる簡単な日付式の例を、次に示します。

- next_date または interval が 1 時間の指定 :

  ```
  SYSDATE + 1/24
  ```

- next_date または interval が 7 日の指定 :

  ```
  SYSDATE + 7
  ```

関連項目 : 日付算術の詳細は、『Oracle9*i* データベース管理者ガイド』および『Oracle9*i* SQL リファレンス』を参照してください。

配置テンプレートの設計

配置テンプレートのパラメータと副問合せによるサブセット化を組み合せることで、DBA は、副問合せと行がサブセット化されたデータを使用して広範囲に分散したデータベース環境を管理できます。列のサブセット化の要件およびレプリケーション環境に必要なデータ・セットについては、さらにいくつかの設計上の考慮事項があります。

マテリアライズド・ビュー・データ・セットはマテリアライズド・ビューの問合せに基づいて定義されます。つまり、ユーザーに表示されるのは、マテリアライズド・ビューの定義問合せに適合するデータのみです。行および列のサブセット化により、カスタマイズ・データ・セットを含むマテリアライズド・ビューを作成できます。このようなマテリアライズド・ビューは、すべてのデータ・セットを必要としない地域事業所や営業担当員に役立ちます。

関連項目 : データのサブセット化の詳細は、3-18 ページの「マテリアライズド・ビューを使用したデータのサブセット化」を参照してください。

配置テンプレートを使用した列のサブセット化

列のサブセット化を使用すると、マスター表の列をマテリアライズド・ビューから除外できます。これを行うには、マテリアライズド・ビューの作成中に、SELECT 文に該当する列を指定します。列のサブセット化は、配置テンプレートを使用した場合のみ可能です。配置テンプレートを作成する前に、テンプレートの設計を行う必要があります。

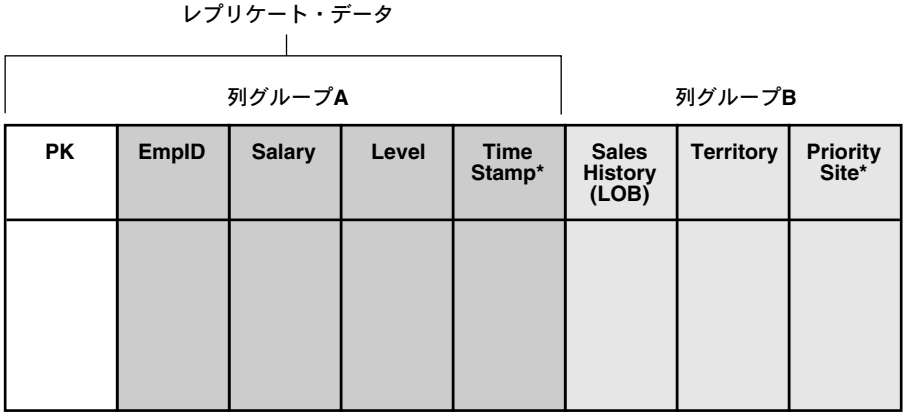
たとえば、多くの軽量なクライアントを含む大量配布の環境では、LOB データ自体を実際にレプリケートするのではなく、LOB データを含む表をレプリケートすることが必要な場合があります。これを行うには、列のサブセットを定義するときに、レプリケート対象として選択した列から LOB 列を除外します。

読取り専用マテリアライズド・ビューでは列の任意のサブセットを選択できます。更新可能マテリアライズド・ビューでは、列のサブセットに次の列が含まれている必要があります。

- 主キー列（複数も可）
- レプリケート列の競合解消に使用するすべての列（図 4-6 を参照してください）

注意： 列のサブセット化は列グループ内に構成することもできますが、これを行うとサイト間でデータの一貫性がなくなる可能性があるため、この方法はお薦めしません。列のサブセット化を使用する場合は、列グループ・レベルで列を除外するようにしてください。

図 4-6 列がサブセット化されたデータのレプリケート



* 列グループの競合解消に使用される列を示します。

列 pk、empid、salary および level（図 4-6 を参照してください）をレプリケートするマテリアライズド・ビューを追加する場合は、Time Stamp 列も含める必要があります。この列は、列グループ A に含まれる列の競合解消に使用されるためです。

注意：

- 列のサブセット化は、レプリケーション・マネージメント・ツールを使用してマテリアライズド・ビューを配置テンプレートに追加する場合にのみ使用できます。レプリケーション・マネージメント API を使用している場合は、列のサブセット化は使用できません。
- 列のサブセット化を定義するときは、マスター定義サイトが使用できる必要があります。複数のマスター・グループからの列がサブセット化されたマテリアライズド・ビューが配置テンプレートに含まれている場合は、各グループのマスター定義サイトが使用できる必要があります。

行のサブセット化

行のサブセット化を使用すると、マスター表の行をマテリアライズド・ビューから除外できます。これには WHERE 句を使用します。たとえば、次の文では、oe.orders@orc1.world マスター表に基づいたマテリアライズド・ビューが作成され、sales_rep_id 番号が 173 の営業担当員の行のみが含まれます。

```
CREATE MATERIALIZED VIEW oe.orders REFRESH FAST FOR UPDATE AS
  SELECT * FROM oe.orders@orc1.world
  WHERE sales_rep_id = 173;
```

orders 表で sales_rep_id 番号が 173 以外の行は、このマテリアライズド・ビューからは除外されます。

割当て表を使用した行のサブセット化

行のサブセット化に**割当て表**を使用する方が役に立つ場合があります。割当て表を使用すると、データベース内のあるエンティティを別のエンティティに関連付けることができ、どちらのエンティティの表にも割当て情報を格納する必要がありません。このテクニックについて、例を使用してわかりやすく説明します。

oe というスキーマの例を考えます。このスキーマでは、product_id 列が product_information 表の主キーで、warehouse_id が warehouses 表の主キーです。このスキーマでは inventories 表が割当て表として機能します。この表は product_id 列と warehouse_id 列を使用して製品を倉庫に割り当てます。この 2 つの列が、inventories 表の主キーを形成します。

oe スキーマ内のこの 3 つの表（inventories、product_information および warehouses）を使用すると、どの製品がどの倉庫にあるかを追跡するために、product_

id 情報を warehouses 表に格納する必要も、warehouse_id 情報を product_information 表に格納する必要もありません。これがいかに重要かを理解するために、inventories が存在せず、warehouse_id 列が product_information 表の外部キーである場合の例について考えてみます。

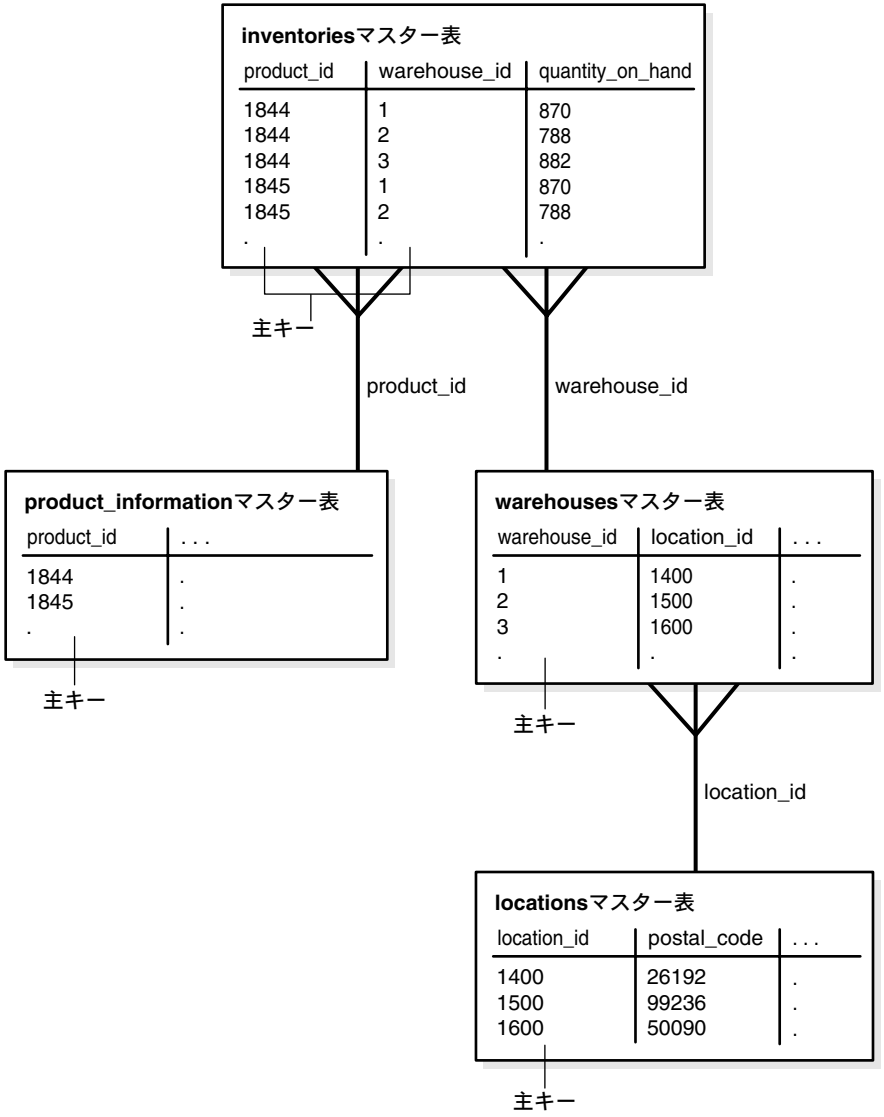
この場合に、営業担当員が最寄の倉庫の製品情報を格納するには、CREATE MATERIALIZED VIEW 文の WHERE 句にその倉庫の warehouse_id を指定する必要があります。たとえば、営業担当員は、次の文を使用してマテリアライズド・ビューを作成します。

```
CREATE MATERIALIZED VIEW oe.product_information REFRESH FAST FOR UPDATE AS
  SELECT * FROM oe.product_information@orc1.world
  WHERE warehouse_id = 1;
```

この構成のデメリットは、warehouse_id がマテリアライズド・ビュー定義にハードコードされているということです。会社が倉庫 1 を閉鎖するか、この営業担当員により近い場所に新しい倉庫を開設した場合、このマテリアライズド・ビュー定義は変更または再作成する必要があります。これと比べて、副問合せでの行のサブセット化と割当て表を使用する場合は、マテリアライズド・ビュー環境に対する変更を容易に制御できます。

oe スキーマでは、warehouse_id 列は product_information 表の一部ではありません。そのかわり、製品は inventories 表を介して倉庫に割り当てられます。製品と倉庫間のこの関係を図 4-7 に示します。

図 4-7 製品と倉庫の関係



新しい倉庫が建てられたり他の倉庫が閉鎖された場合は、inventories 表を使用して製品を別の倉庫に割り当てられます。inventories 表のような割当て表は、単一の管理点を作成するのみでなく、副問合せでの行のサブセット化とともに使用して、セキュリティを確保することもできます。たとえば、必要な場合は、特定の営業担当員が特定の倉庫のデータは見られるがその他の倉庫のデータは見られないように制限できます。

各営業担当員に特定の地域の担当責任があり、その地域の倉庫に格納されている製品の情報のみを必要とすると想定した場合、inventories 表を割当て表として使用し、それとともに副問合せでの行のサブセット化を使用して、特定の営業担当員に関係する情報のみを含む product_information マテリアライズド・ビューを作成できます。次の文により、適切なデータが営業担当員に提供されます。

```
CREATE MATERIALIZED VIEW oe.product_information REFRESH FAST FOR UPDATE AS
  SELECT * FROM oe.product_information@orc1.world pi
  WHERE EXISTS
    (SELECT * FROM oe.inventories@orc1.world inv
    WHERE pi.product_id = inv.product_id
    AND EXISTS
      (SELECT * FROM oe.warehouses@orc1.world w
      WHERE inv.warehouse_id = w.warehouse_id
      AND EXISTS
        (SELECT * FROM hr.locations@orc1.world loc
        WHERE w.location_id = loc.location_id
        AND loc.postal_code = :p_code)));
```

注意： この oe.product_information マテリアライズド・ビューを作成するには、postal_code を hr.locations 表のマテリアライズド・ビュー・ログにロギングする必要があります。詳細は、6-16 ページの「マテリアライズド・ビュー・ログへの列のロギング」を参照してください。

---

product_information マテリアライズド・ビューは、*p_code* 変数により指定された郵便番号が示す倉庫に格納されている製品の製品情報が移入されます。CREATE MATERIALIZED VIEW 文の最後の行にある *p_code* 変数に注意してください。

この柔軟な機能により、管理者は inventories 表に簡単な変更を加えるだけで、マテリアライズド・ビューのデータ・セットを容易に制御できます。マテリアライズド・ビュー作成のために SQL 文を変更する必要はありません。たとえば、新製品が特定の倉庫に追加された場合、管理者はこの製品を倉庫に割り当てる行を inventories 表に追加するのみです。マテリアライズド・ビューの次のリフレッシュの後、倉庫の製品情報を追跡するマテリアライズド・ビュー・サイトに製品のデータが追加されます。

データ・セット

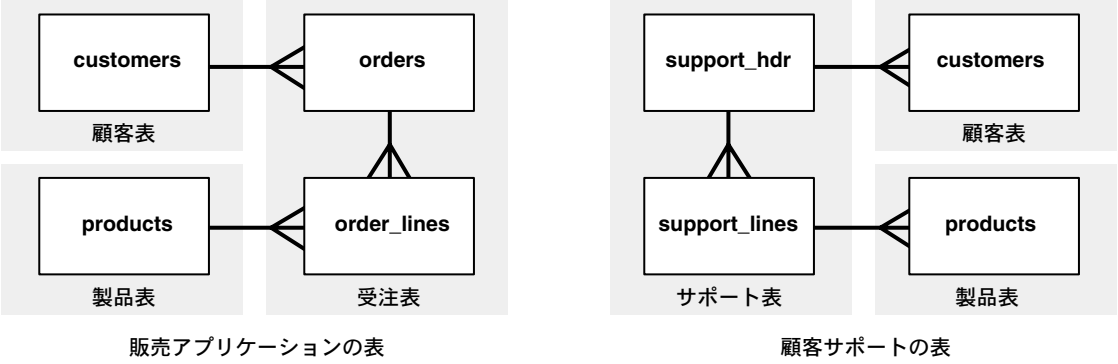
配置テンプレートを設計するときは、ターゲット・データにアクセスする様々なユーザーのタイプを考慮します。たとえば、営業担当員と技術者はいずれも顧客情報を必要としますが、両方が販売情報を必要とするわけではありません。ユーザーが自分に無関係なデータを含む配置テンプレートをインスタンス化すると、余分な記憶領域が必要になり、リフレッシュ時間が長くなるため、このようなインスタンス化は避けてください。

逆に、販売情報と顧客情報の両方を必要とするユーザーが、重複したデータを共有する複数の配置テンプレートをインスタンス化する必要があることも不都合です。複数のテンプレートをインスタンス化すると、データ整合性の問題が発生することがあります。つまり、各配置テンプレートが異なるリフレッシュ・グループを使用するため、2 つの配置テンプレートのデータが別々のタイミングでリフレッシュされてデータ整合性の問題が生じる可能性があります。

この場合は、営業担当員、顧客サービス技術者、および両方のデータ・セットを必要とするユーザー向けに、それぞれ 1 つずつ配置テンプレートを作成するのが最善のソリューションです。

時間と DBA の手間をかけずに 3 種類の配置テンプレートを作成するには、まず両方のデータ・セットを含むテンプレートを作成し、それを 2 つコピーして、それぞれ不要な項目を削除して 2 つのテンプレートを作成するのが最善の方法です。

図 4-8 営業担当員用テンプレートと顧客サービス技術者用テンプレートの相違



設計上のもう 1 つの考慮点は、パラメータの使用です。図 4-8 に示されている表の多くで `customer_id` フィールドが使用される場合、テンプレート・オブジェクトのそれぞれに同じパラメータを定義できます。同じパラメータを使用すると、必要なデフォルト・パラメータ値の定義は 1 回のみで、インスタンス化プロセス時にその定義がすべてのオブジェクトに使用されます。

副問合せに基づくサブセット化を行うマテリアライズド・ビューで使用する場合は、1 つのテンプレート・パラメータを使用するとさらに便利です。パラメータを 1 つにすることによって、ユーザーは必要な顧客に関するデータのみを受け取ることができます。たとえば、次の CREATE MATERIALIZED VIEW 文について考えます。

```
CREATE MATERIALIZED VIEW sales.orders AS
 SELECT * FROM sales.orders@orc1.world o
  -- conditions for customers
  WHERE EXISTS
  ( SELECT c_id FROM sales.customer@orc1.world c
    WHERE o.c_id = c.c_id
      AND EXISTS
    ( SELECT * FROM sales.assignment@orc1.world a
     WHERE a.c_id = c.c_id
     AND EXISTS
     ( SELECT * FROM sales.salesperson@orc1.world s
       WHERE s.s_id = :salesperson_id)));

CREATE MATERIALIZED VIEW sales.customer AS
SELECT c_id FROM sales.customer@orc1.world c
  -- conditions for customers
  WHERE EXISTS
   ( SELECT * FROM sales.assignment@orc1.world a
     WHERE a.c_id = c.c_id
     AND EXISTS
     ( SELECT * FROM sales.salesperson@orc1.world s
       WHERE s.s_id = :salesperson_id)));
```

作成する 2 つのマテリアライズド・ビューには明示的な salesperson_id フィールドが含まれていませんが、副問合せに基づくサブセット化を行うことによって、パラメータを使用して必要なデータ・セットのみをインスタンス化できるので、非常に効率的です。1 つのパラメータ（:salesperson_id）を使用すると、マテリアライズド・ビューの管理およびインスタンス化が、DBA および配置テンプレートを使用するユーザーの両方にとって容易になります。

設計に関するその他の考慮事項

最後に、リモート・マテリアライズド・ビュー・サイトで他にどのようなオブジェクトを作成する必要があるかを考えます。具体的には、次の点を検討してください。

- マテリアライズド・ビュー・サイトとマスター・サイト間に必要なデータベース・リンクを作成する DDL を含める必要があるかどうか
- マテリアライズド・ビュー環境で必要となるトリガーまたはプロシージャの種類
- 非レプリケート・データを格納する表を作成する必要があるかどうか
- 索引を追加する必要があるかどうか

マテリアライズド・ビュー作成のローカル制御

配置テンプレートは、マテリアライズド・ビュー環境を作成および配布するための最も効率的な方法です。配布先が 2 つまたは 3 つのサイトのみの場合でも、配置テンプレートを使用することによって、個々のサイトにマテリアライズド・ビュー環境を作成するよりもはるかに少ない手順でマテリアライズド・ビュー環境を作成できます。配置テンプレートを使用すると、1 度マテリアライズド・ビュー環境を作成して、必要な数のサイトにそれを配布できます。

ただし、疑問が 1 つ残っています。マテリアライズド・ビュー環境を作成して配布するのに配置テンプレートを使用することが最も効率的な方法ならば、どのような場合にリモート・マテリアライズド・ビュー・サイトでローカルにマテリアライズド・ビュー環境を作成するのかということです。通常、マテリアライズド・ビュー・サイトでローカル制御を維持する必要がある場合は、「Materialized View Group Wizard」を使用してマテリアライズド・ビュー環境を作成するか、マテリアライズド・ビュー・サイトでローカルにマテリアライズド・ビュー環境を作成します。

マテリアライズド・ビュー作成のローカル制御が便利なのは、マテリアライズド・ビュー・サイトで、受け取るデータを制御することが望ましい場合などです。たとえば、通常は読取り専用マテリアライズド・ビュー・サイトである意思決定支援サイト（DSS）の場合、特に便利です。DSS サイトでは複雑な問合せを実行することがありますが、OLTP サイトの速度を低下させないようにしたり、OLTP サイトの DBA の負担を軽減するにはローカル制御が必要です。

マテリアライズド・ビューのローカル制御

配置テンプレートの主な利点の 1 つに、DBA が配置テンプレートの作成を集中して管理できる点があります。ただし、マテリアライズド・ビュー・サイトでなんらかの制御が必要になる場合があります。

次の場合はローカル制御が必要です。

- マテリアライズド・ビュー・サイトに熟練した DBA がいる場合
- マテリアライズド・ビュー・サイトが信頼できるサイトである場合
- マテリアライズド・ビュー・サイトで行のサブセット化を実行する場合

マテリアライズド・ビュー・グループは、マテリアライズド・ビュー・サイトの DBA または SQL の知識を持ったシステム・アナリストがレプリケーション・マネージメント・ツールの「Materialized View Group Wizard」を使用してマテリアライズド・ビュー・サイトでローカルに作成するため、マテリアライズド・ビュー・サイトでも制御できます。

ローカル制御を行う最適な例として、次の例を考えます。マルチマスター・レプリケーションでは行および列のサブセット化ができないため、主にデータのサブセット化のみの目的で更新可能マテリアライズド・ビュー・サイトが作成されることがあります。これらのサイトは通常は安全で、熟練した DBA が存在し、ユーザーやアプリケーションの要件を満たすためにローカル制御を行う必要があります。「Materialized View Group Wizard」またはレプリケーション・マネージメント API を使用して作成したマテリアライズド・ビュー・グループ

では、安全性の高い更新可能マテリアライズド・ビュー・サイトの要件を満たすために必要なローカル制御を実行できます。

また、前述のように、配置テンプレートを使用してマテリアライズド・ビュー環境を作成すると、マテリアライズド・ビュー環境のすべてのオブジェクトが同じリフレッシュ・グループに追加されます。このことはほとんどのインストレーションでは問題ありませんが、場合によってはマテリアライズド・ビュー・グループ内のオブジェクトをいくつかの異なるリフレッシュ・グループに割り当てる必要があります。

5

競合解消の概念およびアーキテクチャ

レプリケーション環境によっては、複数サイト間でのデータ・レプリケーションによって発生する可能性のあるデータ競合を解消する競合解消方法を作成する必要があります。この章の内容は、次のとおりです。

- 競合解消の概念
- 競合解消のアーキテクチャ

競合解消の概念

レプリケーションの競合は、複数のサイトから同じデータを同時に更新できるレプリケーション環境で発生します。たとえば、別々のサイトから発生した 2 つのトランザクションによって、同じ行がほとんど同時に更新されたときに競合が発生します。レプリケーション環境を構成するときは、レプリケーションの競合が発生する可能性の有無を考慮する必要があります。システム設計に競合を発生させる可能性がある場合や、競合が発生した場合には、なんらかの方法で競合を解消しないかぎりシステム・データは収束しません。

一般的に、まず必要なのは競合が発生しないレプリケーション環境を設計することです。ほとんどのシステム設計では、いくつかのテクニックを使用することによって、レプリケートするデータの全体または大部分で競合を回避できます。しかし、多くのアプリケーションでは、データの一部を複数のサイトからいつでも更新できる必要があります。その場合、レプリケーションの競合の発生に対処できる必要があります。

次の各項で、競合解消に関する次の項目について説明します。

- レプリケーションの競合を考慮してレプリケーション・システムを設計する方法
- 使用するレプリケーション環境で発生する可能性のある競合タイプの判別方法
- レプリケーション環境の設計においてレプリケーションの競合を回避する方法
- 競合を回避できない設計で Oracle が競合を検出および解消する方法

データ要件およびアプリケーション要件の理解

どのようなタイプのデータベース・アプリケーションおよびサポート・データベースを設計する場合でも、データベースまたはアプリケーション自体を構築する前にアプリケーション要件を理解することは非常に重要です。たとえば、各アプリケーションは、機能境界や依存性（受注入力、出荷および請求など）が正しく定義され、モジュール化されている必要があります。さらに、サポート・データベースのデータを正規化して、アプリケーション・システムのモジュール間の隠れた依存性を軽減する必要があります。

レプリケーション環境で動作するデータベースを作成する場合は、基本的なデータベース設計のルールのみでなく、その他の要件も検討する必要があります。まず、レプリケート・データを扱うアプリケーションの一般的な要件を考慮します。たとえば、一部のアプリケーションは読取り専用マテリアライズド・ビューの使用でも十分に対応できるので、結果的にレプリケーションの競合の発生を完全に回避できます。これに対して、大部分のレプリケート・データが読取り専用で、一部のデータ（たとえば、1 つまたは 2 つの表、あるいは特定の表内の 1 つまたは 2 つの列）をすべてのレプリケーション・サイトで更新可能にする必要があるアプリケーションもあります。この場合は、レプリケーションの競合が発生したときにレプリケート・データの整合性が完全に保たれるように、競合解消方法を決定する必要があります。

競合の検出と解消の例

競合を考慮したレプリケート・データベースの設計方法をよく理解するために、競合の検出および解消が可能な場合と不可能な場合がある、次のような環境を考えてみます。

- 通常、同じ商品に対して複数の予約を受け付けることができない予約システムでは、競合を解消できません。たとえば、コンサートの特定の座席を予約するときには競合を解消する方法がないため、予約システムの異なるレプリカにアクセスする異なるエージェントが、複数の顧客のために同じ座席を予約できません。
- 顧客管理システムでは、通常、競合を解消できます。たとえば、複数の営業担当員がレプリケーション環境内の異なるデータベースの顧客住所情報をメンテナンスできます。競合が発生しても、レコードに最新の更新内容を適用することで、更新の競合を解消できます。

レプリケーションの競合のタイプ

レプリケート・データベース環境で検出されるデータ競合のタイプは、次のとおりです。

- 更新の競合
- 一意性競合
- 削除の競合

レプリケーション環境で発生する可能性が最も高いのは更新の競合です。ただし、一意性競合および削除の競合にも対処できるように準備する必要があります。これらの 3 つのタイプの競合を回避できるようにデータベースを設計することをお薦めします。

更新の競合

更新の競合は、ある行に対する更新のレプリケーションが同じ行に対する別の更新と競合したときに発生します。異なるサイトから発生した 2 つの異なるトランザクションによって、同じ行がほとんど同時に更新されたときに発生することがあります。

一意性競合

一意性競合は、行のレプリケーションによってエンティティの整合性（PRIMARY KEY 制約または UNIQUE 制約など）が失われるときに発生します。たとえば、2 つの異なるサイトから発生した 2 つのトランザクションで、同じ主キー値を使用してそれぞれの表レプリカに行を挿入した場合について考えます。この場合、トランザクションのレプリケーションが原因で一意性競合が発生します。

削除の競合

削除の競合は、異なるサイトから発生した 2 つのトランザクションのうち、1 つのトランザクションで行が削除され、もう 1 つのトランザクションで同じ行に対して更新または削除が行われたときに発生します。これは、更新または削除の対象となる行が存在しなくなるためです。

データの競合およびトランザクションの順序

競合

順序競合は、マスター・サイトが 3 つ以上あるレプリケーション環境で発生します。なんらかの理由でマスター・サイト X への伝播がブロックされた場合、レプリケート・データへの更新を他のマスター・サイト間で伝播できます。伝播を再開したときに、これらの更新がその他のマスターで発生した順序と異なる順序でサイト X に伝播されると、更新の競合が発生する可能性があります。デフォルトでは、発生した競合はエラー・ログに記録され、依存関係にあるトランザクションが伝播および適用された後で再実行できます。順序競合の例は、5-18 ページの表 5-1 を参照してください。

マスター・サイトが 3 つ以上あるレプリケーション環境でデータ収束を保証するには、マスター・サイトがいくつあっても確実にデータ収束を行える競合解消方法（最も新しいタイムスタンプ、最小値、最大値、優先グループ、加算）を選択する必要があります。

最小値、最大値、優先グループおよび加算による競合解消方法は、一定の条件が存在するかぎり、マスター・サイトがいくつあってもデータ収束が保証されます。詳細は、5-19 ページの「競合解消のアーキテクチャ」で説明している競合解消方法を参照してください。

参照整合性

サポート・データがリモート・サイトに正常に伝播されなかった場合、レプリケート・トランザクションが順不同に適用されて、データ競合の他に参照整合性の問題が発生する可能性もあります。新規顧客が受注部門に電話をかけた場合を考えます。この場合、顧客レコードが作成され、受注が設定されます。顧客データより先に受注データがリモート・サイトに伝播された場合、受注によって参照される顧客がリモート・サイトに存在しないため、参照整合性のエラーが発生します。

参照整合性エラーが発生した場合は、エラーが生じたトランザクションをサポート・データがリモート・サイトに伝播された後で再度実行すると、簡単に解決できます。

競合の検出

レプリケーション・システムの各マスター・サイトは、レプリケーションの競合が発生するとそれを自動的に検出し、解消します。たとえば、あるマスター・サイトがシステム内の別のマスター・サイトに遅延トランザクション・キューをプッシュするときに、レプリケーションの競合があれば、その競合は呼び出された受信サイトのリモート・プロシージャによって自動的に検出されます。

マテリアライズド・ビュー・サイトが対応するマスター・サイトまたはマスター・マテリアライズド・ビュー・サイトに遅延トランザクションをプッシュすると、受信サイトで競合の検出および解消が行われます。マテリアライズド・ビュー・サイトでは、マテリアライズド・ビューのリフレッシュを実行することによってデータをリフレッシュします。このリフレッシュの仕組みによって、リフレッシュ完了後には、競合解消結果を含めたマテリアライズド・ビューのデータが、対応するマスター表またはマスター・マテリアライズド・ビューのデータと同一になります。したがって、マテリアライズド・ビュー・サイトでレプリケーションの競合の検出または解消を実行する必要はありません。

異なるタイプの競合の検出方法

レプリケーション・システムでの受信側のマスター・サイトまたはマスター・マテリアライズド・ビュー・サイトでは、次のような更新の競合、一意性競合および削除の競合が検出されます。

- レプリケート行の古い値（変更前の値）と受信サイトの同じ行の現在の値が異なる場合、受信サイトで更新の競合が検出されます。
- レプリケート行の INSERT または UPDATE の実行中に一意制約違反が発生した場合、受信サイトで一意性競合が検出されます。
- 行の主キーが存在しないために受信サイトで UPDATE 文または DELETE 文の対象の行が見つからない場合、受信サイトで削除の競合が検出されます。

注意： 行の更新の競合を検出および解消するには、伝播サイトから受信サイトに新旧の行に関するデータを送信する必要があります。パフォーマンスを最大にするには、更新の競合の検出および解消サポートに使用するデータの量をチューニングします。詳細は、5-34 ページの「パフォーマンスのメカニズムと競合の解消」を参照してください。

競合検出時の行の識別

レプリケーションの競合を正確に検出するには、データ・レプリケーション時に別々のサイトの対応する行を一意に識別し、一致させる必要があります。通常、Advanced Replication では、表の主キーを使用して表内の行を一意に識別します。表に主キーがない場合は、代替キーを指定する必要があります。代替キーとは、データ・レプリケーション時に表内の行を一意に識別するために使用する列または列のセットです。

注意： アプリケーションで表の主キーまたは代替キーを更新しないようにしてください。これは、Oracle で行を識別し、レプリケート・データの整合性を保つためです。

競合解消

競合が検出された場合は、競合を解消してすべてのサイトでデータ収束を実行します。Oracle には、更新の競合を解消するビルトインの競合解消方法が用意されています。通常は、この方法を使用することで、様々なレプリケーション環境のデータ収束を保証できます。また、一意性競合に対する競合解消方法もありますが、これはデータ収束を保証しません。

削除の競合または順序競合については、ビルトインの競合解消方法はありません。ただし、ビジネス・ルールに固有の競合解消方法を独自に作成できます。データ収束を保証しない競合解消方法（特に、一意性競合および削除の競合の解消方法）を作成する場合、通知機能を作成して DBA にデータ収束を手動で実現するように依頼する必要があります。

ビルトインの競合解消方法またはユーザー定義の競合解消方法のいずれを使用する場合も、競合が検出されると即時に適用されます。定義した競合解消方法で競合を解消できない場合は、その競合はエラー・キューにロギングされます。

競合解消が 1 つの時点で失敗するのを回避するために、別の競合解消方法を定義して、主要な競合解消方法の予備を作成できます。たとえば、タイムスタンプが同一であるため、最も新しいタイムスタンプによる競合解消方法では競合を解消できない場合（このような状況が発生することはほとんどありません）、サイト優先の競合解消方法を定義できます。これにより、タイムスタンプを使用せずにデータ競合を解消できます。

> **関連項目：** 変更をレプリケートせずに表を変更する方法の詳細は、『Oracle9*i* レプリケーション・マネージメント API リファレンス』を参照してください。自動的に解消されない競合を手動で解消するときに、この方法が必要になることがあります。

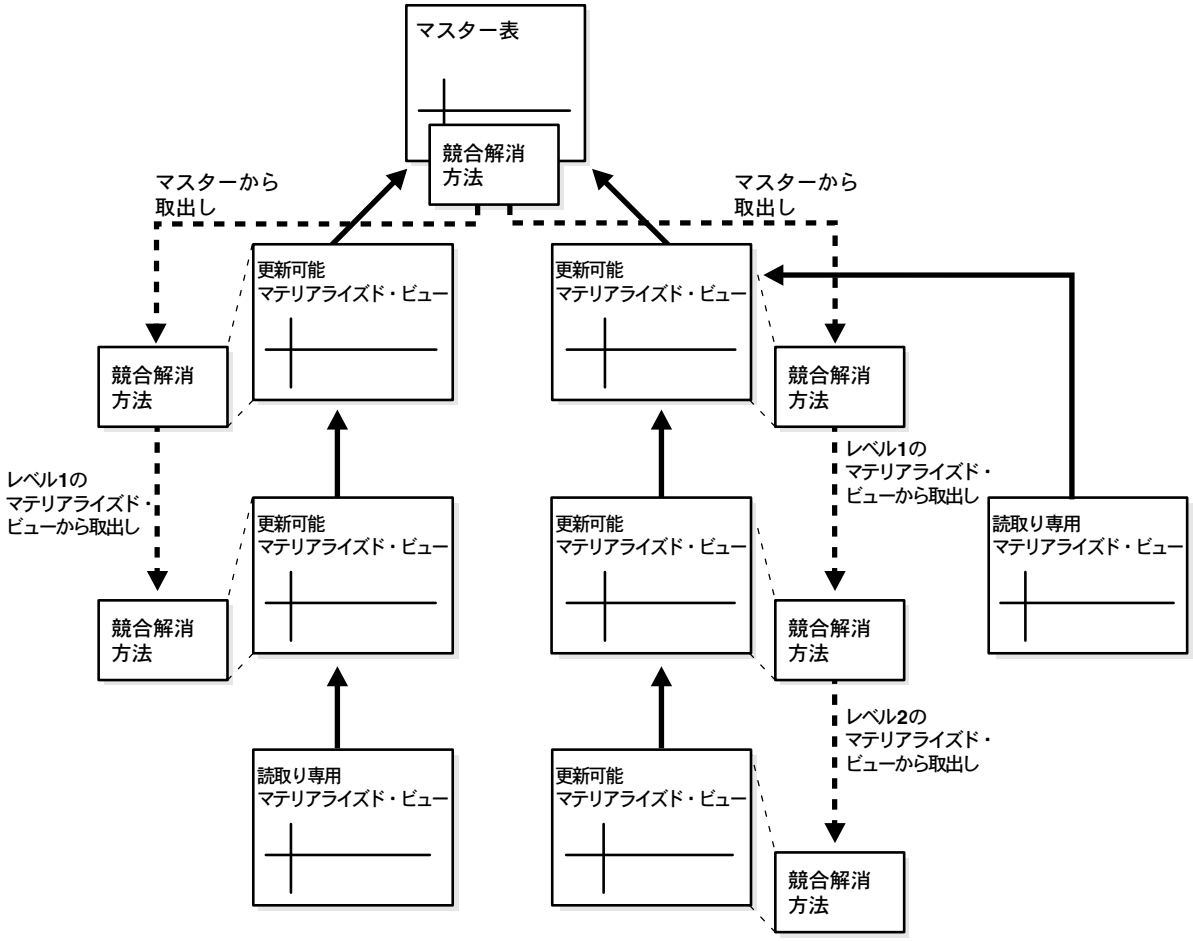
多重化マテリアライズド・ビューと競合解消

マスター表とこのマスター表に基づいた更新可能マテリアライズド・ビューがあった場合、マテリアライズド・ビューをリフレッシュすると変更がマスター・サイトにプッシュされます。プッシュによって競合が発生した場合には、マスター・サイトが構成済みの競合解消方法を使用して競合を処理します。次に、マテリアライズド・ビューはリフレッシュ完了時にマスターの変更を取り出します。リフレッシュは、常にマテリアライズド・ビュー・サイトで開始されます。

同様に、更新可能マテリアライズド・ビューのマスター・マテリアライズド・ビューは、マスター表と同じ動作をします。ただし、マスター・マテリアライズド・ビューでは、マテリアライズド・ビューからのプッシュによって発生する競合を処理するときに、マスターからプルした競合解消方法を使用します。ここで、マスターはマスター・サイトのマスター表、あるいは別のマテリアライズド・ビュー・サイトのマスター・マテリアライズド・ビューのいずれかです。競合解消方法をマテリアライズド・ビューに直接構成することはできません。かわりに、更新可能マテリアライズド・ビューを作成したとき、およびマテリアライズド・ビューのレプリケーション・サポートを生成したときに、上位のマスターから競合解消方法が自動的にプルされます。読取り専用マテリアライズド・ビューは、マスターから競合解消方法をプルしません。

たとえば、レベル 3 のマテリアライズド・ビューがレベル 2 のマスター・マテリアライズド・ビューに変更をプッシュしたとします。このプッシュにより、レベル 2 のマテリアライズド・ビューで競合が発生したとします。レベル 2 のマテリアライズド・ビューでは、競合を処理するために、レベル 1 のマスター・マテリアライズド・ビューから以前にプルした競合解消方法を使用します。同様に、レベル 1 のマテリアライズド・ビューでは、マスター・サイトから以前にプルした競合解消方法を使用して競合を処理します。図 5-1 に、この構成を示します。

図 5-1 競合解消と多重化マテリアライズド・ビュー



各更新可能マテリアライズド・ビューは、従属するマテリアライズド・ビューがない場合でも、マスターから競合解消方法をプルします。読取り専用マテリアライズド・ビューは、マスターから競合解消方法をプルしません。

多重化マテリアライズド・ビューが含まれる環境でマスター表の競合解消方法を変更する場合は、次の一般的な手順を実行します。

1. 列グループまたはキー列のいずれかを変更する場合で、マスター表に基づいたどの更新可能マテリアライズド・ビューとも最小限の通信しか実行していない場合は、次のサブステップを実行します。

   a. 変更するマスター表から最下位のマテリアライズド・ビューをリフレッシュします。リフレッシュにより、各マテリアライズド・ビューからすべての遅延トランザクションがそれぞれのマスターにプッシュされます。たとえば、マテリアライズド・ビューが 3 レベルある場合は、レベル 3 のマテリアライズド・ビューをリフレッシュします。

   b. 手順 a でリフレッシュしたマテリアライズド・ビューで、データ操作言語（DML）による変更をすべて停止します。

   c. マスター・サイトのマスター表に基づいたレベル 1 のマテリアライズド・ビューに手順 a および手順 b を実行するまで、各マテリアライズド・ビュー・レベルに対してこの 2 つの手順を繰り返します。

2. 必要な場合は、マスター・グループを静止します。静止が必要な場合の詳細は、このマニュアルの最初の方にある「Advanced Replication の新機能」を参照してください。

3. マスター定義サイトで競合解消方法を変更します。

4. DBMS_REPCAT パッケージの GENERATE_REPLICATION_SUPPORT プロシージャまたはレプリケーション・マネージメント・ツールを使用して、マスター定義サイトで影響されるオブジェクトのレポート・サポートを再生成します。

5. 手順 2 でマスター・グループを静止した場合は、マスター・グループのレプリケーション・アクティビティを再開します。

6. レプリケーション・サポートがまだ再生成されていない、最上位のレベル番号のマテリアライズド・ビューのレプリケーション・サポートを再生成します。再生成中に、現在の競合解消方法が上位のマスターからプルされます。この手順を初めて実行した場合、これがレベル 1 のマテリアライズド・ビュー用で、2 回目に実行した場合はレベル 2 のマテリアライズド・ビュー用、というようになります。DBMS_REPCAT パッケージの GENERATE_MVIEW_SUPPORT プロシージャまたはレプリケーション・マネージメント・ツールを使用して、マテリアライズド・ビューのレポート・サポートを再生成します。

7. 手順 1 でサブステップを実行した場合は、現在 DML 変更が禁止されている、最上位のレベル番号のマテリアライズド・ビューで DML 変更を使用できるようにします。この手順を初めて実行した場合、これがレベル 1 のマテリアライズド・ビュー用で、2 回目に実行した場合はレベル 2 のマテリアライズド・ビュー用、というようになります。

8. マスター表から最下位のマテリアライズド・ビューに手順 6 および手順 7 を実行するまで、各マテリアライズド・ビュー・レベルに対してこの 2 つの手順を繰り返します。たとえば、マテリアライズド・ビューが 3 レベルある場合は、最後の手順はレベル 3 のマテリアライズド・ビューに対して実行されます。

レプリケーション・サポートのこの再生成は、自動的には実行されません。複数のデータベース管理者がマスター・サイトとマテリアライズド・ビュー・サイトを管理する環境では、マスター・サイトのデータベース管理者は影響される全マテリアライズド・ビュー・サイトのデータベース管理者に対して、競合解消方法の変更を通知する必要があります。その後、すべてのデータベース管理者が協力して、前述の手順を適切に実行する必要があります。

更新可能マテリアライズド・ビューの列のサブセット化と競合解消

列のサブセット化を使用すると、マスター表にある列をマテリアライズド・ビューから除外できます。これには、マテリアライズド・ビューの作成中に SELECT 文に特定の列を指定します。列グループの列のサブセットのみを更新可能マテリアライズド・ビューに含める場合、廃棄による解消方法およびサイト優先順位による解消方法以外の競合解消方法をこの列グループに対して作成しないでください。競合解消方法がサイト優先順位による場合は、列のサブセット化は、マスター・サイトがマテリアライズド・ビュー・サイトよりも優先順位の高いシングル・マスター・レプリケーション環境でのみ使用する必要があります。

前述の廃棄による解消方法およびサイト優先順位による解消方法以外の競合解消方法の場合、更新可能マテリアライズド・ビューは列グループ内のいくつかの列に関する変更情報を送信しますが、その他の列に関しては情報を送信しないため、競合解消方法が適用されるときにエラーが戻されます。廃棄による解消方法とサイト優先順位による解消方法は列情報に依存しないため、これらの競合解消方法は列のサブセット化とともに使用できます。

たとえば、employees マスター表に列グループがあり、この列グループには employee_id 列、manager_id 列、department_id 列および timestamp 列が含まれているとします。最新のタイムスタンプによる競合解消方法をマスター・サイトの列グループに対して定義します。次に、employees マスター表に基づいて employees_mv という更新マテリアライズド・ビューを作成しますが、列のサブセット化を使用して、このマテリアライズド・ビューから department_id 列を除外します。マテリアライズド・ビューで employee_id 列または manager_id 列が更新されると、次のリフレッシュ時にこの変更情報がマスター・サイトに送信されます。マスター・サイトでは、競合解消方法が適用されるときに、列グループの残りの列（department_id）に関する情報がないため、エラーが戻されます。

多重化マテリアライズド・ビューを使用する場合は、前述のことを考慮します。競合解消方法がマスター・サイトからマスター・マテリアライズド・ビュー・サイトに取り出されるため、マスター・マテリアライズド・ビュー・サイトとそれに基づいた更新可能マテリアライズド・ビューに対しても、これと同じ規則が適用されます。

関連項目： 4-21 ページ「配置テンプレートを使用した列のサブセット化」

ネストした表と競合解消

ネストした表の各列に対しては、NESTED_TABLE_ID 列という非表示列がその表に作成されます。その他、ネストした表の要素を格納する記憶表という表も作成されます。記憶表は、親表の各行に対して、ネストした表の各要素を 1 行として格納します。記憶表には NESTED_TABLE_ID 列も含まれます。この列は親表の NESTED_TABLE_ID 列に対応し、ネストした表の要素を特定の親行に対して識別します。Advanced Replication では、ネストした表の列に対して特別の注意が必要です。競合解消に関しては、基になる記憶表に対しても、親表と同様の考慮が必要です。さらに、他にも考慮が必要な問題があります。

Oracle Replication では、ネストした表に対するデータ操作言語（DML）文を、親表および記憶表に対する DML 文とは別に処理します。ネストした表の列に対して DML 文が実行された場合、DML 文のタイプにより実行されるアクションが異なります。DML 文のタイプとそれぞれのタイプに対して実行されるアクションを、次の表に示します。

INSERT 文	DELETE 文	UPDATE 文
■ 新規行を記憶表に挿入します（ネストした表の値が NULL でも空でもない場合）。 ■ 新規行を親表に挿入し、NESTED_TABLE_ID の値は以前に挿入された記憶表の行を指します。	■ 削除される親表に対応付けられている記憶表の行を削除します。 ■ 親表の行を削除します。	■ 新規行を記憶表に挿入します（ネストした表の列が、NULL でも空でもない表の値に設定されている場合）。 ■ 親表の行を更新します。 ■ 古い記憶表の行を削除します。

ネストした表の競合の例 次の例は、ネストした表に対する DML 文が、解消の困難な競合につながる可能性があることを示しています。コード例に続いて、競合を最小化する方法について説明しています。

各学部の情報を department 表に格納する大学があったとします。この表には、各学部の履修課程の情報を格納するネストした表が含まれています。

```
CREATE TYPE Course AS OBJECT (
    course_no NUMBER(4),
    title VARCHAR2(35),
    credits NUMBER(1));
/

CREATE TYPE CourseList AS TABLE OF Course;
/
```

```
CREATE TABLE department (
      name VARCHAR2(20) primary key,
      director VARCHAR2(20),
      office VARCHAR2(20),
      courses CourseList)
      NESTED TABLE courses STORE AS courses_tab(
        (PRIMARY KEY(nested_table_id,course_no)));
```

この大学はアメリカ全土にキャンパスがあり、マルチマスター・レプリケーションを使用して様々なキャンパスをサポートしています。各キャンパスでは department 表を更新できます。この表はレプリケート表です。マスター・サイトの 1 つである univ1.world で、Psychology 学部に対して情報が挿入されます。

```
INSERT INTO department
      VALUES('Psychology', 'Irene Friedman', 'Fulton Hall 133',
      CourseList(Course(1000, 'General Psychology', 5),
      Course(2100, 'Experimental Psychology', 4),
      Course(2200, 'Psychological Tests', 3),
      Course(2250, 'Behavior Modification', 4),
      Course(3540, 'Groups and Organizations', 3),
      Course(3552, 'Human Factors in Business', 4),
      Course(4210, 'Theories of Learning', 4)));
```

Advanced Replication により、この挿入が全マスターに伝播されます。

次に、Psychology クラスに関する変更情報が届きます。つまり、クラスが 1 つ追加されます。この情報が univ1.world で更新されます。

```
UPDATE department SET courses = CourseList(
      Course(1000, 'General Psychology', 5),
      Course(2100, 'Experimental Psychology', 4),
      Course(2200, 'Psychological Tests', 3),
      Course(2250, 'Behavior Modification', 4),
      Course(3540, 'Groups and Organizations', 3),
      Course(3552, 'Human Factors in Business', 4),
      Course(4210, 'Theories of Learning', 4),
      Course(4320, 'Cognitive Processes', 4))
      WHERE name = 'Psychology';
```

univ1.world が更新をコミットした後、他のマスター・サイトに変更が伝播される前に、別のマスター・サイト univ2.world が、4320 および 4410 という 2 つのクラスが追加されたという情報を受け取ります。

```
UPDATE department SET courses = CourseList(
      Course(1000, 'General Psychology', 5),
      Course(2100, 'Experimental Psychology', 4),
      Course(2200, 'Psychological Tests', 3),
      Course(2250, 'Behavior Modification', 4),
```

```
Course(3540, 'Groups and Organizations', 3),
Course(3552, 'Human Factors in Business', 4),
Course(4210, 'Theories of Learning', 4),
Course(4320, 'Cognitive Processes', 4),
Course(4410, 'Abnormal Psychology', 4))
WHERE name = 'Psychology';
```

univ1.world での更新と univ2.world での更新の両方がプッシュされます。

department 表に対する更新の競合が発生します。更新を行った 2 人の各ユーザーは、自分の更新が INSERT 文以降発生した最初の更新であると考えます。しかし、実際には、最初に発生するのはローカルな更新であり、したがって、NESTED_TABLE_ID が変更されます。この 2 つの更新は親表に対する更新であるためです。問題になるのは、ネストした表の列に対する更新（記憶表の行と NESTED_TABLE_ID を変更する更新）です。親表の他の列の更新では問題はありません。

ローカルな表を最新状態にすることにより、この競合が解消されるとします。ローカルな更新によりすでに削除されている行を無視するように、記憶表に対して削除による競合解消方法が必要になります。記憶表に挿入された新規行は、リモート・サイトでの更新のため、親表に参照先がありません。記憶表のこの新規行も処理する必要があります。そうしない場合、これらの行が孤立してしまいます。記憶表には、department 表からアクセスできない course 行が増えてしまいます。

記憶表の行を操作しながら親表を更新するという競合解消は、マルチマスター・レプリケーション環境に 2 つのマスター・サイトがある場合は非常に困難で、マスター・サイトが増えた場合はほとんど不可能です。このような更新が必要な場合は、ネストした表に競合解消方法を定義せずに、競合は手動で解消するというのが最善の方法です。競合解消方法が正しくない場合、データの相違が発生する可能性があります。つまり、異なるマスター上の表が一致しなくなる危険性があります。

問題になる可能性のある更新を回避するための推奨事項 次に示した方法（推奨）により、前項で説明した更新の問題を回避できます。

- ネストした表に対して、最初は遅延されている外部キー制約を使用します。この制約により、記憶表で参照先がない行が発生しません。このような外部キー制約の例を、次に示します。

  ```
  ALTER TABLE courses_tab add CONSTRAINT courses_fk
    FOREIGN KEY(NESTED_TABLE_ID) REFERENCES department(courses)
    INITIALLY DEFERRED;
  ```

- 親表に対するすべての挿入で、空のネストした表が挿入されるようにします。ネストした表の値に NULL は使用しないでください。これにより、再利用可能な NESTED_TABLE_ID を作成できます。空のネストした表を含む挿入の例を、次に示します。

  ```
  INSERT INTO department (name, director, office, courses)
     VALUES('Psychology', 'Irene Friedman', 'Fulton Hall 133', CourseList());
  ```

- すべての挿入、削除および更新は、DML を介して親表に実行するのではなく、ネストした表に対して直接実行します。これにより、現在の NESTED_TABLE_ID 値を再使用できます。

  ネストした表から行を直接削除する例を、次に示します。

  ```
  DELETE FROM TABLE
     (SELECT courses FROM department WHERE name = 'Psychology');
  ```

  univ1.world のネストした表に次の行が直接挿入される例を考えてみます。

  ```
  INSERT INTO TABLE
     (SELECT courses FROM department WHERE name = 'Psychology')
     VALUES (Course(5000, 'Social Psychology', 5));

  INSERT INTO TABLE
     (SELECT courses FROM department WHERE name = 'Psychology')
     VALUES (Course(5100, 'Psychology of Personality', 4));
  ```

  次に、univ1.world に対する前述の挿入がプッシュされる前に、次の行が univ2.world のネストした表に直接挿入されます。

  ```
  INSERT INTO TABLE
     (SELECT courses FROM department WHERE name = 'Psychology')
     VALUES (Course(5000, 'Social Psychology', 5));

  INSERT INTO TABLE
     (SELECT courses FROM department WHERE name = 'Psychology')
     VALUES (Course(5100, 'Psychology of Personality', 4));

  INSERT INTO TABLE
     (SELECT courses FROM department WHERE name = 'Psychology')
     VALUES (Course(5500, 'Cognitive Neuroscience', 5));
  ```

  ここで、course 5000 および 5100 に関して記憶表に挿入された行で主キー競合が発生しますが、1 つのサイトからの挿入が失敗しても問題のない競合解消方法を記憶表に定義すると正しい結果が得られます。ただし、この場合、これらの挿入により 5-10 ページの「ネストした表の競合の例」で説明された複数の表が関与する複雑な問題は発生せず、NESTED_TABLE_ID の値も変更されないため、この値が失われることはありません。

- ネストした表の列の操作を含む挿入および更新を回避するようなトリガーを、親表に対して使用することを考慮します。このようなトリガーの例を次に示します。

```
CREATE OR REPLACE TRIGGER depart_trig
  AFTER INSERT OR UPDATE ON department
  FOR EACH ROW
  DECLARE
  new_ntid raw(100);
  old_ntid raw(100);
  BEGIN
-- obtain the nested table ids
  SELECT sys_op_tosetid(:new.courses) INTO new_ntid from dual;
  SELECT _tosetid(:old.courses) INTO old_ntid from dual;
  IF INSERTING THEN
-- raise error on insert of a null nested table column
    IF :new.courses IS NULL THEN
      raise_application_error(-20011, 'inserting null nested table ref');
    END IF;
-- raise error if new rows are inserted in the storage table
-- this is not strictly necessary, but it does enforce DML access
-- semantics of separate DMLS on parent table and storage table
    IF :new.courses.count != 0 THEN
      raise_application_error(-20012,
      'inserting rows into storage table while inserting parent table row');
    END IF;
  ELSE
-- raise error if update has caused the NESTED_TABLE_ID to change
    IF new_ntid != old_ntid THEN
      raise_application_error(-20013,
        'updating storage table reference while updating parent table row');
    END IF;
  END IF;
END;
/
```

前述の推奨事項は、記憶表の行が別の記憶表の行の親になる複数レベルのネストにも適用されます。これらの推奨事項は、ネストの各レベルでの適切な競合対策になります。

競合回避のテクニック

Oracle ではデータ競合を解消する強力な競合解消方法がありますが、レプリケート・データベースやフロントエンド・アプリケーションの設計時にまず優先する必要があるのは、データ競合を回避することです。以降の項では、レプリケーション競合の一部または全部を回避するためのテクニックを簡単に説明します。

列グループの使用

列グループを使用すると、列グループに対して競合解消方法を適用しない場合でも、競合の回避に役立ちます。レプリケート表に複数列グループが含まれている場合、更新の競合が発生しているかどうかの分析は各列グループごとに個別に行われます。

たとえば、列グループ a_cg と列グループ b_cg が含まれているレプリケート表を考えてみます。列グループ a_cg には、a1、a2 および a3 という列が含まれています。列グループ b_cg には、b1、b2 および b3 という列が含まれています。レプリケーション・サイトの sf.world と la.world で次の更新が発生します。

- ユーザー wsmith が sf.world の 1 つの行の列 a1 を更新します。
- 同一時刻に、ユーザー mroth が la.world の同一行の列 b2 を更新します。

この場合、Oracle では列グループ a_cg と b_cg での更新は別々に分析されるため、競合は発生しません。ただし、列グループ a_cg と b_cg がなかった場合は、表の全列が 1 つの列グループに入ることになり、競合が発生します。また、列グループが存在し、ユーザー mroth が列 b2 ではなく a3 を更新した場合は、a1 と a3 が同じ a_cg 列グループにあるため、競合が発生します。

関連項目： 列グループの詳細は、5-19 ページの「列グループ」を参照してください。

プライマリ・サイトおよび動的サイト所有権データ・モデルの使用

レプリケーションの競合の発生を回避する 1 つの方法として、システム内でレプリケート・データに同時更新アクセスできるサイトの数を制限する方法があります。レプリケート・データの 2 つの所有権モデルにより、この方法がサポートされます。プライマリ・サイト所有権と動的サイト所有権です。

プライマリ・サイト所有権 プライマリ所有権は、読取り専用レプリケーション環境でサポートされるレプリケート・データ・モデルです。プライマリ所有権によって、一連のレプリケート・データに更新アクセスできるサーバーは 1 つに限定されるため、すべてのタイプのレプリケーションの競合が発生しません。

表レベルでデータの所有権を制御せずに、アプリケーションで行のサブセット化および列のサブセット化の機能を使用すると、データの静的所有権をより細かく設定できます。たとえば、サイト別の基準でレプリケート表の特定の列または行に更新アクセスできます。

動的サイト所有権 動的所有権レプリケート・データ・モデルは、プライマリ・サイト所有権よりも制限の少ないモデルです。動的所有権の場合、データ・レプリカを更新する権利はサイトからサイトへ移動しますが、ある時点で特定のデータに更新アクセスできるサイトは 1 つしかありません。ワークフロー・システムに動的所有権の概念がわかりやすく示されています。たとえば、関連する部門のアプリケーションでは、注文を更新できる時期を判断するために、製品注文のステータス・コード（`enterable`、`shippable`、`billable` など）を使用できます。

> **関連項目：** 動的所有権データ・モデルの使用方法の詳細は、『Oracle9*i* レプリケーション・マネージメント API リファレンス』を参照してください。

特定のタイプの競合の回避

プライマリ・サイト所有権および動的所有権データ・モデルを使用するとアプリケーション要件に対して制限が多すぎる場合には、共有所有権データ・モデルを使用する必要があります。その場合でも、通常は、特定のタイプの競合を回避するためにいくつかの簡単な方法を使用する必要があります。

一意性競合の回避 一意性競合が発生しないようにレプリケーション環境を構成するのは非常に簡単です。たとえば、順序が互いに排他的な一連の順序番号を生成するように、各サイトで順序を作成できます。ただし、この方法を使用する場合、サイト数またはレプリケート表内の項目数が増加するほど問題が多く発生します。

また、複合主キーの一部として一意のサイト識別子を追加する方法もあります。

さらに、`SYS_GUID` ファンクションを使用してグローバルに一意の値を選択できます。選択した値を主キー（または一意の）値として使用すると、一意性競合がグローバルに回避されます。

注意： 順序は、レプリケーションの有効なオブジェクト型ではありません。したがって、各サイトで順序を作成する必要があります。

> **関連項目：** 順序の詳細は、2-28 ページの「順序のレプリケーションの代替方法」を、`SYS_GUID` ファンクションの詳細は『Oracle9*i* SQL リファレンス』を参照してください。

削除の競合の回避 削除の競合は、レプリケート・データ環境では常に回避する必要があります。一般的に、非同期の共有所有権データ・モデル環境で動作するアプリケーションでは、`DELETE` 文を使用して行を削除しないでください。かわりに、アプリケーションで行に削除のマークを付けて、論理的に削除された行をプロシージャ・レプリケーションを使用して定期的にパージするようにシステムを構成できます。

関連項目： 削除の競合を防止するように表を準備し、マークされた行をパージするレプリケート・プロシージャを作成する方法は、『Oracle9*i* レプリケーション・マネージメント API リファレンス』に記載された削除の競合防止方法の作成手順を参照してください。

更新の競合の回避 レプリケーション・システムでは、一意性競合と削除の競合の発生を回避するように試みた後、さらに更新の競合の発生数を少なく抑える必要があります。ただし、共有所有権データ・モデルでは、更新の競合をすべて回避することはできません。更新の競合をすべて回避できない場合は、発生する可能性のあるレプリケーションの競合のタイプを正確に理解し、競合が発生したときに解消できるようにシステムを構成する必要があります。

順序競合の回避 順序競合はできるかぎり回避するか自動的に解消するようにします。たとえば、順序競合が発生する可能性のあるマルチマスター構成で収束を保証する競合解消方法を選択します。

表 5-1 の例は、3 つのマスター・サイトでどのような場合に順序競合が発生するかを示しています。マスター・サイト A の優先順位は 30、マスター・サイト B の優先順位は 25、マスター・サイト C の優先順位は 10 です。*x* は、**サイト優先順位**による競合解消方法が割り当てられている列グループ内の特定の行の列です。優先順位の値が最も高いサイトに、最も高い優先順位が割り当てられています。優先順位値は任意の Oracle 番号で、連続する整数でなくてもかまいません。

表 5-1 例 : サイト優先順位による競合解消方法を使用した場合の順序競合

時間	アクション	サイト A	サイト B	サイト C
1	すべてのサイトで x=2 に更新され、一致しています。	2	2	2
2	サイト A で x=5 に更新します。	5	2	2
3	サイト C が使用不可になります。	5	2	ダウン
4	サイト A がサイト B に更新をプッシュします。 サイト A とサイト B が x=5 で一致します。 サイト C は使用不可のままです。 更新トランザクションはサイト A のキューに残っています。	5	5	ダウン
5	サイト C が x=2 で使用可能になります。 サイト A とサイト B が x=5 で一致します。	5	5	2
6	サイト B で x=5 を x=7 に更新します。	5	7	2
7	サイト B がトランザクションをサイト A にプッシュします。 サイト A とサイト B が x=7 で一致します。 サイト C は x=2 のままです。	7	7	2
8	サイト B がトランザクションをサイト C にプッシュします。 サイト C の古い値は x=2 です。 サイト B の古い値は x=5 です。 競合が検出され、サイト C の優先順位（10）より高い優先順位（25）を持つサイト B の更新を適用することで、競合が解消されます。 すべてのサイトが x=7 で一致します。	7	7	7
9	サイト A がそのトランザクション（x=5）を正常にサイト C にプッシュします。 競合が検出され、サイト C の優先順位（10）より高い優先順位（25）を持つサイト B の更新を適用することで、競合が解消されます。 サイト C（x=7）がサイト A の古い値（x=2）と一致しません。 サイト A はサイト C（10）より高い優先順位（30）を持ちます。 サイト A（x=5）の古い更新を適用することで、競合が解消されます。 この競合処理の順序が原因で、サイトのデータは収束しません。	7	7	5

競合解消のアーキテクチャ

レプリケーション環境に競合解消方法を実装するときに、アーキテクチャ上のメカニズムやプロセスを意識することはほとんどありません。この項では、競合解消に関連するいくつかのサポート・メカニズムについて説明し、Oracle に組み込まれている競合解消方法を様々な観点から説明します。

サポート・メカニズム

Oracle の競合解消に関連する最も重要なメカニズムは、列グループです。更新の競合の検出および解消はすべて、列グループに基づいて行われます。また、エラー・キューには、レプリケーション環境の競合検出アクティビティを監視するための重要な情報が記録されています。

列グループ

Oracle では、更新の競合を検出および解消するために列グループを使用します。列グループは、レプリケート表の 1 つ以上の列を論理的にグループ化したものです。レプリケート表の各列は、1 つの列グループの一部になります。マスター定義サイトでレプリケート表を構成するときに、列グループを作成して、各グループに列および対応する競合解消方法を割り当てることができます。

列グループには次のような特性があります。

- 列が属するのは 1 つの列グループのみです。
- 列グループは、表の 1 つ以上の列で構成できます。
- 競合解消が適用されるのは、列グループ内の列のみです。

> **関連項目：** 列グループを使用した競合回避の詳細は、5-15 ページの「列グループの使用」を参照してください。

複数列グループでのデータ整合性の保証 列グループを使用すると、データのタイプに応じて異なる競合解消方法を指定できます。たとえば、数値データの多くには算術的な解決メソッドが適しており、文字データの多くにはタイムスタンプによる解決メソッドが適しています。ただし、列グループの列を選択するときには、十分に考えた上で列をグループ化することが重要です。表内の複数の列で相互に整合性を保つ必要がある場合は、データ整合性を保証するためにそれらの列を同じ列グループに入れます。

たとえば、顧客表の郵便番号列で 1 つの解決メソッドを使用し、都市名列には別の解決メソッドを使用している場合、サイトは都市名に一致しない郵便番号に収束します。したがって、一般的には、競合解消方法が住所全体に対して適用されるように、1 つの住所のすべての構成要素を 1 つの列グループに入れる必要があります。

シャドウ列グループ デフォルトでは、各レプリケート表にシャドウ列グループがあります。表のシャドウ列グループには、特定の列グループ内にないすべての列が含まれます。表のシャドウ・グループには、競合解消方法を割り当てることはできません。したがって、列に競合解消方法を使用する必要がある場合は、列グループに必ずその列を入れてください。シャドウ列グループ内の列に関係する競合は検出されますが、このような競合を解消する競合解消方法は適用されません。

列オブジェクトと列グループ ユーザー定義型に基づいた Oracle オブジェクトのうち、表の 1 列を占有する Oracle オブジェクトを列オブジェクトと呼びます。1 つの列オブジェクトは複数の列グループにまたがることはできません。つまり、1 つの列グループと 1 つの列オブジェクトがあった場合、列オブジェクトとその全属性が列グループに含まれているか、列オブジェクトとその全属性が列グループから除外されているかのいずれかです。

Oracle のビルトインの競合解消方法では、未定義の列オブジェクトの属性値に基づいて競合を解消することはできません。列オブジェクトが NULL の場合、その属性は未定義です。

オブジェクト表と列グループ オブジェクト表とは、各行がユーザー定義型に基づいたオブジェクトを表す特殊な表です。オブジェクト表内の列のサブセットを含む列グループを指定できます。

ネストした表と列グループ ネストした表の記憶表は、競合解消では独立した表として扱われます。したがって、記憶表内の列のサブセットに基づいた列グループを作成できます。

エラー・キュー

競合解消方法によってデータ競合を解消できない場合、または競合解消方法を定義していなかった場合は、エラー・キューにデータ競合に関する情報が記録されます。

> **関連項目：** エラー・キューの詳細は、2-29 ページの「エラー・キュー」を参照してください。

更新の競合の一般的な解消方法

Oracle では更新の競合の解消方法として 8 つのビルトインの方法を用意していますが、このうち最も一般的に実装されるのは、最新のタイムスタンプによる競合解消方法および上書きによる競合解消方法です。

この 2 つの競合解消方法は使いやすく、適切な環境ではデータ収束が保証されるため、最も一般的です。最新のタイムスタンプによる競合解消方法および上書きによる競合解消方法の詳細は、以降の 2 つの項で説明します。

表 5-2 更新の競合の一般的な解消方法のデータ収束プロパティ

解消方法	複数のマスター・サイトでの収束
最新のタイムスタンプ	収束します （予備の方法を設定する場合）
上書き	収束しません

注意： Oracle のビルトインの競合解消方法では、1 つ以上のマテリアライズド・ビュー・サイトを持つシングル・マスター・サイト環境での収束を提供します。

最新のタイムスタンプ

最新のタイムスタンプによる方法を使用すると、更新発生時のタイムスタンプによって識別された最も新しい更新を原因とする競合を解消できます。

最新のタイムスタンプを使用した更新の競合解消方法の例を次に示します。

1. フェニックスの顧客が地元の営業担当員に電話をかけ、自分の住所情報を更新します。
2. 電話を切った後で、営業担当員に伝えた郵便番号が間違っていたことに顧客が気付きます。
3. 顧客は営業担当員に電話して正しい郵便番号を伝えようとしますが、担当者に電話がつながりません。
4. 顧客はニューヨーク本社に電話します。フェニックス・サイトではなく、ニューヨーク・サイトで住所情報が正しく更新されます。
5. ニューヨーク本社とフェニックス営業所を接続するネットワークが一時的にダウンします。
6. ニューヨークとフェニックスを接続するネットワークが回復すると、同じ住所に対する 2 つの更新が認識され、各サイトでの競合が検出されます。
7. 最新のタイムスタンプによる方法を使用して最新の更新内容が選択され、正しい郵便番号の住所が適用されます。

ターゲット環境　最新のタイムスタンプによる競合解消方法は、複数のマスター・サイトがあるレプリケーション環境でデータが収束するように機能します。タイムスタンプは常に増加しているので、この方法は複数のマスター・サイト間でのデータ収束を保証できる数少ない競合解消方法の 1 つです。また、任意の数のマテリアライズド・ビューがある環境でもうまく機能します。

サポート・メカニズム　タイムスタンプによる方法を使用するには、レプリケート表内の DATE 型の列を指定する必要があります。アプリケーションで列グループ内の列を更新するときは、ローカル SYSDATE で指定されたタイムスタンプ列の値も更新する必要があります。他のサイトから適用される変更については、タイムスタンプ値を発行元サイトのタイムスタンプ値に設定する必要があります。

注意：　タイムスタンプによる競合解消方法を使用するときは、2 つのサイトのタイムスタンプが同じ場合にサイト優先順位などの予備の方法がコールされるように設定する必要があります。

タイムスタンプの設定に関する問題

タイムスタンプによる解消方法を使用する場合は、レプリケート・データを管理している異なるサイトの標準時間帯を慎重に考慮してください。たとえば、レプリケーション環境が複数の標準時間帯にわたっている場合、システムを使用するアプリケーションは、すべてのタイムスタンプをグリニッジ標準時（GMT）などの共通の標準時間帯に変換する必要があります。さらに、システム内の 2 つのサイトのシステム時計が正しく同期化されていない場合、タイムスタンプの比較が不正確になり、アプリケーション要件を満たすことができない可能性があります。

EARLIEST または LATEST タイムスタンプによる更新の競合の解消方法を使用する場合は、次の方法でタイムスタンプ列をメンテナンスできます。

- 各アプリケーションに、タイムスタンプを同期化する機能を組み込む方法
- すべてのアプリケーションのタイムスタンプを自動的に同期化させる、レプリケート表のトリガーを作成する方法

クロックは、秒数を増加する値としてカウントします。タイムスタンプのメカニズムを正しく設計し、2 つのサイトのタイムスタンプが同一である場合のバックアップ方式を設定したと想定すると、最新のタイムスタンプによる方法で収束が保証されます（最大値による方法と同様）。ただし、最も古いタイムスタンプによる方法では、複数のマスター・サイトの収束は保証されません。

最新のタイムスタンプによる方法の実装　レプリケーション・マネージメント・ツールを使用して最新のタイムスタンプによる競合解消方法を定義する方法については、レプリケーション・マネージメント・ツールのオンライン・ヘルプを参照してください。

関連項目： レプリケーション・マネージメント API を使用してこのタイプの競合解消方法を定義する方法については、『Oracle9*i* レプリケーション・マネージメント API リファレンス』を参照してください。

上書き

上書きによる方法では、接続先サイトの現在の値を、発行元サイトの新しい値で上書きします。このため、マスター・サイトが複数の場合のデータ収束は保証されません。この方法は、1 つのマスター・サイトおよび複数のマテリアライズド・ビュー・サイトで使用するように設計されています。複数のマスター・サイトでこの方法を使用することもできますが、データ収束は保証されないので、なんらかのユーザー定義による通知機能とともに使用する必要があります。

たとえば、主に問合せに使用する 1 つのマスター・サイトがあり、すべての更新をそのマテリアライズド・ビュー・サイトで実行する場合は、廃棄による競合解消方法を選択できます。上書きによる方法は、次のような場合に役立ちます。

- 最大の目的がデータ収束である場合
- マスター・サイトが 1 つしかない場合
- 更新の選択について、ビジネス・ルールが特にない場合
- 複数のマスター・サイトがある環境で、DEFERROR データ・ディクショナリ・ビューでロギングした競合の解消をローカルの DBA にまかせる方法ではなく、データが正しく適用されるように確認する担当者への通知機能を使用する場合

ターゲット環境　上書きによる競合解消方法では、1 つのマスター・サイトと任意の数のマテリアライズド・ビューから構成されるレプリケーション環境でのデータ収束が保証されます。この点を考慮すると、上書きによる競合解消方法は大量配布環境で理想的な方法です。

競合が検出された場合、マテリアライズド・ビュー・サイトの値が使用されます。つまり、最後にリフレッシュが実行されたマテリアライズド・ビューに優先順位が与えられます。

サポート・メカニズム　上書きによる競合解消方法では、その他のサポート・メカニズムは必要ありません。

実装の概要　レプリケーション・マネージメント・ツールを使用して上書きによる競合解消方法を定義する方法については、レプリケーション・マネージメント・ツールのオンライン・ヘルプを参照してください。

関連項目： レプリケーション・マネージメント API を使用してこのタイプの競合解消方法を定義する方法については、『Oracle9*i* レプリケーション・マネージメント API リファレンス』を参照してください。

更新の競合のその他の解消方法

最新のタイムスタンプまたは上書きによる競合解消方法を使用してもレプリケーション環境で発生した更新の競合を解消できない場合は、その他の 6 つのビルトインの競合解消方法を使用できます。

表 5-3 更新の競合のその他の解消方法のデータ収束プロパティ

解消方法	複数のマスター・サイトでの収束
加算	収束します
平均	収束しません
廃棄	収束しません
最も古いタイムスタンプ	収束しません
最大値	収束します （列値は常に増加する必要あり）
最小値	収束します （列値は常に減少する必要あり）
優先グループ	収束します （順序付けられた更新値がある場合）
サイト優先順位	収束しません

加算

加算による競合解消方法は、数値型の単一列のみで構成される列グループで機能します。競合が発生した場合、いずれか一方の値を選択するのではなく、2 つの値の差が現在の値に加算されます。

加算による方法では、次の式に基づいて、発行元サイトの古い値と新しい値の差を、接続先サイトの現在の値に加算します。

```
current value = current value + (new value - old value)
```

この競合解消方法では、マスター・サイトやマテリアライズド・ビュー・サイトがいくつあっても値は収束します。

ターゲット環境　加算による競合解消方法では、最も適切なデータを選択するのではなくデータを収束するように設定されています。この方法は、預金の預入れや引出しが頻繁に行われてデータ競合が発生する可能性がある金融環境で便利です。収支では、いずれか一方の値を選択するのではなく、データを収束させることが重要です（引出しの値ではなく預入れの値を常に選択できればよいのですが）。

サポート・メカニズム　加算による競合解消方法では、その他のサポート・メカニズムは必要ありません。

加算による方法の実装　レプリケーション・マネージメント・ツールを使用して加算による競合解消方法を定義する方法については、レプリケーション・マネージメント・ツールのオンライン・ヘルプを参照してください。

> **関連項目：**　レプリケーション・マネージメント API を使用してこのタイプの競合解消方法を定義する方法については、『Oracle9*i* レプリケーション・マネージメント API リファレンス』を参照してください。

平均

平均による競合解消方法は、加算による方法の場合と同様に、数値型の単一列のみで構成される列グループで機能します。この方法では、現在の値に 2 つの値の差を追加するのではなく、現在の値と新しい値の平均を計算することで競合を解消します。

平均による競合解消方法では、発行元サイトの新しい列値と接続先サイトの現在の値の平均をとります。

```
current value = (current value + new value)/2
```

レプリケーション環境に複数のマスター・サイトが存在する場合、平均による方法では収束を保証できません。

ターゲット環境　平均による方法では、複数のマスター・サイトが存在するレプリケーション環境でのデータ収束を保証できないので、この方法は、1 つのマスター・サイトと任意の数の更新可能マテリアライズド・ビューから構成される大量配布の環境で実装するのが理想的です。

平均による方法は、いずれか一方の値を選択するのではなく、2 つの値の平均を取る科学アプリケーションで使用すると便利です（たとえば、平均温度や平均体重を計算する場合などです）。

サポート・メカニズム　平均による競合解消方法では、その他のサポート・メカニズムは必要ありません。

平均による方法の実装　レプリケーション・マネージメント・ツールを使用して平均による競合解消方法を定義する方法については、レプリケーション・マネージメント・ツールのオンライン・ヘルプを参照してください。

> **関連項目：**　レプリケーション・マネージメント API を使用してこのタイプの競合解消方法を定義する方法については、『Oracle9*i* レプリケーション・マネージメント API リファレンス』を参照してください。

廃棄

廃棄による競合解消方法では、発行元サイトの値は無視されるため、複数のマスター・サイトでのデータ収束が保証されることはありません。廃棄による方法では、発行元サイトからの新しい値を無視し、接続先サイトの現在の値を保持します。この方法は、1 つのマスター・サイトと複数のマテリアライズド・ビュー・サイトの間で使用するか、なんらかのユーザー定義による通知機能とともに使用してください。

たとえば、1 つのマスター・サイトとこのマスター・サイトに基づいた複数のマテリアライズド・ビューがあり、マテリアライズド・ビュー・サイトは主に問合せに使用され、更新はすべてマスター・サイトで実行される場合は、廃棄による競合解消方法を選択できます。廃棄による方法は、次のような場合に役立ちます。

- 最大の目的がデータ収束である場合
- マスター・サイトが 1 つしかない場合
- 更新の選択について、ビジネス・ルールが特にない場合
- 複数のマスター・サイトがある環境で、DEFERROR ビューでロギングした競合の解消をローカルの DBA にまかせる方法ではなく、データが正しく適用されるように確認する担当者への通知機能を使用する場合

ターゲット環境　廃棄による競合解消方法は、1 つのマスター・サイトと任意の数のマテリアライズド・ビュー・サイトから構成される大量配布モデルに最も適しています。競合が検出された場合、マテリアライズド・ビュー・サイトの値は無視されます。つまり、最初にリフレッシュが実行されたマテリアライズド・ビューに優先順位が与えられます。

サポート・メカニズム　廃棄による競合解消方法では、その他のサポート・メカニズムは必要ありません。

廃棄による方法の実装　レプリケーション・マネージメント・ツールを使用して廃棄による競合解消方法を定義する方法については、レプリケーション・マネージメント・ツールのオンライン・ヘルプを参照してください。

> **関連項目：** レプリケーション・マネージメント API を使用してこのタイプの競合解消方法を定義する方法については、『Oracle9*i* レプリケーション・マネージメント API リファレンス』を参照してください。

最も古いタイムスタンプ

最も古いタイムスタンプによる方法を使用すると、更新発生時のタイムスタンプによって識別された最も古い更新を原因とする競合を解消できます。

ターゲット環境 . 最も古いタイムスタンプによる競合解消方法は、1 つのマスター・サイトと任意の数のマテリアライズド・ビューから構成されるレプリケーション環境でデータが収束するように機能します。タイムスタンプは常に増加しているので、最も古いタイムスタンプによる競合解消方法では、複数のマスター・サイトがあるレプリケーション環境のデータ収束は保証できません。この方法は、2 つのトランザクションのタイムスタンプが同じである場合に備えて予備の競合解消方法を作成してある場合は、マテリアライズド・ビューの数にかかわらず機能します。

サポート・メカニズム . タイムスタンプによる方法を使用するには、レプリケート表内の DATE 型の列を指定する必要があります。アプリケーションで列グループ内の列を更新するときは、ローカル SYSDATE で指定されたタイムスタンプ列の値も更新する必要があります。他のサイトから適用される変更については、タイムスタンプ値を発行元サイトのタイムスタンプ値に設定する必要があります。5-22 ページの「タイムスタンプの設定に関する問題」を確認してください。

注意： タイムスタンプによる競合解消方法を使用するときは、2 つのサイトのタイムスタンプが同じ場合にサイト優先順位などの予備の方法がコールされるように設定する必要があります。

最も古いタイムスタンプによる方法の実装　レプリケーション・マネージメント・ツールを使用して最も古いタイムスタンプによる競合解消方法を定義する方法については、レプリケーション・マネージメント・ツールのオンライン・ヘルプを参照してください。

関連項目： レプリケーション・マネージメント API を使用してこのタイプの競合解消方法を定義する方法については、『Oracle9*i* レプリケーション・マネージメント API リファレンス』を参照してください。

最大値

Advanced Replication によって列グループに競合が検出され、**最大値**による競合解消方法がコールされると、列グループ内の指定した列について、発行元サイトの新しい値と接続先サイトの現在の値が比較されます。最大値による競合解消方法を選択する場合は、列を指定する必要があります。

指定した列の新しい値が現在の値より大きい場合は、発行元サイトの列グループの値が接続先サイトに適用されます。このとき、その行のその他のエラーはすべて解消されていると想定します。指定した列の新しい値が現在の値より小さい場合は、列グループの現在の値を変更しないかぎり競合は解消されます。

> **注意：** 指定した列の 2 つの値が同じ場合（たとえば、指定した列が競合を発生させた列ではない場合）、競合は解消されず、列グループの列の値は変更されません。この場合は、予備の競合解消方法を指定してください。

列グループ内の列のデータ型に制約はありません。複数のマスター・サイトのデータ収束が保証されるのは、列の値が常に増加している場合のみです。

> **注意：** 競合解消が制約によって妨げられる可能性があるため、CHECK 制約を使用して常に増加する値を制限しないでください。

ターゲット環境　最大値による競合解消方法を定義済みで、競合解消に使用するターゲット列の値がすべてのサイトで常に増加している場合は、マスター・サイトやマテリアライズド・ビュー・サイトの数にかかわらずデータ収束は保証されます。

サポート・メカニズム　最大値による競合解消方法では、その他のサポート・メカニズムは必要ありません。

最大値による方法の実装　レプリケーション・マネージメント・ツールを使用して最大値による競合解消方法を定義する方法については、レプリケーション・マネージメント・ツールのオンライン・ヘルプを参照してください。

> **関連項目：** レプリケーション・マネージメント API を使用してこのタイプの競合解消方法を定義する方法については、『Oracle9*i* レプリケーション・マネージメント API リファレンス』を参照してください。

最小値

Advanced Replication によって列グループに競合が検出され、**最小値**による競合解消方法がコールされると、列グループ内の指定した列について、発行元サイトの新しい値と接続先サイトの現在の値が比較されます。最小値による競合解消方法を選択する場合は、列を指定する必要があります。

指定した列の新しい値が現在の値より小さい場合は、発行元サイトの列グループの値が接続先サイトに適用されます。このとき、その行のその他のエラーはすべて解消されていると想定します。指定した列の新しい値が現在の値より大きい場合は、列グループの現在の値を変更しないかぎり競合は解消されます。

注意： 指定した列の 2 つの値が同じ場合（たとえば、指定した列が競合を発生させた列ではない場合）、競合は解消されず、列グループの列の値は変更されません。この場合は、予備の競合解消方法を指定してください。

列グループ内の列のデータ型に制約はありません。複数のマスター・サイトのデータ収束が保証されるのは、列の値が常に減少している場合のみです。

注意： 競合解消が制約によって妨げられる可能性があるため、CHECK 制約を使用して常に減少する値を制限しないでください。

ターゲット環境　最小値による競合解消方法を定義済みで、競合解消に使用するターゲット列の値がすべてのサイトで常に減少している場合は、マスター・サイトやマテリアライズド・ビュー・サイトの数にかかわらずデータ収束は保証されます。

サポート・メカニズム　最小値による競合解消方法では、その他のサポート・メカニズムは必要ありません。

最小値による方法の実装　レプリケーション・マネージメント・ツールを使用して最小値による競合解消方法を定義する方法については、レプリケーション・マネージメント・ツールのオンライン・ヘルプを参照してください。

関連項目： レプリケーション・マネージメント API を使用してこのタイプの競合解消方法を定義する方法については、『Oracle9*i* レプリケーション・マネージメント API リファレンス』の最小値による方法と最大値による方法の説明を参照してください。

優先グループ

優先グループを使用すると、特定の列に格納される可能性がある各値に優先順位レベルを割り当てることができます。競合が検出されると、「優先順位」列の値が低い表が、より高い優先順位値を持つ表のデータによって更新されます。したがって、値が大きいことは優先順位が高いことを意味します。

優先グループを使用しており、優先順位列の値が常に増加している場合は、複数のマスター・サイトでの収束が保証されます。つまり、優先順位列の値は、イベントの順序、たとえば ordered（注文）、shipped（出荷）、billed（請求）に対応します。

図 5-2 に示すように、各優先グループのメンバーに割り当てられている優先順位レベル（「優先順位」列に格納される値）は DBA_REPPRIORITY ビューに表示されます。「優先順位」列に格納される可能性があるすべての値に対して、優先順位を指定する必要があります。

図 5-2 優先グループの使用方法

customer表

custno	name	addr1	addr2	site
153	Kelly	104 First St.	Jones, NY	new_york.world
118	Klein	22 Iris Ln.	Planes, NE	houston.world
121	Lee	71 Blue Ct.	Aspen, CO	houston.world
204	Potter	181 First Av.	Aspen, CO	houston.world
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.

DBA_REPPRIORITYデータ・ディクショナリ・ビュー

...	PRIORITY_GROUP	PRIORITY	...	VARCHAR2_VALUE
	site-priority	1		houston.world
	site-priority	2		new_york.world
	order-status	1		ordered
	order-status	2		shipped
	order-status	3		billed
	...	...		...

現行の位置で定義されたすべての優先グループの値は、`DBA_REPPRIORITY` ビューに表示されます。図 5-2 の例では、site-priority と order-status という 2 つの優先グループがあります。`customer` 表は、site-priority 優先グループを使用しています。この例の order-status 優先グループでは、`billed`（優先順位値 =3）は `shipped`（優先順位値 =2）よりも優先順位が高く、`shipped` は `ordered`（優先順位値 =1）よりも優先順位が高くなっています。

レプリケーション・マネージメント・ツールを使用して優先グループによる更新の競合解消方法を選択するには、その前に表内に優先順位列を指定する必要があります。

ターゲット環境　優先グループによる競合解消方法は、ワークフロー環境用に設定されたレプリケーション環境で役に立ちます。たとえば、order-status が `shipping` 状態になると、`order entry` 部門による更新は常に上書きされます。

サポート・メカニズム　ターゲット列に含まれる値の優先順位を定義する必要があります。この優先順位を定義するのは、指定された列値の優先順位に基づいて競合を解消するためです。優先順位の定義は、優先グループに格納されます。

優先グループによる方法の実装　レプリケーション・マネージメント・ツールを使用して優先グループによる競合解消方法を定義する方法については、レプリケーション・マネージメント・ツールのオンライン・ヘルプを参照してください。

> **関連項目：** レプリケーション・マネージメント API を使用してこのタイプの競合解消方法を定義する方法については、『Oracle9*i* レプリケーション・マネージメント API リファレンス』を参照してください。

サイト優先順位

サイト優先順位は、特殊な種類の優先グループです。サイト優先順位を使用する場合、指定する優先順位列は、更新が発行されたサイトのグローバル・データベース名に自動的に更新されます。各データベース・サイトに割り当てられた優先順位レベルは、`DBA_REPPRIORITY` ビューに表示されます。

あるサイトが最も正確な情報を持っている可能性が高いとみなされる場合、サイト優先順位が役に立つことがあります。たとえば、5-30 ページの図 5-2 では、`new_york.world` サイト（優先順位値 =2）は本社で、`houston.world` サイト（優先順位値 = 1）は営業所にある更新可能マテリアライズド・ビューです。したがって、本社オフィスは、各顧客のクレジット限度額に関する最も正確な情報を持っている可能性が営業所よりも高いと考えられます。

> **注意：** `DBA_REPPRIORITY` ビューの優先グループ列には、site-priority 優先グループおよび order-status 優先グループの両方が表示されます。

サイト優先順位による方法のみを使用する場合は、複数のマスター・サイトでのデータ収束は保証されませんが、サイト優先順位はマルチマスター環境に適した予備の競合解消方法です。特に最新のタイムスタンプによる方法で 2 つのタイムスタンプがまったく同じである場合の予備の方法として役立ちます。

優先グループの場合と同様、レプリケーション・マネージメント・ツールを使用して列グループに対してサイト優先順位による競合解消方法を選択する前に、いくつかの準備を済ませておく必要があります。

ターゲット環境　優先グループと同様に、サイト優先順位による競合解消方法は、ワークフロー環境で実装されるのが一般的です。また、サイト優先順位による競合解消方法を大量配布の環境（1 つのマスター・サイトおよび任意の数のマテリアライズド・ビュー・サイトから構成される環境）で使用すると、データ収束を保証できます。

サイト優先順位による競合解消方法は、マルチマスター環境で主要な競合解消方法が失敗した場合の予備の方法としても適しています。

サポート・メカニズム　行が更新されたときにサイト情報を格納するための列を指定する必要があります。また、行が更新または挿入されたときにこのサイト列に更新中のサイトのグローバル名を移入するトリガーも作成する必要があります。このトリガーの例は、レプリケーション・マネージメント・ツールのオンライン・ヘルプ、および『Oracle9*i* レプリケーション・マネージメント API リファレンス』を参照してください。

レプリケーション環境に関係する各サイトの優先順位も定義する必要があります。この優先順位を定義するのは、更新または挿入を実行したサイトの優先順位に基づいて競合を解消するためです。サイト優先順位の定義は、優先グループに格納されます。

サイト優先順位による方法の実装　レプリケーション・マネージメント・ツールを使用してサイト優先順位による競合解消方法を定義する方法については、レプリケーション・マネージメント・ツールのオンライン・ヘルプを参照してください。

> **関連項目：**　レプリケーション・マネージメント API を使用してこのタイプの競合解消方法を定義する方法については、『Oracle9*i* レプリケーション・マネージメント API リファレンス』を参照してください。

一意性競合の解消方法

Oracle には、一意性競合を解消するビルトインの方法が 3 つ用意されています。

- 発行元サイトのグローバル・サイト名を、発行元サイトの列値に追加する方法
- 生成された順序番号を、発行元サイトの列値に追加する方法
- 発行元サイトの行値を廃棄する方法

次の項では、一意性競合解消方法について、詳しく説明します。

注意：　Oracle のビルトインの一意性競合の解消方法では、実際にはレプリケーション環境のデータは収束しません。これらの方法は、制約違反を解決するテクニックのみを提供します。Oracle の一意性競合の解消方法を使用する場合には、一意性競合の発生を知らせる通知メカニズムを同時に使用して、必要に応じて手動でレプリケート・データを収束する必要があります。

注意：　列に対する一意性競合の解消方法を追加するには、その列に対する一意索引の名前が、一意制約または主キー制約の名前と同じである必要があります。

サイト名の追加

サイト名の追加による競合解消方法では、トランザクションの発生元サイトのグローバル・データベース名が、dup_val_on_index 例外を生成しているレプリケートされた列値に追加されます。この方法を使用すると一意の整合性制約に違反せずに列を挿入または更新できますが、複数のマスター・サイト間での収束は提供されません。そのために生じた矛盾は、手動で解決する必要があります。したがって、この方法は、なんらかの通知機能とともに使用することを想定しています。

注意： サイト名の追加および順序の追加は、いずれもキャラクタ列のみに使用できます。

サイト名の追加による解消方法は、データの完全な正確さよりもデータの可用性の方が重要な場合に役立ちます。データがレプリケートされた後ですぐに使用可能な状態にするには、次のようにします。

- サイト名の追加による方法を選択します。
- 競合をロギングするのではなく、通知スキームを使用して、適切な担当者に重複を解消するように警告します。

サイト名の追加による方法では、一意性競合が発生すると、トランザクションの発行元サイトのグローバル・データベース名が、レプリケートされた列値に追加されます。名前は最初のピリオド（.）までが追加されます。たとえば、houston.world は houston になります。

順序の追加

順序の追加による競合解消方法では、生成された順序番号が、dup_val_on_index 例外を生成している列値に追加されます。この方法を使用すると一意の整合性制約に違反せずに列を挿入または更新できますが、複数のマスター・サイト間での収束は提供されません。そのために生じた矛盾は、手動で解決する必要があります。したがって、この方法は、なんらかの通知機能とともに使用することを想定しています。

注意： サイト名の追加および順序の追加は、いずれもキャラクタ列のみに使用できます。

順序の追加による解消方法は、データの完全な正確さよりもデータの可用性の方が重要な場合に役立ちます。データがレプリケートされた後ですぐに使用可能な状態にするには、次のようにします。

- 順序の追加による方法を選択します。
- 競合をロギングするのではなく、通知スキームを使用して、適切な担当者に重複を解消するように警告します。

順序の追加による方法では、生成された順序番号が列値に追加されます。列値は必要に応じて切り捨てられます。列値の生成された部分が列の長さを超えた場合、この競合解消方法ではエラーは解決されません。

廃棄

一意性の廃棄による競合解消方法では、エラーを発生させた発行元サイトの列を単に廃棄することによって一意性競合を解消します。この方法では、複数のマスター・サイト間での収束は保証されないため、通知機能とともに使用する必要があります。

追加による方法とは異なり、一意性の廃棄による方法では、データの正確さを検証できるまで、データの伝播が最小限にとどめられます。

削除の競合解消方法

Oracle では、削除の競合を解消するビルトインの方法は提供していません。5-16 ページの「削除の競合の回避」の項で説明されているように、データベースとフロントエンド・アプリケーションは削除の競合を回避するように設計する必要があります。これには、行に削除のマークを付け、プロシージャ・レプリケーションを使用して、論理的に削除された行を定期的にパージします。

関連項目：

- 削除の競合の回避方法は、5-16 ページの「削除の競合の回避」を参照してください。
- レプリケーション環境に競合の回避方法を作成する方法は、『Oracle9*i* レプリケーション・マネージメント API リファレンス』を参照してください。

パフォーマンスのメカニズムと競合の解消

行の更新の競合を検出および解消するには、伝播サイトから受信サイトに新旧の行に関するデータを送信する必要があります。環境によって、更新の競合の検出および解消をサポートするために伝播されるデータの量は異なります。

最小限の通信

すべての関連サイトが Oracle8 以上のデータベースを使用している環境でレプリケート表を作成する場合、**最小限の通信**機能を使用可能にすると、表内の変更された各行に競合が発生しているかどうかを判断するために通信する必要があるデータの量を最小化できます。最小限の通信機能を使用可能にすると、次の値が伝播されます。

- 主キーおよび列グループの各列の古い値（変更前の値）
- 列グループの更新された各列の新しい値

注意： 挿入された行には、古い値はありません。また、削除された行には、新しい値はありません。

注意： ネストした表の記憶表は、独立した列グループとして扱われます。

一般的に、Oracle8 以上のレプリケーション環境では、データ伝播を最小化して、ネットワーク上で送信されるデータの量を減らすことが可能です。それにより、システム全体のパフォーマンスが向上します。

これに対して、レプリケーション環境に Oracle7 のサイトと Oracle8 以上のサイトが混在している場合は、行データの通信を最小化して更新の競合を解消することはできません。この場合、競合を解消するには、変更された各行の新旧の値をすべて伝播する必要があります。

シングル・マスター・サイト環境での静止の減少とマテリアライズド・ビュー シングル・マスター・サイト環境で、すべての更新可能マテリアライズド・ビューが競合の検出に最小限の通信を使用しない場合は、マスター・グループに対する特定の管理操作で静止が不要になります。

関連項目： これらの操作については、このマニュアルの最初の方にある「Advanced Replication の新機能」の「最小限の通信を使用しないシングル・マスター環境での可用性の拡張」にリストされています。

少なくとも 1 つの更新可能マテリアライズド・ビューで、競合の検出に最小限の通信を使用するとき、次のような場合には静止が不要です。

- マスター・グループへの管理操作を実行する前に、最小限の通信を使用する更新可能マテリアライズド・ビューを含むデータベースで遅延トランザクションをすべて伝播する場合。
- マスター・サイトに対する管理操作を完了し、最小限の通信を使用する更新可能マテリアライズド・ビューのレプリケーション・サポートを再生成するまで、そのマテリアライズド・ビューに対してデータ操作言語（DML）による変更を許可しない場合。

これらのアクションを実行しない場合は、マスター・グループに管理操作を実行する前にマスター・グループを静止します。

古い値の送信および比較

前述の方法でデータ伝播を最小化している場合、DBMS_REPCAT.SEND_OLD_VALUES プロシージャおよび DBMS_REPCAT.COMPARE_OLD_VALUES プロシージャを使用することで、データ伝播をさらに削減できる場合があります。これらのプロシージャを使用すると、競合の検出および解消に必要な場合に限って古い値が送信されます。たとえば、最新のタイムスタンプによる競合の検出および解消方法には、非キー列および非タイムスタンプ列の古い値が必要ありません。

提案： LOB データ型をレプリケートしていて、これらの列で競合が発生しないと考えられる場合は、古い値の伝播をさらに最小化することが特に有効です。

注意： 予想される競合を検出および解消できるように、適切な古い値を伝播する必要があります。ユーザー指定の競合解消プロシージャでは、送信される古い列値 NULL を適切に処理できる必要があります。データ伝播をさらに削減するために SEND_OLD_VALUES プロシージャおよび COMPARE_OLD_VALUES プロシージャを使用すると、予期しない競合に対する保護が低下します。

データ伝播をさらに削減するには、次のプロシージャを実行します。

```
DBMS_REPCAT.SEND_OLD_VALUES(
   sname           IN  VARCHAR2,
   oname           IN  VARCHAR2,
   { column_list   IN  VARCHAR2,
   | column_table  IN  DBMS_UTILITY.VARCHAR2s | DBMS_UTILITY.LNAME_ARRAY,}
   operation       IN  VARCHAR2 := 'UPDATE',
   send            IN  BOOLEAN  := true );

DBMS_REPCAT.COMPARE_OLD_VALUES(
   sname           IN  VARCHAR2,
   oname           IN  VARCHAR2,
   { column_list   IN  VARCHAR2,
   | column_table  IN  DBMS_UTILITY.VARCHAR2s | DBMS_UTILITY.LNAME_ARRAY,}
   operation       IN  VARCHAR2 := 'UPDATE',
   compare         IN  BOOLEAN := true );
```

これらのプロシージャを実行した後で、変更を有効にするために、DBMS_REPCAT.GENERATE_REPLICATION_SUPPORT プロシージャで min_communication を TRUE に設定してレプリケーション・サポートを生成する必要があります。

注意： operation パラメータを使用すると、行が削除されたときまたは非キー列が更新されたとき（あるいはその両方）に、非キー列の古い値を送信するかどうかを決定できます。古い値を送信しない場合、古い値のかわりに NULL が送信され、更新または削除が適用されるときに古い値とターゲット・サイドの列の現在の値が等しいとみなされます。

古い列値に対して指定した動作は、DBA_REPCOLUMN データ・ディクショナリ・ビューの 2 つの列、COMPARE_OLD_ON_DELETE（Y または N）および COMPARE_OLD_ON_UPDATE（Y または N）に公開されます。

次の例では、これらのプロシージャを使用してデータ伝播をさらに削減する方法を示します。列が 3 つある rsmith.reports という表を考えてみます。列 1 は主キーであり、独自の列グループ（列グループ 1）を構成します。列 2 および列 3 は、2 番目の列グループ（列グループ 2）に属します。

図 5-3　列グループとデータ伝播

2 番目の列グループの競合解消方針は、サイト優先順位です。列 2 は、サイト名を含む VARCHAR2 型の列です。列 3 は、LOB 列です。LOB を更新するときは必ず、更新が行われたサイトのグローバル名で列 2 も更新する必要があります。LOB に対するピース単位の更新にはトリガーがないため、LOB に対してピース単位の更新を行う場合は常に列 2 を明示的に更新する必要があります。

DBMS_REPCAT.GENERATE_REPLICATION_SUPPORT プロシージャで min_communication を TRUE に設定して rsmith.reports のレプリケーション・サポートを生成し、次に UPDATE 文を使用して列 2（サイト名）と列 3（LOB）を変更するとします。サイト名の新しい値と LOB の新しい値が更新されているので、遅延リモート・プロシージャ・コール（RPC）にはこれらの値が含まれます。遅延 RPC には、主キー（列 1）の古い値、サイト名（列 2）の古い値および LOB（列 3）の古い値も含まれます。

注意： 競合の検出および解消には、LOB の古い値は必要ありません。競合の検出と解消に必要なのは、列 C2（サイト名）のみです。LOB の古い値を送信すると、伝播時間が大幅に増加する可能性があります。

列 C2 または列 C3 が更新されたときに LOB の古い値を伝播しない場合は、次のコールを実行します。

```
BEGIN
  DBMS_REPCAT.SEND_OLD_VALUES(
     sname           =>  'rsmith',
     oname           =>  'reports'
     column_list     =>  'c3',
     operation       =>  'UPDATE',
     send            =>  FALSE );
END;
/

BEGIN
  DBMS_REPCAT.COMPARE_OLD_VALUES(
     sname           =>  'rsmith',
     oname           =>  'reports'
     column_list     =>  'c3',
     operation       =>  'UPDATE',
     compare         =>  FALSE);
END;
/
```

変更を有効にするには、`DBMS_REPCAT.GENERATE_REPLICATION_SUPPORT` プロシージャで `min_communication` を `TRUE` に設定して `rsmith.reports` のレプリケーション・サポートを生成する必要があります。続いて `UPDATE` 文を使用して列 2（サイト名）と列 3（LOB）を変更するとします。遅延 RPC には、主キー（列 1）の古い値、サイト名（列 2）の古い値と新しい値、および LOB（列 3）の新しい値が含まれます。遅延 RPC は、主キーの新しい値と LOB の古い値については NULL を含みます。

注意： Oracle の競合解消では、LOB のピース単位の更新はサポートしていません。

列オブジェクト使用時の送信および比較 リーフ属性がレプリケーション・キー列でない場合は、古い値を送信および比較するときに、列オブジェクトのリーフ属性を指定できます。たとえば、次の cust_address_typ オブジェクト型を作成するとします。

```
CREATE TYPE cust_address_typ AS OBJECT
     (street_address     VARCHAR2(40),
      postal_code        VARCHAR2(10),
      city               VARCHAR2(30),
      state_province     VARCHAR2(10),
      country_id         CHAR(2));
```

この型を列オブジェクトとして使用し、customers 表を作成します。

```
CREATE TABLE customers
     (customer_id        NUMBER(6),
      cust_first_name    VARCHAR2(20),
      cust_last_name     VARCHAR2(20),
      cust_address       cust_address_typ,
      phone_numbers      phone_list_typ);
```

customers 表の cust_address_typ の street_address 属性の古い値を送信および比較する場合は、次のプロシージャを使用して、属性値を送信および比較するように指定します。

```
BEGIN
  DBMS_REPCAT.SEND_OLD_VALUES(
     sname           =>  'oe',
     oname           =>  'customers',
     column_list     =>  'cust_address.street_address', -- object attribute
     operation       =>  'UPDATE',
     send            =>   true );
END;
/

BEGIN
  DBMS_REPCAT.COMPARE_OLD_VALUES(
     sname           =>  'oe',
     oname           =>  'customers',
     column_list     =>  'cust_address.street_address', -- object attribute
     operation       =>  'UPDATE',
     compare         =>   true);
END;
/
```

注意： 1 つの列オブジェクトにオブジェクト属性が複数レベルある場合、column_list パラメータに指定できるのは最後の属性（またはリーフ属性）のみです。中間の属性は指定できません。

列オブジェクト全体を送信および比較することも指定できます。たとえば、次のプロシージャでは、cust_address 列オブジェクト全体を指定しています。

```
BEGIN
  DBMS_REPCAT.SEND_OLD_VALUES(
     sname           =>  'oe',
     oname           =>  'customers',
     column_list     =>  'cust_address', -- entire column object
     operation       =>  'UPDATE',
     send            =>   true );
END;
/

BEGIN
  DBMS_REPCAT.COMPARE_OLD_VALUES(
     sname           =>  'oe',
     oname           =>  'customers',
     column_list     =>  'cust_address', -- entire column object
     operation       =>  'UPDATE',
     compare         =>   true);
END;
/
```

関連項目： DBMS_REPCAT.SEND_OLD_VALUES プロシージャおよび DBMS_REPCAT.COMPARE_OLD_VALUES プロシージャの詳細は、『Oracle9*i* レプリケーション・マネージメント API リファレンス』を参照してください。

6

レプリケーション環境の計画

レプリケーション環境の計画を開始する前に、これまでの章で説明したレプリケーションの概念およびアーキテクチャについて理解することが重要です。この章では、レプリケーションの概念およびアーキテクチャについて理解していることを前提に、レプリケーション環境の計画に関する重要な考慮事項について説明します。

この章の内容は、次のとおりです。

- レプリケート表に関する考慮事項
- 初期化パラメータ
- マスター・サイトとマテリアライズド・ビュー・サイトの比較
- スケジュール・リンクのガイドライン
- 遅延トランザクション・キューのスケジュール・パージのガイドライン
- シリアル伝播およびパラレル伝播
- 配置テンプレート
- 競合解消
- セキュリティ
- 耐障害性を実現するための設計

レプリケート表に関する考慮事項

次の項では、レプリケーション環境で使用する表に関する考慮事項について説明します。

主キー

各レプリケート表には、可能なかぎり主キーを設定します。主キーを設定できない場合、各レプリケート表に、表の各行の一意の識別子として使用できる列のセットが含まれている必要があります。レプリケーション環境で使用する表に主キーまたは一意の列のセットがない場合は、レプリケート表を必要に応じて修正します。さらに、マスター表またはマスター・マテリアライズド・ビューに基づいて主キー・マテリアライズド・ビューを作成する場合は、そのマスターに主キーを設定する必要があります。

外部キー

外部キー参照制約を持つ表をレプリケートする場合は、親表で更新および削除が許可されていない場合を除き、常に外部キー列に索引を付けて、これらの索引をレプリケートすることをお薦めします。索引は自動的にはレプリケートされません。索引をレプリケートするには、レプリケーション・マネージメント・ツールまたは DBMS_REPCAT パッケージ内の CREATE_MASTER_REPOBJECT プロシージャを使用して、索引を付けた表を含むマスター・グループに索引を追加します。

データ型に関する考慮事項

Advanced Replication では、次のデータ型の列を持つ表およびマテリアライズド・ビューのレプリケーションをサポートしています。

- VARCHAR2
- NVARCHAR2
- NUMBER
- DATE
- TIMESTAMP
- TIMESTAMP WITH TIME ZONE
- TIMESTAMP LOCAL TIME ZONE
- INTERVAL YEAR TO MONTH
- INTERVAL DAY TO SECOND
- RAW
- ROWID
- CHAR
- NCHAR

- ユーザー定義データ型

また、次のラージ・オブジェクト（Large Object: LOB）型の列を持つ表およびマテリアライズド・ビューのレプリケーションもサポートしています。

- バイナリ・ラージ・オブジェクト（Binary Large Object: BLOB）
- キャラクタ・ラージ・オブジェクト（Character Large Object: CLOB）
- 各国語キャラクタ・ラージ・オブジェクト（National Character Large Object: NCLOB）

ピース単位の更新と追加がこれらの LOB 列に適用されると、マルチマスター・レプリケーションに使用する遅延および同期のリモート・プロシージャ・コール・メカニズムによって、サポートされている LOB データ型にピース単位の変更のみが伝播されます。ただし、マテリアライズド・ビューの定義問合せの WHERE 句で LOB 列は参照できません。

ユーザー定義型（列オブジェクト、オブジェクト表、REF、VARRAY およびネストした表など）を使用する表およびマテリアライズド・ビューはレプリケートできます。

Oracle では、LONG データ型および LONG RAW データ型を使用する列のレプリケーションはサポートしていません。これらのデータ型を含む列は、レプリケート表から単純に削除されます。LONG データ型は LOB に変換する必要があります。

外部 LOB またはファイル・ベースの LOB（BFILE）のレプリケーションもサポートされていません。このデータ型の列を含む表をマスター表として構成すると、エラー・メッセージが戻されます。

マスター表または更新可能マテリアライズド・ビュー内の UROWID 列のレプリケーションもサポートされていません。ただし、読取り専用マテリアライズド・ビューでは UROWID 列を使用できます。

注意： Oracle では、一部のサイトで Oracle7 リリース 7.3 が稼働しているレプリケーション環境での LOB データ型のレプリケーションをサポートしていません。

関連項目：

- レプリケート表での LONG 列から LOB 列への変換の詳細は、『Oracle9*i* レプリケーション・マネージメント API リファレンス』を参照してください。
- データ型の詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

行レベルでの依存性の追跡

表を作成するときは、システム変更番号（SCN）を追跡するために次のオプションを指定できます。

- NOROWDEPENDENCIES。SCN がデータ・ブロック・レベルで追跡されることを指定します（デフォルト）。
- ROWDEPENDENCIES。SCN が表の各行に対して追跡されることを指定します。

ROWDEPENDENCIES オプションを使用すると、パラレル伝播使用時のパフォーマンスと拡張性が向上しますが、このオプションでは、各行に 6 バイトの追加記憶領域が必要になります。

次の SQL 文では、ROWDEPENDENCIES オプションを使用して表を作成します。

```
CREATE TABLE order_items
  (order_id          NUMBER(12),
  line_item_id       NUMBER(3)  NOT NULL,
  product_id         NUMBER(6)  NOT NULL,
  unit_price         NUMBER(8,2),
  quantity           NUMBER(8)
  ) ROWDEPENDENCIES;
```

この order_items 表の各行で SCN が追跡されます。表がクラスタの一部である場合は、CREATE CLUSTER 文で ROWDEPENDENCIES オプションを使用することもできます。

> **関連項目：** ROWDEPENDENCIES オプションの詳細は、2-50 ページの「データ伝播の依存性の維持」を参照してください。

初期化パラメータ

表 6-1 に、レプリケーション環境の操作性、信頼性およびパフォーマンスにとって重要な初期化パラメータをリストしています。

> **関連項目：** これらの初期化パラメータの詳細は、『Oracle9*i* データベース・リファレンス』を参照してください。

表 6-1 レプリケーションにおいて重要な初期化パラメータ

パラメータ	値	説明	推奨事項
COMPATIBLE	**デフォルト：** 8.1.0 **範囲：** 8.1.0 ～現行のリリース番号	新しいリリースが使用でき、また以前のリリースとの下位互換性も保証されます。これは、以前のリリースを使用する必要が生じた場合に便利です。 このパラメータには、Oracle サーバーが互換性を保つ必要のあるリリースを指定します。互換性のレベルの異なる複数の Oracle サーバーは相互に運用できます。	Oracle9*i* で使用可能な新しいレプリケーション機能を使用する場合は、このパラメータを 9.0.1 以上に設定します。
GLOBAL_NAMES	**デフォルト：** FALSE **範囲：** TRUE または FALSE	データベース・リンクの名前を接続先データベースの名前と同じにするかどうかを指定します。	GLOBAL_NAMES は、マスター・サイトおよびマテリアライズド・ビュー・サイトなどのレプリケーション環境に含まれる各データベースで TRUE に設定する必要があります。
JOB_QUEUE_PROCESSES	**デフォルト：** 0 **範囲：** 0 ～ 1000	各インスタンスの J*n* ジョブ・キュー・プロセスの番号を指定します（J000 ...J999）。ジョブ・キュー・プロセスでは、DBMS_JOB によって作成された要求が処理されます。 サイトで JOB_QUEUE_PROCESSES を 0 に設定した場合、サイトのすべてのグループに対して手動で管理要求を適用し、遅延トランザクション・キューを手動でプッシュおよびパージする必要があります。 JOB_QUEUE_PROCESSES の設定は、ALTER SYSTEM 文を使用して動的に変更できます。	このパラメータは 1 以上であり、かつ同時に実行できるジョブの最大数に 1 を加えた値と同じ値に設定する必要があります。

表 6-1 レプリケーションにおいて重要な初期化パラメータ（続き）

パラメータ	値	説明	推奨事項
OPEN_LINKS	**デフォルト：** 4 **範囲：** 0 ～ 255	1 つのセッションで同時にオープンするリモート・データベース接続の最大数を指定します。この接続には、データベース・リンクと呼ばれるスキーマ・オブジェクトと、外部プロシージャおよびカートリッジ（それぞれ別のプロセスを使用します）が含まれます。	同期レプリケーションを使用する場合は、OPEN_LINKS を少なくともマスター・サイト数と同じ数に設定する必要があります。たとえば、マスター・サイトが 5 つある環境では、OPEN_LINKS を少なくとも 5 に設定する必要があります。
PARALLEL_AUTOMATIC_TUNING	**デフォルト：** FALSE **範囲：** TRUE または FALSE	TRUE に設定されている場合、パラレル実行を制御する初期化パラメータのデフォルト値が決定されます。	初期化パラメータの使用方法の詳細は、『Oracle9*i* データベース・リファレンス』および『Oracle9*i* データベース移行ガイド』を参照してください。
PARALLEL_MAX_SERVERS	**デフォルト：** 次のパラメータの値を使用します。 CPU_COUNT PARALLEL_AUTOMATIC_TUNING PARALLEL_ADAPTIVE_MULTI_USER **範囲：** 0 ～ 3599	インスタンスのパラレル実行プロセスおよびパラレル・リカバリ・プロセスの最大数を指定します。インスタンス起動時に作成されるプロセスの数は、要求が増えるに従い、このパラメータに指定した値まで増えます。	パラレル伝播を使用する場合は、パラレル伝播をサポートするのに十分な大きさの値を指定してください。
PARALLEL_MIN_SERVERS	**デフォルト：** 0 **範囲：** 0 ～ PARALLEL_MAX_SERVERS の値	インスタンスのパラレル実行プロセスの最小数を指定します。この値は、インスタンスの起動時に作成されるパラレル実行プロセスの数です。	パラレル伝播を使用する場合、各ストリームに少なくとも 1 つのプロセスがあることを確認してください。
PROCESSES	**デフォルト：** PARALLEL_MAX_SERVERS から導出 **範囲：** 6 ～オペレーティング・システム固有の上限	Oracle に同時に接続できる、オペレーティング・システムのユーザー・プロセスの最大数を指定します。	このパラメータの値は、ロック、ジョブ・キュー・プロセスおよびパラレル実行プロセスなどのすべてのバックグラウンド・プロセスを使用できる値にする必要があります。

表 6-1 レプリケーションにおいて重要な初期化パラメータ（続き）

パラメータ	値	説明	推奨事項
REPLICATION_DEPENDENCY_TRACKING	**デフォルト：** TRUE **範囲：** TRUE または FALSE	データベースに対する読取り / 書込み操作の依存性の追跡を使用可能または使用禁止にします。依存性の追跡は、レプリケーション環境で変更をパラレル伝播するときは必須です。 TRUE: 依存性の追跡を使用可能にします。 FALSE: データベースに対する読取り / 書込み操作の実行を高速化しますが、パラレル伝播を実行するための依存性情報は生成しません。	通常は TRUE を指定します。アプリケーションがレプリケート表に読取り / 書込み操作を実行しないことが確認できない場合は、FALSE を指定しないでください。
SHARED_POOL_SIZE	**デフォルト：** 16 MB （64 ビットの場合は 64MB） **範囲：** 300 KB ～オペレーティング・システム固有の上限	共有プールのサイズをバイト単位で指定します。共有プールには、共有カーソル、ストアド・プロシージャ、制御構造およびその他の構造が含まれます。この値を大きくするほど、マルチユーザー・システムのパフォーマンスは向上します。この値を小さくすると、使用するメモリー量は減ります。	一般的に、Oracle サーバーの共有プールは、非レプリケーション環境よりもレプリケーション環境の場合のほうを大きくする必要があります。 共有プールの使用率は、ビュー V$SGASTAT を問い合せることにより監視できます。
UTL_FILE_DIR	**デフォルト：** なし **範囲：** 有効な任意のディレクトリ・パス	生成されたファイルのデフォルトの場所を指定します。これらのファイルは、レプリケーション・マネージメント・ツールまたはレプリケーション・マネージメント API により生成されるファイルです。 たとえば、配置テンプレート用のオフライン・インスタンシエーション・ファイルを、この初期化パラメータで指定されたディレクトリに入れられます。	有効なディレクトリを指定します。

マスター・サイトとマテリアライズド・ビュー・サイトの比較

レプリケーション環境を計画するときは、レプリケーション環境に関係する各サイトをマスター・サイトにするかマテリアライズド・ビュー・サイトにするかを決定する必要があります。これを決定する際は、両方のタイプのレプリケーション・サイトの特性と利点を考慮してください。1 つのレプリケーション環境で、マスター・サイトとマテリアライズド・ビュー・サイトの両方をサポートできます。

表 6-2 マスター・サイトとマテリアライズド・ビュー・サイトの特性

マスター・サイト	マテリアライズド・ビュー・サイト
■ 通常、少数のマスター・サイトと通信しますが、多数のマテリアライズド・ビュー・サイトとも通信する場合もあります。	■ 1 つのマスター・サイトまたは 1 つのマスター・マテリアライズド・ビュー・サイトと通信します。
■ 他のマスター・サイトのデータの完全なコピーである大量のデータを含んでいます。	■ マスター・サイトまたはマスター・マテリアライズド・ビュー・サイトのデータのサブセットである少量のデータを含んでいます。
■ 通常、データ伝播の間隔は短く、継続的に通信します。	■ バルク・データ移送の間隔は長く、定期的に通信します。

注意： マスター・サイトには Oracle Enterprise Edition または Oracle Standard Edition が必要です。Oracle Standard Edition は、シングル・マスター・サイト構成のみをサポートしています。Oracle を実行するマテリアライズド・ビュー・サイトには、Oracle Enterprise Edition、Oracle Standard Edition または Oracle Personal Edition のいずれかが必要です。また、レプリケーションは、すべてのマスター・サイトおよびマテリアライズド・ビュー・サイトにインストールする必要があります。

マスター・サイトの利点

マスター・サイトには次のような利点があります。

- リモート・サイトからのデータ・アクセスの可用性を高めるサポートを提供します。
- 変更が自動的にかつ通常は短い間隔で伝播されるので、データ操作言語（DML）による変更が頻繁に行われても適切に対処できます。
- 表をロックせずに、DML による変更とデータ伝播を同時に行うことができます。
- フェイルオーバーによる保護を提供できます。

マスター・サイトの設定には、レプリケーション・マネージメント・ツールの「Setup Wizard」またはレプリケーション・マネージメント API のいずれかを使用します。

関連項目：

- 「Setup Wizard」を使用してマスター・サイトを設定する方法の詳細は、レプリケーション・マネージメント・ツールのオンライン・ヘルプを参照してください。
- レプリケーション・マネージメント API を使用してマスター・サイトを設定する方法の詳細は、『Oracle9*i* レプリケーション・マネージメント API リファレンス』を参照してください。
- フェイルオーバーによる保護を使用したレプリケーション環境の設計の詳細は、6-32 ページの「耐障害性を実現するための設計」を参照してください。

マテリアライズド・ビュー・サイトの利点

マテリアライズド・ビュー・サイトには次のような利点があります。

- モバイル・コンピューティングをサポートしています。
- マスター・サイトまたはマスター・マテリアライズド・ビュー・サイトのデータのサブセットを含められます。

マテリアライズド・ビュー・サイトの設定には、レプリケーション・マネージメント・ツールの「Setup Wizard」またはレプリケーション・マネージメント API のいずれかを使用できます。

関連項目：

- 「Setup Wizard」を使用してマテリアライズド・ビュー・サイトを設定する方法の詳細は、レプリケーション・マネージメント・ツールのオンライン・ヘルプを参照してください。
- レプリケーション・マネージメント API を使用してマテリアライズド・ビュー・サイトを設定する方法の詳細は、『Oracle9*i* レプリケーション・マネージメント API リファレンス』を参照してください。

マテリアライズド・ビューの準備

マテリアライズド・ビュー・レプリケーションに関する問題の多くは、環境の準備が正しく行われていないことによるものです。マテリアライズド・ビュー環境を作成するには、事前に次の 4 つのことを行う必要があります。

- 必要なスキーマの作成
- 必要なデータベース・リンクの作成
- 適切な権限の割当て
- 十分なジョブ・プロセスの割当て

レプリケーション・マネージメント・ツールの「Setup Wizard」は、これらの作業を自動的に実行します。以降では、レプリケーション環境の理解とレプリケーション・マネージメント API の使用に役立つ情報を提供します。「Setup Wizard」を実行した後で、必要なマテリアライズド・ビュー・ログを作成します。「Setup Wizard」を使用してマテリアライズド・ビュー・サイトを設定する方法の詳細は、レプリケーション・マネージメント・ツールのオンライン・ヘルプを参照してください。

関連項目： 6-16 ページ「マテリアライズド・ビュー・ログの作成」

レプリケーション・マネージメント・ツールを使用できない場合は、『Oracle9*i* レプリケーション・マネージメント API リファレンス』の第 2 章の「マテリアライズド・ビュー・サイトの設定」の項の、レプリケーション・マネージメント API を使用したマテリアライズド・ビュー・サイトの設定方法の詳細を参照してください。

次の項では、レプリケーション・マネージメント・ツールの「Setup Wizard」または『Oracle9*i* レプリケーション・マネージメント API リファレンス』に記述されているスクリプトを使用してマテリアライズド・ビュー・サイトを設定する方法について説明します。

マテリアライズド・ビュー・サイトのユーザーの作成

各マテリアライズド・ビューには、マテリアライズド・ビュー・サイトで管理アクティビティやリフレッシュ・アクティビティを実行する何人かのユーザーが必要です。マテリアライズド・ビューの管理者とリフレッシャには、必要な権限を作成して付与する必要があります。

マスター・サイト・ユーザーの作成

マテリアライズド・ビュー・サイトのユーザーのかわりに作業を実行するには、ターゲット・マスター・サイトにも同等のプロキシ・ユーザーが必要です。通常は、プロキシ・マテリアライズド・ビュー管理者とプロキシ・リフレッシャを作成します。

マテリアライズド・ビュー・サイトでのスキーマの作成

リモート・データベースのマテリアライズド・ビューを含むスキーマは、マスター・データベースのマスター表を含むスキーマに対応している必要があります。したがって、マテリアライズド・ビューを使用してレプリケートするマスター表を含むスキーマを識別してください。マスター・データベースのターゲット・スキーマを識別した後、リモート・データベースで、対応するアカウントを同じ名前で作成します。たとえば、すべてのマスター表が `ny.world` データベースの `sales` スキーマ内にある場合は、対応する `sales` スキーマをマテリアライズド・ビュー・データベース `sf.world` に作成します。

> **関連項目：**『Oracle9*i* レプリケーション・マネージメント API リファレンス』に記載されている手順では、マテリアライズド・ビュー・グループを作成する指示の中で説明されているスクリプトの一部として、必要なスキーマが作成されます。

データベース・リンクの作成

マテリアライズド・ビューの定義問合せでは、1 つ以上のデータベース・リンクを使用してリモート表のデータを参照します。マテリアライズド・ビューを作成するには、使用する予定のデータベース・リンクが使用可能になっている必要があります。さらに、リモート・データベースにアクセスするためにデータベース・リンクで使用するアカウントには、セキュリティ・コンテキストが定義されます。そのコンテキストに基づいてマテリアライズド・ビューが作成され、リフレッシュが実行されます。

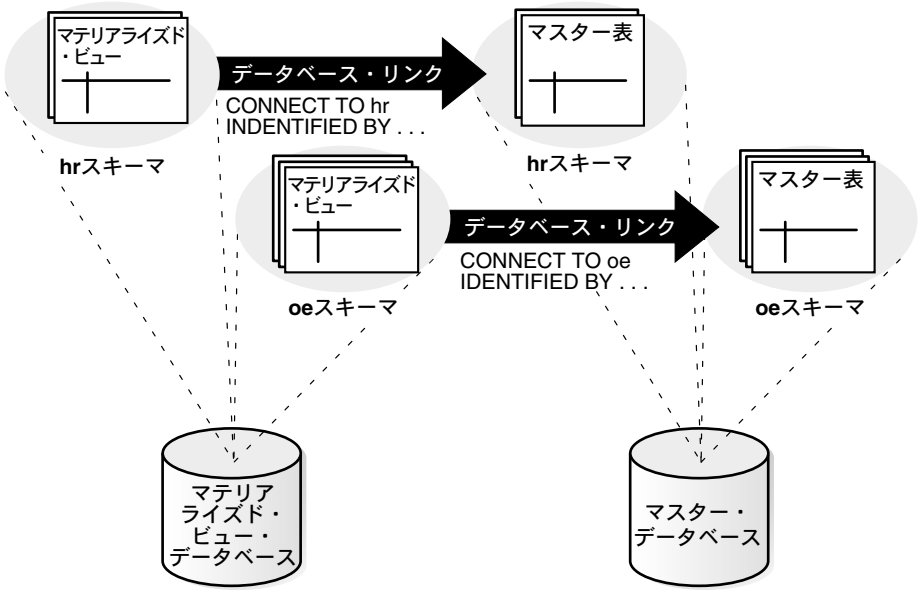
正常な動作を保証するために、マテリアライズド・ビューの定義問合せでは、ユーザー名とパスワードが定義に埋め込まれているデータベース・リンクを使用する必要があります。マテリアライズド・ビューを作成するときは、パブリック・データベース・リンクを使用できません。ユーザー名とパスワードが埋め込まれているデータベース・リンクは常に、指定されたアカウントを使用してリモート・データベースとの接続を確立します。また、リンクで使用するリモート・アカウントには、マテリアライズド・ビューの定義問合せの中で参照されるデータにアクセスするために必要な `SELECT` 権限が必要です。

マテリアライズド・ビューを作成するには、いくつかの管理データベース・リンクを作成する必要があります。具体的には、マテリアライズド・ビュー・サイトからマスター・サイトへの `PUBLIC` データベース・リンクを作成します。これを作成すると、各プライベート・データベース・リンク内に `USING` 句を含める必要がなくなるため、プライベート・データベース・リンクを定義しやすくなります。また、マテリアライズド・ビュー管理者からプロキシ管理者へのプライベート・データベース・リンクおよびプロパゲータから受信者へのプライベート・データベース・リンクも作成する必要があります。ただし、これらのデータベース・リンクは、レプリケーション・マネージメント・ツールの「Setup Wizard」を使用すると自動的に作成されます。

関連項目： 詳細は、『Oracle9*i* レプリケーション・マネージメント API リファレンス』に説明されているセキュリティ・オプションを参照してください。

管理データベース・リンクを作成した後、マテリアライズド・ビュー・データベースの各レプリケート・マテリアライズド・ビュー・スキーマをマスター・データベースの対応するスキーマに接続するプライベート・データベース・リンクを作成する必要があります。マテリアライズド・ビュー・データベースの各プライベート・データベース・リンクには、対応付けられたマスター・データベースのアカウント情報を埋め込むことが必要です。たとえば、マテリアライズド・ビュー・データベースの hr スキーマには、hr というユーザー名とパスワードを使用してマスター・データベースに接続するプライベート・データベース・リンクが必要です。

図 6-1 推奨されるスキーマおよびデータベース・リンク構成



関連項目： データベース・リンクの詳細は、『Oracle9*i* データベース管理者ガイド』を参照してください。

権限の割当て

マテリアライズド・ビューの作成者と所有者は、ともにマテリアライズド・ビューの定義 SELECT 文を発行できる必要があります。マテリアライズド・ビューの所有者とは、マテリアライズド・ビューが含まれているスキーマのことです。レプリケーション管理者またはマテリアライズド・ビュー管理者以外のユーザーがマテリアライズド・ビューを作成する場合は、そのユーザーに CREATE MATERIALIZED VIEW 権限および定義 SELECT 文を実行する適切な SELECT 権限が付与されている必要があります。

> **関連項目：**『Oracle9*i* レプリケーション・マネージメント API リファレンス』に記載されている手順では、マテリアライズド・ビュー・グループを作成する指示の中で説明されているスクリプトの一部として、必要な権限が付与されます。権限要件は、3-14 ページの「マテリアライズド・ビューの操作に必要な権限」にも説明されています。

マスター・サイトでのスケジュール・パージ

遅延トランザクション・キューのサイズを適正に保つために、パージ操作をスケジュールして、正常に送信されたすべての遅延トランザクションを遅延トランザクション・キューから削除します。この操作はマスター・サイトですでに実行済みの場合があります。パージ操作を再びスケジュールしてもマスター・サイトに悪影響はありませんが、パージ・スケージュールの特性が変更される可能性があります。

スケジュール・プッシュ

マテリアライズド・ビュー・サイトでプッシュをスケジュールすると、データベース・リンクを使用して、マテリアライズド・ビュー・サイトの遅延トランザクションが対応するターゲット・マスター・サイトに自動的に伝播されます。このタイプのデータベース・リンクは、スケジュール・リンクと呼ばれます。マテリアライズド・ビュー・グループは 1 つのターゲット・マスター・サイトのみを持つので、通常、マテリアライズド・ビュー・サイトの各マテリアライズド・ビュー・グループには 1 つのスケジュール・リンクのみが存在します。

ジョブ・キュー・プロセスの割当て

レプリケーションの自動化を処理するために、十分なジョブ・キュー・プロセスを割り当てることが重要です。ジョブ・キュー・プロセスは、遅延トランザクション・キューの伝播、遅延トランザクション・キューのパージ、マテリアライズド・ビューのリフレッシュなどを自動的に行います。

マルチマスター・レプリケーションの場合、各マスター・サイト間にスケジュール・リンクがあります。たとえば、マスター・サイトが 6 つある場合、各サイトには他の 5 つのサイトへのスケジュール・リンクがあります。通常、スケジュール・リンクごとにプロセスが 1 つ必要です。遅延トランザクション・キューのパージやその他のユーザー定義ジョブのために、追加のジョブ・プロセスが必要になります。

マテリアライズド・ビュー・レプリケーションの特性として、マテリアライズド・ビュー・サイトには、通常、マスター・データベースへのスケジュール・リンクが 1 つあり、少なくとも 1 つのジョブ・プロセスが必要です。マテリアライズド・ビュー・サイトで必要なジョブ・プロセスの数は、パージ・スケジュール、ユーザー定義ジョブおよびスケジュール・リンクに依存しますが、通常は 1 つから 3 つを必要とします。また、それぞれの並列度に対して少なくとも 1 つのジョブ・キュー・プロセスが必要です。

ユーザーがアプリケーション・インタフェースを使用してマテリアライズド・ビューを手動でリフレッシュする場合は、スケジュール・リンクを作成する必要がなくなり、マテリアライズド・ビュー・サイトで必要なジョブ・プロセスが 1 つ減ります。

ジョブ・キュー・プロセスは、データベースの初期化パラメータ・ファイルにある JOB_QUEUE_PROCESSES 初期化パラメータを使用して定義されます。ジョブ・キュー・プロセスは、次のいずれかの方法で設定できます。

- JOB_QUEUE_PROCESSES 設定の動的な変更
- インスタンス管理ツールを使用した初期化パラメータの編集
- 初期化パラメータ・ファイルの手動による編集

ジョブ・キュー・プロセスの間隔は自動的に判断されます。つまり、ジョブ・キュー・プロセスが起動してジョブを実行するタイミングは Oracle が判断します。

> **関連項目：** JOB_QUEUE_PROCESSES の詳細は、6-4 ページの「初期化パラメータ」および『Oracle9*i* データベース・リファレンス』を参照してください。

JOB_QUEUE_PROCESSES 設定の動的な変更 JOB_QUEUE_PROCESSES 初期化パラメータは動的です。したがって、インスタンスの実行中に、ALTER SYSTEM コマンドを使用して変更できます。

次の文は、動的に JOB_QUEUE_PROCESSES を 50 に設定します。

```
ALTER SYSTEM SET JOB_QUEUE_PROCESSES = 50;
```

ALTER SYSTEM コマンドを使用して動的パラメータを変更するたびに、パラメータを変更するコマンドがアラート・ログに記録されます。前述の例の ALTER SYSTEM コマンドは、インスタンス内の全セッションで、このパラメータのグローバル値をインスタンスの継続時間中（データベースがシャットダウンされるまで）変更します。データベースをシャットダウンした後再起動した場合は、初期化パラメータ・ファイル内の JOB_QUEUE_PROCESSES の設定が使用されます。

インスタンス管理ツールを使用した初期化パラメータの編集 マテリアライズド・ビュー・サイトへの専用ネットワーク・リンクがある場合や、ネットワーク・リンクをスケジュールできる場合は、インスタンス管理ツールの「Edit Database」ダイアログ・ボックスを使用してマテリアライズド・ビュー・サイトのジョブ・キュー・プロセスを設定できます。インスタンス管理ツールはマテリアライズド・ビュー・サイトにインストールされることはほとんどなく、設定はマスター・サイトからリモートで行う必要があるため、この操作が必要になります。リモートで設定できない場合は、次の項を参照してください。

Instance Manager を使用してジョブ・プロセスを設定するには、次の操作を実行します。

1. Oracle Enterprise Manager のコンソールを起動します。
2. ナビゲータ・ペインで「Database」を選択します。
3. SYSDBA 権限を持つユーザーとしてログインします。
4. 「Instance Management tool」を展開します。
5. 「Configuration」を選択します。
6. 右ペインの「General」タブで、「All Initialization Parameters」ボタンをクリックします。「Edit Database」ダイアログ・ボックスが表示されます。
7. 初期化パラメータの設定を編集します。
8. 「OK」ボタンをクリックします。

   この設定は保存できます。設定を保存すると、Instance Manager を使用してデータベースを管理する場合に便利です。

> **関連項目：** インスタンス管理ツールの使用方法の詳細は、『Oracle Enterprise Manager 管理者ガイド』およびこのツールのオンライン・ヘルプを参照してください。

初期化パラメータ・ファイルの手動による編集 インスタンス管理ツールにアクセスできない場合は、初期化パラメータ・ファイルを手動で編集する方法もあります。初期化パラメータ・ファイルの内容を変更するには、テキスト・エディタを使用します。

通常、レプリケーションで使用される初期化パラメータはすべて、初期化パラメータ・ファイルの「Advanced Replication」という項目の下にまとめられています。初期化パラメータ・ファイルの内容を変更した後で、データベースを再起動して新しい設定を有効にします。

> **関連項目：** Advanced Replication の初期化パラメータの詳細は、6-4 ページの「初期化パラメータ」および『Oracle9*i* データベース・リファレンス』を参照してください。データベースの再起動の詳細は、『Oracle9*i* データベース管理者ガイド』を参照してください。

マテリアライズド・ビュー・ログの作成

リモート・マテリアライズド・ビュー・サイトのマテリアライズド・ビュー・グループおよびマテリアライズド・ビューを作成する前に、必要なマテリアライズド・ビュー・ログをマスター・サイトまたはマスター・マテリアライズド・ビュー・サイトに作成しておく必要があります。マテリアライズド・ビュー・ログは、高速リフレッシュを使用するマテリアライズド・ビューを 1 つでもサポートしているすべてのマスター表またはマスター・マテリアライズド・ビューで必要になります。

マテリアライズド・ビュー・ログを作成するには、次の権限が必要です。

- CREATE ANY TABLE
- CREATE ANY TRIGGER
- SELECT（マテリアライズド・ビュー・ログのマスターに対する権限）
- COMMENT ANY TABLE

関連項目： レプリケーション・マネージメント・ツールを使用してマスター・サイトまたはマスター・マテリアライズド・ビュー・サイトにマテリアライズド・ビュー・ログを作成する方法の詳細は、レプリケーション・マネージメント・ツールのオンライン・ヘルプの「Creating Materialized View Logs」の項目を参照してください。この項目を参照するには、「Help Contents」で「Materialized View Replication」を選択します。

マテリアライズド・ビュー・ログへの列のロギング

マテリアライズド・ビュー・ログを作成する場合は、必要に応じてログに列を追加できます。マテリアライズド・ビューを高速リフレッシュするには、マテリアライズド・ビュー・ログに次のタイプの列を追加する必要があります。

- 副問合せの WHERE 句で参照される列で、等価結合の一部ではなく、主キー列でもない列。このような列は、フィルタ列と呼ばれます。
- 副問合せが多対多または 1 対多のいずれかの関係の場合に、等価結合内の列で主キー列ではない列。副問合せが多対 1 の場合は、マテリアライズド・ビュー・ログに結合列を追加する必要はありません。

コレクション列はマテリアライズド・ビュー・ログに追加できません。また、完全リフレッシュされるマテリアライズド・ビューの場合は、マテリアライズド・ビュー・ログは不要です。

たとえば、次のような DDL を考えてみます。

```
1) CREATE MATERIALIZED VIEW oe.customers REFRESH FAST AS
2)  SELECT * FROM oe.customers@orc1.world c
3)  WHERE EXISTS
4)    (SELECT * FROM oe.orders@orc1.world o
5)     WHERE c.customer_id = o.customer_id AND o.order_total > 20000);
```

この DDL の 5 行目では、WHERE 句内で 3 つの列が参照されています。列 orders.customer_id と customers.customer_id は等価結合句の一部として参照されています。customers.customer_id は主キー列であるため、この列はデフォルトでロギングされますが、orders.customer_id は主キー列ではないため、マテリアライズド・ビュー・ログに追加する必要があります。また、列 orders.order_total は追加フィルタ列であるため、これもロギングする必要があります。

したがって、orders.customer_id と orders.order_total を oe.orders 表のマテリアライズド・ビュー・ログに追加します。

これらの列が追加されたマテリアライズド・ビュー・ログを作成するには、次の文を発行します。

```
CREATE MATERIALIZED VIEW LOG ON oe.orders
  WITH PRIMARY KEY (customer_id,order_total);
```

oe.customers 表にマテリアライズド・ビュー・ログがすでに存在する場合は、次の文を発行してこれらの列を追加できます。

```
ALTER MATERIALIZED VIEW LOG ON oe.orders ADD (customer_id,order_total);
```

ユーザー定義データ型を使用している場合は、列オブジェクトの属性をマテリアライズド・ビュー・ログにロギングできます。たとえば、oe.customers 表に cust_address.postal_code 属性がある場合、次の文を発行すると、この属性をマテリアライズド・ビュー・ログにロギングできます。

```
ALTER MATERIALIZED VIEW LOG ON oe.customers ADD (cust_address.postal_code);
```

作成するマテリアライズド・ビューの定義問合せを分析し、マテリアライズド・ビュー・ログに追加する必要がある列を識別することをお薦めします。マテリアライズド・ビュー・ログに列を追加せずに、追加列を必要とするマテリアライズド・ビューを作成またはリフレッシュしようとした場合、マテリアライズド・ビューの作成またはリフレッシュは失敗します。

注意： マテリアライズド・ビューを高速リフレッシュするには、副問合せ内の結合列をマテリアライズド・ビュー・ログに追加する必要があります（結合列が主キー列ではなく、副問合せが多対多または 1 対多のいずれかの関係の場合）。副問合せが多対 1 の場合は、マテリアライズド・ビュー・ログに結合列を追加する必要はありません。

関連項目：

- 副問合せを含むマテリアライズド・ビューの詳細は、3-18 ページの「マテリアライズド・ビューを使用したデータのサブセット化」を参照してください。
- 副問合せを含むマテリアライズド・ビューに関する追加情報は、3-24 ページの「副問合せを使用するマテリアライズド・ビューでの制限事項」を参照してください。
- マテリアライズド・ビュー・ログの作成方法の詳細は、6-16 ページの「マテリアライズド・ビュー・ログの作成」を参照してください。

マテリアライズド・ビュー環境の作成

マテリアライズド・ビュー環境は、様々な場所から様々な方法で作成できます。一般的には、マスター・サイトの配置テンプレートを使用してマテリアライズド・ビュー環境をローカルで事前作成し、それをターゲット・マテリアライズド・ビュー・サイトに個別に配置します。

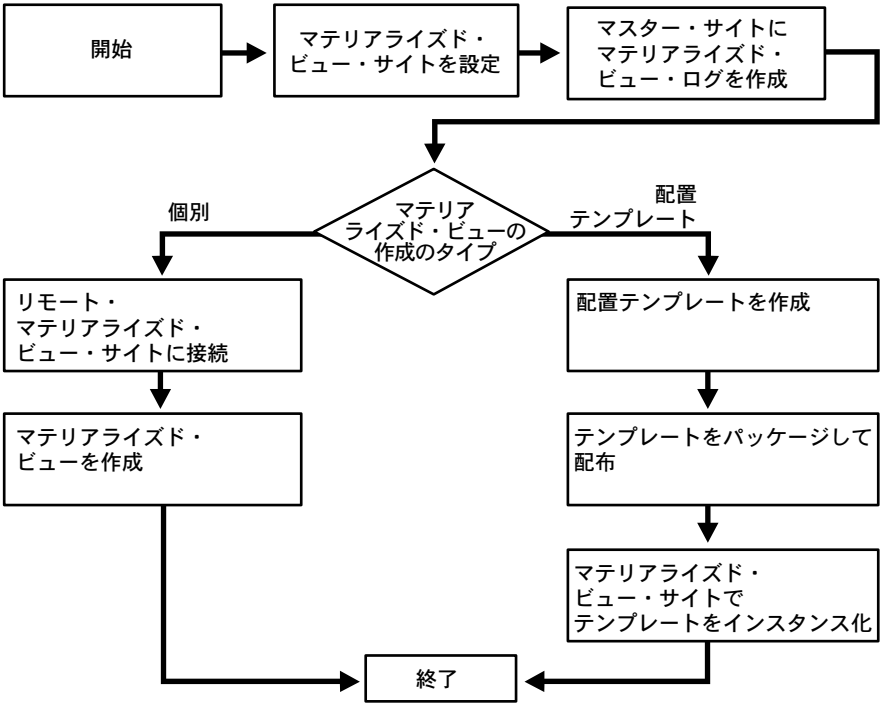
また、マテリアライズド・ビュー・サイトへの接続を確立してマテリアライズド・ビュー環境を直接作成する方法もあります。

レプリケーション・マネージメント・ツールを使用したマテリアライズド・ビュー環境の作成

配置テンプレートの情報を使用してレプリケーション・マネージメント・ツールでマテリアライズド・ビュー環境をまとめて作成する方法の詳細は、レプリケーション・マネージメント・ツールのオンライン・ヘルプを参照してください。

また、リモート・マテリアライズド・ビュー・サイトに直接接続し、レプリケーション・マネージメント・ツールを使用してマテリアライズド・ビュー・サイトを個別に作成する方法の詳細は、レプリケーション・マネージメント・ツールのオンライン・ヘルプを参照してください。

図 6-2 マテリアライズド・ビュー環境作成のためのフローチャート



レプリケーション・マネージメント API を使用したマテリアライズド・ビュー環境の作成

配置テンプレートを使用してレプリケーション・マネージメント API でマテリアライズド・ビュー環境をまとめて事前作成する方法の詳細は、『Oracle9*i* レプリケーション・マネージメント API リファレンス』の配置テンプレートの作成に関する説明を参照してください。

また、リモート・マテリアライズド・ビュー・サイトに直接接続し、レプリケーション・マネージメント API を使用してマテリアライズド・ビュー・サイトを個別に作成する方法の詳細は、『Oracle9*i* レプリケーション・マネージメント API リファレンス』のマテリアライズド・ビュー・グループの作成に関する説明を参照してください。

新規マテリアライズド・ビュー・サイトを追加するときの問題の回避

1 つ以上のマテリアライズド・ビュー・サイトを含むマテリアライズド・ビュー環境を作成した後、新しくマテリアライズド・ビュー・サイトを追加することがあります。次の 2 つの条件がどちらも成立する場合は、新規マテリアライズド・ビュー・サイトに作成したマテリアライズド・ビューを高速リフレッシュしようとしたときに問題が発生することがあります。

- 新規マテリアライズド・ビュー・サイトのマテリアライズド・ビューと、他のマテリアライズド・ビュー・サイトの既存マテリアライズド・ビューが、同一のマスター表またはマスター・マテリアライズド・ビューに基づいている場合。
- 新規マテリアライズド・ビュー・サイトに新規マテリアライズド・ビューを作成中に、既存のマテリアライズド・ビューをリフレッシュする場合。

問題が発生するのは、新規マテリアライズド・ビューが最初の高速リフレッシュを実行する前に、マスターのマテリアライズド・ビュー・ログがパージされたときです。このようなときに新規マテリアライズド・ビュー・サイトでマテリアライズド・ビューを高速リフレッシュしようとすると、次のエラーが発生することがあります。

ORA-12004 REFRESH FAST はマテリアライズド・ビュー *materialized_view_name* に使用できません。
ORA-12034 *materialized_view_name* のマテリアライズド・ビュー・ログは最新のリフレッシュよりも新しいものです。

これらのエラーを受け取った場合の解決策は、新規マテリアライズド・ビューの完全リフレッシュを実行することのみです。

この問題を回避するには、次のオプションのいずれかを選択します。

- 配置テンプレートを使用してマテリアライズド・ビュー・サイトにマテリアライズド・ビュー環境を作成します。配置テンプレートを使用すると、この問題は発生しません。

  関連項目： 配置テンプレートの詳細は、第 4 章「配置テンプレートの概要およびアーキテクチャ」を参照してください。

- 新規マテリアライズド・ビュー・サイトにダミーのマテリアライズド・ビューを作成してから、本番のマテリアライズド・ビューを作成します。ダミーのマテリアライズド・ビューを作成することにより、本番のマテリアライズド・ビューを作成中にマテリアライズド・ビュー・ログがパージされなくなります。

マテリアライズド・ビュー・サイトにダミーのマテリアライズド・ビューを作成する場合は、次の手順を実行します。

1. マスター表またはマスター・マテリアライズド・ビューに基づいた dummy_mview という名前のダミーのマテリアライズド・ビューを作成します。たとえば、sales というマスター表に基づいたダミー・マテリアライズド・ビューを作成するには、新規マテリアライズド・ビュー・サイトで次の文を発行します。

```
CREATE MATERIALIZED VIEW dummy_mview REFRESH FAST AS
  SELECT * FROM pr.sales@orc1.world WHERE 1=0;
```

2. 本番のマテリアライズド・ビューを新規マテリアライズド・ビュー・サイトに作成します。
3. 新規マテリアライズド・ビュー・サイトで本番のマテリアライズド・ビューの高速リフレッシュを実行します。
4. ダミー・マテリアライズド・ビューを削除します。

スケジュール・リンクのガイドライン

スケジュール・リンクによって、あるマスター・サイトから別のマスター・サイトへの遅延トランザクション・キューの伝播方法、あるいは、あるマテリアライズド・ビュー・サイトからそのマスター・サイトまたはマスター・マテリアライズド・ビュー・サイトへの遅延トランザクション・キューの伝播方法が決まります。スケジュール・リンクを作成すると、遅延トランザクション・キューをシステムの別のサイトにプッシュするジョブがローカル・ジョブ・キューに作成されます。リモート・マスター・サイトへの遅延トランザクションの伝播は、レプリケーション・プロパゲータのセキュリティ・コンテキスト内で行われます。

スケジュール・リンクを構成すると、シリアル伝播またはパラレル伝播を使用して情報をプッシュできます。一般的に、parallelism パラメータを 1 に設定した場合でも、パラレル伝播を使用する必要があります。

レプリケーション環境のスケジュール・リンクを作成する前に、レプリケーションをシステム全体でグローバルに発生させる方法を慎重に検討する必要があります。たとえば、一定の間隔で遅延トランザクションを伝播できます。この場合、プッシュする頻度と時刻を決定する必要があります。また、リアルタイム・レプリケーション（同期レプリケーション）をシミュレートするために、各スケジュール・リンクを使用して、マスター・サイトの遅延トランザクション・キューをその接続先に連続的にプッシュする場合もあります。

あるいは、夜間など、必ず接続できる時間帯や通信コストが最低の時間帯にプッシュをスケジュールする場合もあります。さらに、レプリケーションによるネットワーク・リソースへの負荷を分散するために、各マスター・サイト間でリンクのスケジュールをずらすこともできます。

関連項目： シリアル伝播およびパラレル伝播に関連する問題の詳細は、6-27 ページの「シリアル伝播およびパラレル伝播」を参照してください。

定期的なプッシュのスケジューリング

サイトの遅延トランザクション・キューをリモートの接続先に定期的にプッシュするようにスケジュールできます。たとえば、1 日に 1 回または 1 時間に 1 回というように設定できます。これを行うには、DBMS_DEFER_SYS.SCHEDULE_PUSH プロシージャを使用し、表 6-3 に示されている設定を指定します。

表 6-3 定期的なプッシュのスケジューリングの設定

SCHEDULE_PUSH プロシージャ・パラメータ	値
delay_seconds	0
interval	適切な日付式。たとえば、1 時間の間隔を指定するには、'sysdate + 1/24' を使用します。

定期的なプッシュのスケジューリングには、レプリケーション・マネージメント・ツールを使用することもできます。これには、次の場所でスケジュール・リンクを設定するときに、「Delay Seconds」をデフォルト値の 0 に設定します。

- レプリケーション・マネージメント・ツールの「Setup Wizard」
- 「Edit Push Schedule」ダイアログ・ボックス

次に、遅延トランザクション・キューを定期的にプッシュする間隔を設定します。

1 時間ごとの定期的プッシュをスケジュールする例を、次に示します。

```
BEGIN
   DBMS_DEFER_SYS.SCHEDULE_PUSH (
      destination => 'orc2.world',
      interval => 'SYSDATE + (1/24)',
      next_date => SYSDATE,
      delay_seconds => 0);
END;
/
```

関連項目：

- 遅延秒数の設定の詳細は、2-47 ページの「遅延秒数」を参照してください。
- DBMS_DEFER_SYS.SCHEDULE_PUSH プロシージャの詳細は、『Oracle9*i* レプリケーション・マネージメント API リファレンス』を参照してください。
- レプリケーション・マネージメント・ツールの使用方法の詳細は、このツールのオンライン・ヘルプを参照してください。

連続的なプッシュのスケジューリング

非同期レプリケーション・メカニズムを使用している場合でも、スケジュール・リンクを構成すると、連続的なリアルタイム・レプリケーションをシミュレートできます。これを行うには、DBMS_DEFER_SYS.SCHEDULE_PUSH プロシージャを使用し、表 6-4 に示されている設定を指定します。

表 6-4 連続的なプッシュをシミュレートする設定

SCHEDULE_PUSH プロシージャ・パラメータ	値
delay_seconds	1200
interval	delay_seconds 設定よりも低い値
parallelism	1 以上
execution_seconds	delay_seconds 設定よりも高い値

このように設定すると、「Interval」に設定した時間中ずっと、遅延トランザクション・キューに入れられるトランザクションが連続的にプッシュされます。delay_seconds パラメータに指定されている間に、伝播するトランザクションが遅延トランザクション・キューにない場合は、このジョブにより使用されているリソースが解放され、次のジョブ・キュー・プロセスが使用可能になったときに新しく開始されます。

parallelism パラメータを 0（ゼロ）に設定してシリアル伝播を使用している場合は、delay_seconds および interval パラメータの設定を低くして環境に適切な値にし、連続的なプッシュをシミュレートできます。ただし、シリアル伝播を使用している場合、連続的なプッシュのシミュレーションでプッシュ・ジョブを頻繁に開始する必要がある場合はコストが高くなります。

連続的なプッシュをシミュレートする例を次に示します。

```
BEGIN
   DBMS_DEFER_SYS.SCHEDULE_PUSH (
      destination => 'orc2.world',
      interval => 'SYSDATE + (1/144)',
      next_date => SYSDATE,
      parallelism => 1,
      execution_seconds => 1500,
      delay_seconds => 1200);
END;
/
```

関連項目：

- 遅延秒数の設定の詳細は、2-47 ページの「遅延秒数」を参照してください。
- シリアル伝播およびパラレル伝播に関連する問題の詳細は、6-27 ページの「シリアル伝播およびパラレル伝播」を参照してください。
- `DBMS_DEFER_SYS.SCHEDULE_PUSH` プロシージャの詳細は、『Oracle9*i* レプリケーション・マネージメント API リファレンス』を参照してください。

遅延トランザクション・キューのスケジュール・パージのガイドライン

スケジュール・パージによって、マスター・サイトまたはマテリアライズド・ビュー・サイトの遅延トランザクション・キューから適用済みのトランザクションをパージする方法が決定されます。レプリケーション・マネージメント・ツールの「Setup Wizard」を使用してマスター・サイトまたはマテリアライズド・ビュー・サイトを作成すると、一定の基準でローカル遅延トランザクション・キューをパージするジョブが各サイトのローカル・ジョブ・キューに作成されます。レプリケーション環境のサイトを構成する前に、パージを発生させる方法を慎重に検討する必要があります。たとえば、次のような方法があります。

- サイトの遅延トランザクション・キューのプッシュとパージを同期化できます。たとえば、トランザクション・キューのプッシュとパージを連続的に行うように構成できます。このように構成した場合、パフォーマンスが向上することがあります。これは、最後にプッシュされたトランザクションに関する情報が、対応するパージ操作に使用するサーバーのバッファ・キャッシュにすでに存在していることが多いためです。
- サーバーが CPU の限界に達していない場合、遅延トランザクション・キューの連続的なパージをスケジュールすると、可能なかぎりキューのサイズを小さく保つことができます。
- 通常の営業時間帯にトランザクションのスループットが非常に大きくなるサーバーでは、1 日の遅延トランザクション全体を格納できる場合は、オフピーク時にパージが行われるようにスケジュールできます。

定期的なパージのスケジューリング

サイトの遅延トランザクション・キューを定期的にパージするようにスケジュールできます。たとえば、1 日に 1 回または 1 週間に 1 回というように設定できます。これを行うには、DBMS_DEFER_SYS.SCHEDULE_PURGE プロシージャを使用し、表 6-5 に示されている設定を指定します。

表 6-5 定期的なパージのスケジューリングの設定

SCHEDULE_PURGE プロシージャ・パラメータ	値
delay_seconds	0
interval	適切な日付式。たとえば、間隔を 1 日に指定するには、'sysdate + 1' を使用します。

定期的なパージのスケジュールには、レプリケーション・マネージメント・ツールの「Setup Wizard」または「Administration」プロパティ・シートの「Schedule」タブ中にある「Purge」タブを使用することもできます。これを行うには、「Delay Seconds」をデフォルト値の 0（ゼロ）に設定します。次に、遅延トランザクション・キューを定期的にパージする間隔を設定します。

1 日に 1 回の定期的なパージをスケジュールする例を、次に示します。

```
BEGIN
   DBMS_DEFER_SYS.SCHEDULE_PURGE (
      next_date => SYSDATE,
      interval => 'SYSDATE + 1',
      delay_seconds => 0);
END;
/
```

関連項目：

- DBMS_DEFER_SYS.SCHEDULE_PURGE プロシージャの詳細は、『Oracle9*i* レプリケーション・マネージメント API リファレンス』を参照してください。
- レプリケーション・マネージメント・ツールの使用方法の詳細は、このツールのオンライン・ヘルプを参照してください。

連続的なパージのスケジューリング

サイトの遅延トランザクション・キューの連続的なパージを設定するには、DBMS_DEFER_SYS.SCHEDULE_PURGE プロシージャを使用し、表 6-6 に示されている設定を指定します。

表 6-6 定期的なパージのスケジューリングの設定

SCHEDULE_PURGE プロシージャ・パラメータ	値
delay_seconds	500000
interval	delay_seconds 設定よりも低い値
purge_method	dbms_defer_sys.purge_method_quick

連続的なパージのスケジューリングには、レプリケーション・マネージメント・ツールを使用することもできます。これを行うには、「Administration」プロパティ・シートの「Schedule」タブ中にある「Purge」タブで、「Delay Seconds」を 500,000 に設定し、間隔を、「Delay Seconds」設定よりも小さい値に設定します。

連続的なパージをシミュレートする例を次に示します。

```
BEGIN
   DBMS_DEFER_SYS.SCHEDULE_PURGE (
      next_date => SYSDATE,
      interval => 'SYSDATE + (1/144)',
      purge_method => dbms_defer_sys.purge_method_quick,
      delay_seconds => 500000);
END;
/
```

関連項目：

- 遅延秒数の設定の詳細は、2-47 ページの「遅延秒数」を参照してください。
- DBMS_DEFER_SYS.SCHEDULE_PURGE プロシージャの詳細は、『Oracle9*i* レプリケーション・マネージメント API リファレンス』を参照してください。
- レプリケーション・マネージメント・ツールの使用方法の詳細は、このツールのオンライン・ヘルプを参照してください。

シリアル伝播およびパラレル伝播

レプリケーション環境のスケジュール・リンクを作成すると、シリアル伝播またはパラレル伝播を使用して各サイトから接続先に変更を非同期伝播できます。レプリケーション環境を構成する前に、シリアル伝播またはパラレル伝播のいずれを使用するかを決定する必要があります。

- シリアル伝播では、レプリケート・トランザクションは、ソース・システムでコミットされたのと同じ順序で一度に 1 つずつ伝播されます。シリアル伝播を使用するスケジュール・リンクを構成するには、`DBMS_DEFER_SYS.SCHEDULE_PUSH` プロシージャの `parallelism` パラメータを 0（ゼロ）に設定します。または、レプリケーション・マネージメント・ツールを使用して、「Edit Push Schedule」ダイアログ・ボックスの「Parallel Propagation Processes」制御を 0 に設定します。通常、シリアル伝播を使用するのは、レプリケート先が Oracle7 サイトの場合のみです。
- パラレル伝播では、スループットを向上させるため、レプリケート・トランザクションは複数のパラレル・ストリームを使用して伝播されます。必要に応じて依存トランザクションの実行命令が発行されるので、データ整合性が保持されます。パラレル伝播を使用するスケジュール・リンクを構成するには、`DBMS_DEFER_SYS.SCHEDULE_PUSH` プロシージャの `parallelism` パラメータを 1 以上に設定します。または、レプリケーション・マネージメント・ツールを使用して、「Edit Push Schedule」ダイアログ・ボックスの「Parallel Propagation Processes」制御を 1 以上に設定します。

関連項目：

- 2-44 ページ「パラレル伝播」
- `DBMS_DEFER_SYS` パッケージの詳細は、『Oracle9*i* レプリケーション・マネージメント API リファレンス』を参照してください。
- レプリケーション・マネージメント・ツールの使用方法の詳細は、このツールのオンライン・ヘルプを参照してください。

配置テンプレート

レプリケーション環境にマテリアライズド・ビュー・サイトを含める場合、配置テンプレートを使用してマテリアライズド・ビュー・サイトにレプリケート・オブジェクトを作成できます。

関連項目： 配置テンプレートの詳細は、第 4 章「配置テンプレートの概要およびアーキテクチャ」を参照してください。

配置テンプレートをインスタンス化するマテリアライズド・ビュー・サイトの準備

配置テンプレートを使用する場合、配置テンプレートのインスタンス化のためにマテリアライズド・ビュー・サイトを準備する必要があります。配置テンプレートが適切に設計されている場合は、リモート・マテリアライズド・ビュー・サイトで必要な準備作業はほとんどありません。この項では、リモート・マテリアライズド・ビュー・サイトで実行する最も一般的な準備作業について説明します。必要な準備作業がすべて完了した後で、オンライン・インスタンシエーションまたはオフライン・インスタンシエーションを実行できます。

次の点を考慮して、マテリアライズド・ビュー・スナップショット・サイトでインスタンス化を実行するために必要な準備作業を決定します。

- リモート・マテリアライズド・ビュー・サイトからターゲット・マスター・サイトにネットワーク接続されているかどうか。
- マテリアライズド・ビュー・サイトに Oracle Enterprise Edition、Oracle Standard Edition または Oracle Personal Edition のリリース 8.1.5 以上のデータベースがあるかどうか。
- リモート・マテリアライズド・ビュー・サイトがマテリアライズド・ビュー・レプリケーションをサポートするように設定されているかどうか。
- 配置テンプレートに必要なスキーマがマテリアライズド・ビュー・サイトに存在するかどうか。
- 必要なデータベース・リンクが配置テンプレートに含まれていない場合、マテリアライズド・ビュー・サイトからマスター・サイトに必要なデータベース・リンクが存在するかどうか。
- マテリアライズド・ビューで配置テンプレートをインスタンス化するために、オンライン・インスタンシエーションまたはオフライン・インスタンシエーションのどちらを使用するか。
- 配置テンプレートに必要なロールバック・セグメントがマテリアライズド・ビュー・サイトに存在していて、かつそれがオンラインであるかどうか。

次の項では、これらの考慮事項に対する指針を示します。

ネットワーク接続性

ネットワーク接続性は、すべてのレプリケーション環境において Advanced Replication の重要な要素です。Oracle Enterprise Edition、Oracle Standard Edition および Oracle Personal Edition で、ターゲット・マスター・サイトに対する適切な SQL*Net 接続または Oracle Net 接続がリモート・マテリアライズド・ビュー・サイトにあるかどうかを確認します。

> **関連項目：** Oracle ネットワーク接続の設定方法の詳細は、『Oracle9*i* Net Services 管理者ガイド』を参照してください。

データベースのバージョン

マテリアライズド・ビュー・サイトで配置テンプレートをインスタンス化するには、Oracle8*i* リリース 8.1.5 以上のデータベースを使用している必要があります。マテリアライズド・ビュー・サイトがこれらの要件を満たしていない場合は、配置テンプレートをインスタンス化する前にマテリアライズド・ビュー・サイトのデータベースをアップグレードする必要があります。

マテリアライズド・ビュー・サイトの設定

各マテリアライズド・ビュー・サイトには、マテリアライズド・ビュー・サイトをサポートする特殊な権限を持つユーザーが何人か必要です。これらのユーザーは、管理に必要な権限を持ち、さらにデータの伝播とリフレッシュも実行します。

マテリアライズド・ビュー・サイトの設定では、マテリアライズド・ビューの自動リフレッシュ（オプション）や遅延トランザクション・キューのパージを実行するいくつかの自動化ジョブをスケジュールする必要もあります。

マテリアライズド・ビュー・サイトの設定に使用するツールは、次のとおりです。

- **レプリケーション・マネージメント・ツール：** レプリケーション・マネージメント・ツールを使用してリモート・マテリアライズド・ビュー・サイトに接続し、「Setup Wizard」を使用してマテリアライズド・ビュー・サイトを設定できます。

  > **関連項目：** レプリケーション・マネージメント・ツールを使用してマテリアライズド・ビュー・サイトを設定する方法の詳細は、このツールのオンライン・ヘルプを参照してください。

- **レプリケーション・マネージメント API:** レプリケーション・マネージメント・ツールでリモート・マテリアライズド・ビュー・サイトに接続できない場合は、レプリケーション・マネージメント API を使用してマテリアライズド・ビュー・サイトを設定します。マテリアライズド・ビュー・サイトを設定する API コールを含む SQL スクリプトを作成するときに、その他の準備作業（必要なスキーマ、データベース・リンクおよびロールバック・セグメントの作成など、次の 3 つの項で説明する作業）を実行するための DDL と API コールも追加できます。作成したスクリプトは、オフライン・インスタンシエーション・ファイルとともに配布し、オフライン・インスタンシエーション・ファイルよりも前に実行する必要があります。

  関連項目： レプリケーション・マネージメント API を使用してマテリアライズド・ビュー・サイトを設定する方法の詳細は、『Oracle9*i* レプリケーション・マネージメント API リファレンス』を参照してください。

必要なスキーマの作成

インスタンス化する配置テンプレートで複数のスキーマにオブジェクトが作成される場合は、必要なスキーマがすべて作成済みであることを確認する必要があります。また、配置テンプレートをインスタンス化するユーザーは、そのスキーマに対する適切な `CREATE` 権限を持っている必要があります。たとえば、配置テンプレートでスキーマ `oe` にプロシージャを作成し、ユーザー `hr` がこのテンプレートをインスタンス化する場合、`hr` にはスキーマ `oe` に対する `CREATE ANY PROCEDURE` 権限が必要です。

データベース・リンクの作成

リモート・マテリアライズド・ビュー・サイト用に必要なすべてのデータベース・リンクを作成する DDL を配置テンプレートに含めると便利ですが、必ずしもそうする必要はありません。データベース・リンク DDL が配置テンプレートに含まれていない場合は、配置テンプレートをインスタンス化する前に、ターゲット・マスター・サイトへのデータベース・リンクを手動で作成します。データベース・リンクは、オンライン・インスタンシエーション時にマテリアライズド・ビューの実表にデータを移入したり、マテリアライズド・ビュー環境を適切にメンテナンスするために必要です。

オンライン・インスタンシエーションおよびオフライン・インスタンシエーション

配置テンプレートをマテリアライズド・ビュー・サイトでインスタンス化するときは、オンライン・インスタンシエーションまたはオフライン・インスタンシエーションのいずれかを選択できます。オンライン・インスタンシエーションでは、マテリアライズド・ビューのデータはインスタンス化時にマスター・サイトからプルされます。オフライン・インスタンシエーションでは、マテリアライズド・ビューのデータはテンプレート自体にパッケージ化されており、テンプレートをインスタンス化するときにローカルで適用されます。一般的に、マテリアライズド・ビューに大量のデータが含まれている場合は、オフライン・インスタンシエーションの方がネットワーク・リソースの使用率を最小化できます。

関連項目： オンライン・インスタンシエーションおよびオフライン・インスタンシエーションの詳細は、4-9 ページの「配置テンプレートのパッケージ化とインスタンス化」を参照してください。

必要なロールバック・セグメントの作成

インスタンス化している配置テンプレートで、現在リモート・マテリアライズド・ビュー・サイトに存在しない特定のロールバック・セグメントを使用する場合は、必要なロールバック・セグメントを作成します。テンプレート・オブジェクトがデフォルトのロールバック・セグメントと特定のロールバック・セグメントのうちのどちらを使用しているのかを確認するには、DBA_REPCAT_TEMPLATE_OBJECTS ビューを問い合せます。

関連項目： レプリケーションに関係するデータ・ディクショナリ・ビューの詳細は、『Oracle9*i* レプリケーション・マネージメント API リファレンス』を参照してください。

競合解消

非同期のマルチマスター・レプリケーション環境および更新可能マテリアライズド・ビュー・レプリケーション環境では、たとえば異なるサイトから発行された 2 つのトランザクションがほぼ同時に同じ行を更新する場合などに発生する、レプリケーション競合に対処する必要があります。可能なかぎり、競合の発生を回避できるようにレプリケーション環境を計画してください。レプリケーション環境でデータ競合が発生した場合は、ビジネス・ルールに従って競合を解消し、すべてのサイトでデータを適正に収束することを保証するメカニズムが必要です。

関連項目： 競合の回避および競合が発生した場合の競合解消方法の詳細は、第 5 章「競合解消の概念およびアーキテクチャ」を参照してください。

セキュリティ

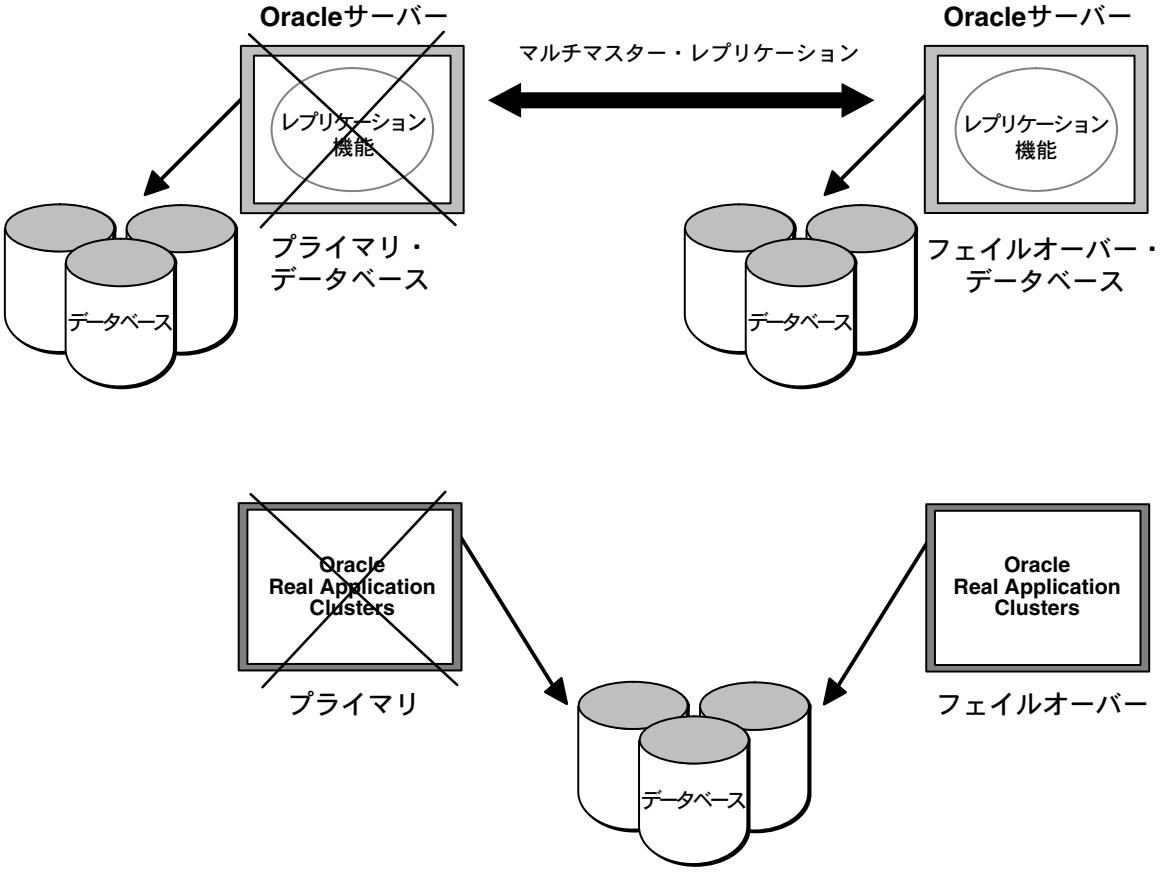
セキュリティは、マルチマスター・レプリケーション環境およびマテリアライズド・ビュー・レプリケーション環境のどちらの環境でも重要な課題です。レプリケーション環境を構成する場合、まずセキュリティ対策を講じる必要があります。

関連項目： レプリケーション環境でのセキュリティ・オプションの詳細は、『Oracle9*i* レプリケーション・マネージメント API リファレンス』を参照してください。

耐障害性を実現するための設計

耐障害性機能とは、システムまたはサイトの障害時にもアプリケーションを継続して実行できる機能です。また、アプリケーションをフェイルオーバー・システム上で実行でき、障害発生時にプライマリ・システム上でアクセスしていたのと同じか、またはほとんど同じデータにアクセスできます。図 6-3 に示されているように、Oracle サーバーではマルチマスター・レプリケーションと Oracle Real Application Clusters という、耐障害性を実現する 2 つのテクノロジを提供しています。

図 6-3 耐障害性を実現するための方法：レプリケーションまたは Oracle Real Application Clusters



Oracle Real Application Clusters とレプリケーションの比較

Oracle Real Application Clusters は、Oracle サーバーのインスタンスをサポートするシステムの障害時にシステムの耐障害性を確保するフェイルオーバーをサポートしています。Oracle Real Application Clusters には、クラスタまたは大規模パラレル・ハードウェア・プラットフォームが必要です。これは、クラスタまたは大規模パラレル・システムが稼働しているローカル環境での処理システムを障害から保護する場合に適しています。

このような環境では、Oracle Real Application Clusters は耐障害性を実現するための理想的なソリューションです。大量のトランザクションをサポートしており、インスタンス障害が発生した場合でもトランザクションの消失やデータの不整合が生じることがありません。インスタンス障害が発生した場合、Oracle Real Application Clusters の正常なインスタンスによって未完了のトランザクションが自動的にリカバリされます。障害が生じたシステムで実行されていたアプリケーションはフェイルオーバー・システムで実行でき、データベースのすべてのデータにアクセスできます。

ただし、Oracle Real Application Clusters には、サイト全体におよぶ障害（停電、洪水、火事、破壊行為など）によってクラスタまたは大規模パラレル・システム全体が操作不能になるような状況に対しては耐障害性がありません。サイト障害に対して耐障害性を確保するために、マルチマスター・レプリケーションを使用して、地理的に離れた場所でデータベースのレプリカをメンテナンスできます。さらに、マルチマスター・レプリケーションを使用すると、異なるオペレーティング・システムまたは Oracle の異なるリリース（あるいはその両方）が稼働するサイト間でデータをレプリケートできます。

ローカル・システムに障害が発生しても、アプリケーションはリモート・サイトで実行を継続できます。マルチマスター・レプリケーションを使用する場合は、障害サイトでのトランザクションのリカバリと、障害サイトを再開したときのデータの非一貫性を防ぐために、なんらかの管理手順が必要になることがあります。

注意： スタンバイ・データベースを構成して、Oracle データベースをサイト障害から守ることもできます。

関連項目：

- 『Oracle9*i* Real Application Clusters 概要』
- スタンバイ・データベースの詳細は、『Oracle9*i* Data Guard 概要および管理』を参照してください。

耐障害性を実現するレプリケーション環境の設計

耐障害性を実現するためにレプリケーション機能を使用する場合は、次の事項を考慮してください。

- レプリケーション機能によって、プライマリ・システムと同じ量のトランザクションを処理できる必要があります。
- プライマリ・サイトで障害が生じた場合、プライマリ・サイトで少し前にコミットされたトランザクションは、フェイルオーバー・サイトにまだ非同期に伝播されていない可能性があります。このトランザクションは消失したように見えます。

  この消失したトランザクションは、プライマリ・サイトがリカバリされたときに処理する必要があります。

  たとえば、リモートのフェイルオーバー・サイトのメンテナンスにレプリケーションを使用する受注入力システムにおいて、システムの実行時にプライマリ・システムで障害が発生したとします。

  障害発生時、少し前にプライマリ・サイトで実行された 2 つのトランザクションの変更は、フェイルオーバー・サイトに伝播および適用させていません。最初のトランザクションは新規の受注の入力で、2 番目のトランザクションは既存の受注の取消しです。

  新規受注入力の場合、フェイルオーバー・システムでの処理の継続中にその受注が存在しないことが発見されると、再入力されると考えられます。既存の受注の取消しの場合は、受注の取消しは発見されずに、受注処理が継続されると考えられます。つまり、取り消された商品が出荷されて、顧客に請求が送られます。

  プライマリ・サイトがリストアされた場合を考えます。フェイルオーバー・システムで実行されたすべての変更をプライマリ・システムに単に戻すと、競合が発生します。

  具体的には、プライマリ・システムに障害が発生する直前に受注された商品に対する注文が重複します。さらに、プライマリ・システムでは取り消されているはずの受注の出荷および請求トランザクションにより、データが変更されます。

こうした状況に対処するには、システムを綿密に設計する必要があります。次の項では、このプロセスについて説明します。

耐障害性の高いシステムの実装

Advanced Replication では、レプリケートされた複数のマスター・サイトを使用してサイト障害に対する耐障害性を提供しています。次の方法のいずれかを使用してシステムを構成する必要があります。実装の最も容易な方法から順番にリストします。

- フェイルオーバー・サイトは読取りアクセス専用で使用します。つまり、プライマリ・サイトで障害が発生した場合でも、フェイルオーバー・サイトでは更新できません。
- 障害の後、プライマリ・サイトは、エクスポートおよびインポート、または全体バックアップによって、フェイルオーバー・サイトからリストアされます。
- すべてのデータおよびトランザクションでは、競合を完全に解消します。このためには、綿密な設計および実現が必要です。重複トランザクションなど、プライマリ・サイトがリストアしたときに発生する競合を、適切かつ確実に解消する必要があります。
- アプリケーション・レベルのルーチンやプロシージャを独自に作成して、プライマリ・サイトがリストアされたとき、およびアクティブなフェイルオーバー・システムでキューに入れられていたトランザクションがプライマリ・サイトに伝播および適用されたときに発生するデータの不整合を処理します。

Oracle Net を使用すると、接続時に自動的に起動されるフェイルオーバーを構成できます。この機能により、最初のマスター・サイトに障害が発生した場合、Oracle Net は別のマスター・サイトに接続を切り替えることができます。自動的な接続時フェイルオーバーを構成するには、`tnsnames.ora` ファイルで `FAILOVER` オプションを on に設定し、複数の接続記述子を指定します。

> **関連項目：** 接続時フェイルオーバーの構成方法の詳細は、『Oracle Net Services 管理者ガイド』を参照してください。

レプリケーション・システムにおけるデータベースのバックアップおよびリカバリ

レプリケーションを使用するデータベースは分散データベースです。レプリケーション・データベースのバックアップを作成するときは、『Oracle9*i* バックアップおよびリカバリ概要』で概説している分散データベースのバックアップのガイドラインに従ってください。レプリケーション・データベースをリカバリするときは、『Oracle9*i* ユーザー管理バックアップおよびリカバリ・ガイド』で説明している協調分散リカバリのガイドラインに従ってください。

協調分散リカバリのガイドラインに従わない場合、レプリケーション・データベースの整合性は保証されません。たとえば、リストアされたマスター・サイトが別のマスター・サイトに異なるトランザクションを伝播していることがあります。場合によっては、不正確なリカバリ操作を修正するために、追加の手順を実行する必要があります。その方法の 1 つは、リカバリしたデータベース内のすべてのレプリケート・オブジェクトを削除し、再作成する方法です。

推奨事項： レプリケート・オブジェクトの削除および再作成を実行する前に、リストアしたデータベースから保留中の遅延トランザクションおよび遅延エラー・レコードを削除し、未解決の分散トランザクションを解決してください。リストアされたデータベースがいくつかのレプリケーション環境のマスター定義サイトであった場合、オブジェクトを削除および作成する前に新しいマスター定義サイトを設定する必要があります。リストアされたデータベースがマスターであるマテリアライズド・ビュー、およびリストアされたデータベース内のマテリアライズド・ビューは、完全リフレッシュを行う必要があります。

データに対する継続的なアクセスを可能にするため、マスター定義サイトの変更（リストアされるデータベースがマスター定義サイトであった場合）、またはマテリアライズド・ビュー・サイトのマスター・サイトの変更（それらのマスター・サイトがリストアされている場合）が必要な場合があります。

インポートされたデータに対するチェックの実行 レプリケート・オブジェクトまたは Advanced Replication で使用するオブジェクト（DBA_REPSITES データ・ディクショナリ・ビューなど）のエクスポートおよびインポートを実行した後は、DBMS_REPCAT パッケージの REPCAT_IMPORT_CHECK プロシージャを実行する必要があります。

次の例では、このプロシージャを使用して、マテリアライズド・ビュー・サイトの acct レプリケーション・グループのオブジェクトをチェックして、オブジェクト識別子とステータス値が適切であることを確認します。

```
BEGIN
DBMS_REPCAT.REPCAT_IMPORT_CHECK( gname     =>   'hr_repg',
                                 master    =>   FALSE);
END;
/
```

関連項目： 『Oracle9*i* レプリケーション・マネージメント API リファレンス』に記載されている REPCAT_IMPORT_CHECK プロシージャを参照してください。

7

レプリケーション・マネージメント・ツールの概要

レプリケーション・マネージメント・ツールは Oracle Enterprise Manager の一部で、レプリケーション環境を設定、管理および監視する Graphical User Interface（GUI）を提供します。レプリケーション・マネージメント・ツールでは、多くの重要な操作をウィザードに従って実行できます。レプリケーション・マネージメント・ツールを使用すると、マルチマスター・レプリケーション環境とマテリアライズド・ビュー・レプリケーション環境の両方を管理できます。

この章では、レプリケーション・マネージメント・ツールの機能について説明します。ただし、レプリケーション・マネージメント・ツールの使用方法の詳細は、このツールのオンライン・ヘルプを参照してください。この章の内容は、次のとおりです。

- レプリケーション・マネージメント・ツールの使用例
- レプリケーション・マネージメント・ツールへのログイン
- レプリケーション・マネージメント・ツールのインタフェース
- レプリケーション・マネージメント・ツールのウィザード
- レプリケーション環境を作成するためのフローチャート

関連項目：

- レプリケーション・マネージメント・ツールの使用方法の詳細は、このツールのオンライン・ヘルプを参照してください。
- Oracle Enterprise Manager の使用方法の詳細は、Oracle Enterprise Manager のマニュアル・セットおよびオンライン・ヘルプを参照してください。

レプリケーション・マネージメント・ツールの使用例

レプリケーション・マネージメント・ツールを使用すると、次の作業を実行できます。

- マスター・サイトおよびマテリアライズド・ビュー・サイトの設定
- レプリケーション環境でのマスター・サイトの追加および削除
- マスター・グループの作成および管理
- トポロジ・ビュー、レポートおよび統計情報を使用した、マスター・サイトとマテリアライズド・ビュー・サイトの監視
- マテリアライズド・ビュー・ログの作成および管理
- マテリアライズド・ビュー・グループの作成および管理
- 個別のマテリアライズド・ビューの作成および管理
- マテリアライズド・ビューのリフレッシュ・グループの作成および管理
- 配置テンプレートの作成および管理
- オフライン・インスタンシエーション用の配置テンプレートのパッケージ化。オンライン・インスタンシエーション用に配置テンプレートをパッケージ化する場合は、レプリケーション・マネージメント API を使用する必要があります。
- 競合解消方法の設定
- スケジュール・リンクの作成、監視および管理
- 管理要求の監視および管理
- 遅延トランザクションの監視および管理
- エラー・トランザクションの監視および管理
- ローカル・ジョブの作成、監視および管理

レプリケーション・マネージメント・ツールへのログイン

レプリケーション環境内のデータベースを表示するように Oracle Enterprise Manager が構成されている場合は、次の手順でレプリケーション・マネージメント・ツールにログインします。

1. Oracle Enterprise Manager をオープンします。

   関連項目： Oracle Enterprise Manager を構成およびオープンする方法の詳細は、Oracle Enterprise Manager のマニュアル・セットを参照してください。

2. データベース・ノードを拡張します。「Database Connect Information」ダイアログ・ボックスが表示されます。

3. データベースのユーザー名とパスワードを入力します。

   レプリケーション環境がすでに設定されている場合は、レプリケーション管理者としてログインします。レプリケーション環境が設定されていない場合は、`SYSTEM` または `SYS` ユーザーとしてログインします。次に、「Setup Wizard」を使用してマスター定義サイトを設定します。マルチマスター・レプリケーション環境を使用する場合は、他のマスター・サイトも設定します。

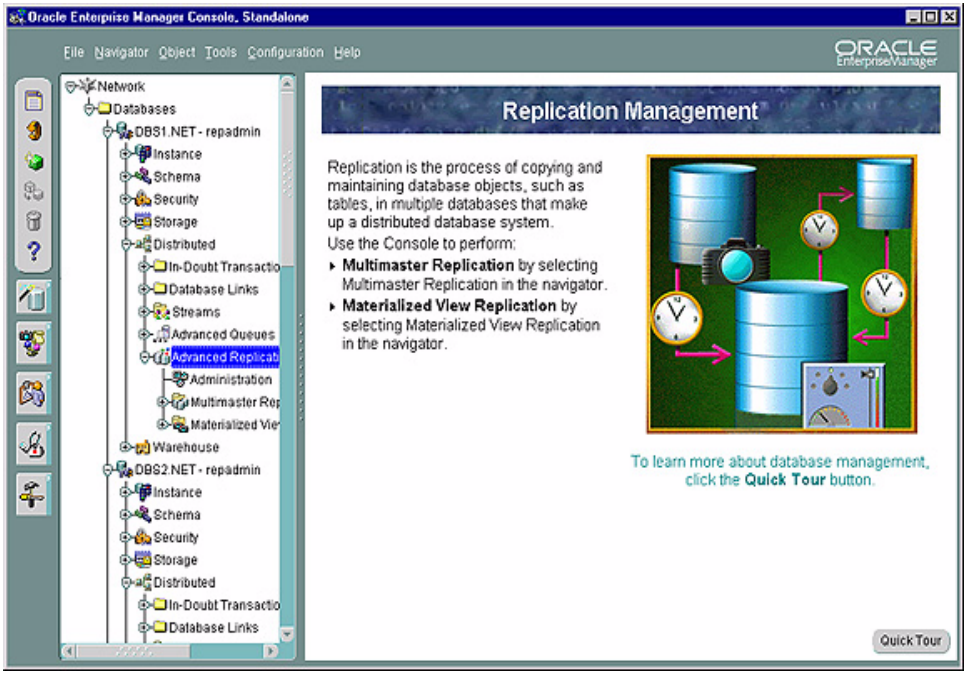
   設定が完了した後で、「Setup Wizard」で指定したレプリケーション管理者としてレプリケーション・マネージメント・ツールにログインします。規則に従えば、レプリケーション管理者のユーザー名は `repadmin` です。設定完了後は、レプリケーション・マネージメント・ツールにログインできるのはレプリケーション管理者のみとし、その他のユーザーはログインしないでください。

4. ナビゲータ・ペインで「Replication」を選択します。

レプリケーション・マネージメント・ツールのインタフェース

Oracle Enterprise Manager におけるレプリケーション・マネージメント・ツールのインタフェースには、左端のツールバーと 2 つのペイン（左のナビゲータ・ペインと右ペイン）が含まれています。ナビゲータ・ペインにはデータベースのツリー構造が表示され、右ペインにはプロパティ・シートが表示されます。

図 7-1 Oracle Enterprise Manager のレプリケーション・マネージメント・ツール



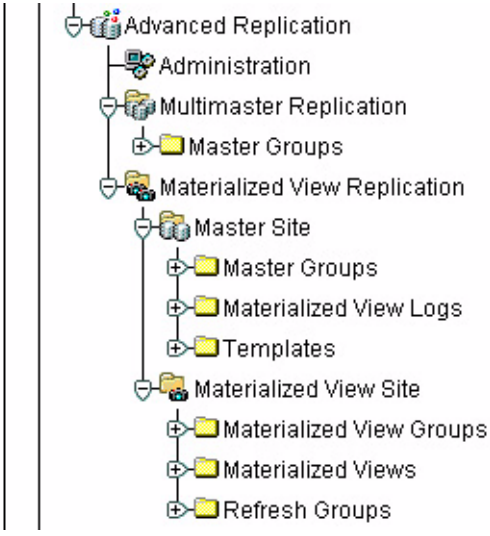
ナビゲータ・ペイン

Oracle Enterprise Manager のナビゲータ・ペインの機能は、Oracle Enterprise Manager の他のアプリケーションのナビゲータ・ペインと同じです。ナビゲータ・ペインでは、次の処理を実行できます。

- レプリケーション環境内のすべてのノードへのアクセス。
- オブジェクトおよびフォルダの拡大および縮小。監視または管理対象のオブジェクトにナビゲートできます。オブジェクトには、マスター・グループ、マテリアライズド・ビュー・グループ、マテリアライズド・ビュー、マテリアライズド・ビュー・ログ、配置テンプレートなどがあります。
- フォルダまたはオブジェクトの右クリック。新規オブジェクトを作成したり、既存オブジェクトに対する操作を実行できます。

Oracle の最新のリリースでは、レプリケーション・マネージメント・ツールのナビゲータ・ツリー構造が新しくなりました。レプリケーション・マネージメント・ツールのナビゲータ・ツリーにある正しいオブジェクトまたはフォルダを選択することで、多数のレプリケーション管理操作を開始できます。

図 7-2 レプリケーション・マネージメント・ツールのナビゲータ・ツリー



次の各項で、ツリー構造内の各オブジェクトおよびフォルダの内容について説明します。

「Administration」オブジェクト

「Administration」オブジェクトを使用すると、レプリケーション・サイト全体を管理できます。「Administration」オブジェクトを選択すると、右ペインに次のタブが表示されます。

- **Topology:** 選択したサイトのレプリケーション環境のグラフィカルなビューを表示します。
- **Errors:** 選択したサイトのローカル・エラーを表示して管理できます。
- **Transactions:** 選択したサイトの遅延トランザクションを表示して管理できます。
- **Schedule:** 選択したサイトのデータベース・リンクのプッシュ・スケジュールを表示して管理できます。また、選択したサイトで正常に伝播された遅延トランザクションのパージ・スケジュールを表示して管理できます。
- **Configuration:** 選択したサイトの構成情報を表示して管理できます。この情報には、レプリケーションで重要な初期化パラメータと管理要求が含まれます。
- **DBMS Jobs:** `DBMS_JOBS` パッケージを使用して作成されたジョブを表示して管理できます。これらのジョブの多くは重要なレプリケーション機能（遅延トランザクションのプッシュや遅延トランザクション・キューのパージなど）を実行します。

「Multimaster Replication」オブジェクト

レプリケーション環境がマルチマスター・レプリケーション用に構成されている場合は、「Multimaster Replication」オブジェクトを使用してマスター・サイトを設定し、マスター・グループを管理します。「Setup Wizard」を使用してマスター・サイトを設定するには、「Multimaster Replication」オブジェクトを右クリックし、「Setup Master Sites」を選択します。

注意： マルチマスター・レプリケーションには Oracle Enterprise Edition が必要です。Oracle Standard Edition がインストールしてある場合は、各マスター・グループに含められるマスター・サイトは 1 つのみです。

「Master Groups」フォルダ「Master Groups」フォルダには、選択されたサイトのマスター・グループが含まれます。マスター・グループを選択すると、次の操作を実行できます。

- マスター・グループの起動と停止（休止）
- マスター・グループの管理要求の表示と適用
- マスター・グループの管理要求のパージ
- マスター・グループ内のオブジェクトに対する DDL 操作の実行

- マスター・グループ内のレプリケート・オブジェクトの管理
- マスター・グループに含まれるレプリケーション・サイトの管理

新規マスター・グループを作成するには、「Master Groups」フォルダを右クリックし、「Create」を選択します。

「Materialized View Replication」オブジェクト

「Materialized View Replication」オブジェクトを使用すると、マテリアライズド・ビュー・レプリケーションに関するレプリケーション・サイトを管理できます。マテリアライズド・ビュー・サイトのマスター・サイトおよびマテリアライズド・ビュー・サイト自体を管理できます。「Setup Wizard」を使用してマスター・サイトまたはマテリアライズド・ビュー・サイトを設定するには、「Materialized View Replication」オブジェクトを右クリックし、「Setup Sites」を選択します。

「Materialized View Replication」オブジェクトには次のオブジェクトが含まれています。

- 「Master Site」オブジェクト
- 「Materialized View Site」オブジェクト

注意： レプリケーション・サイトは次のいずれかになります。

- マスター・グループのみが含まれていてマテリアライズド・ビューが含まれていない場合は、マスター・サイトのみです。この場合、サイトの管理には「Master Site」オブジェクトを使用します。
- マテリアライズド・ビューのみが含まれていてマスター・グループが含まれていない場合は、マテリアライズド・ビュー・サイトのみです。この場合、サイトの管理には「Materialized View Site」オブジェクトを使用します。
- マスター・グループとマテリアライズド・ビューの両方が含まれている場合は、マスター・サイトとマテリアライズド・ビュー・サイトの両方になります。この場合、サイトの管理には「Master Site」オブジェクトと「Materialized View Site」オブジェクトの両方を使用します。

「Master Site」オブジェクト このオブジェクトを使用すると、レプリケーション・サイトのマスター・グループと、マテリアライズド・ビュー・サイトのマスター・サイトを管理できます。具体的には、このオブジェクトを使用すると、配置テンプレートの作成、管理およびパッケージ化と、マテリアライズド・ビュー・ログの作成および管理が可能です。「Setup Wizard」を使用してマスター・サイトを設定するには、「Master Site」オブジェクトを右クリックし、「Setup Master Sites」を選択します。

「Master Site」オブジェクトには次のフォルダが含まれます。

- **「Master Groups」フォルダ :** このフォルダは、「Multimaster Replication」オブジェクトの下の「Master Groups」フォルダと同じ機能を提供します。詳細は、7-6 ページの「「Multimaster Replication」オブジェクト」の「Master Groups」フォルダに関する説明を参照してください。
- **「Materialized View Logs」フォルダ :** このフォルダを使用すると、マスター・サイトのマテリアライズド・ビュー・ログを作成および管理できます。新規マテリアライズド・ビュー・ログを作成するには、「Materialized View Logs」フォルダを右クリックし、「Create」を選択します。
- **「Templates Folder」フォルダ :** このフォルダを使用すると、マスター・サイトで配置テンプレートを作成、管理およびパッケージ化できます。「Templates」フォルダを右クリックし、次のいずれかを選択します。
  - 「Create Using Wizard」。「Deployment Template Wizard」を使用して新規配置テンプレートを作成する場合。
  - 「Copy」。「Copy Template Wizard」を使用して配置テンプレートのローカル・コピーまたはリモート・コピーを作成する場合。
  - 「Compare」。2 つのローカル配置テンプレートを比較する場合。
  - 「Template Script Generation」。「Template Script Generation Wizard」を使用してオフライン・インスタンシエーション用に配置テンプレートをパッケージ化する場合。

「Materialized View Site」オブジェクト このオブジェクトを使用すると、マテリアライズド・ビュー・サイトを管理できます。具体的には、このオブジェクトを使用すると、マテリアライズド・ビュー・グループ、マテリアライズド・ビューおよびリフレッシュ・グループを作成および管理できます。「Setup Wizard」を使用してマテリアライズド・ビュー・サイトを設定するには、「Materialized View Site」オブジェクトを右クリックし、「Setup Materialized View Sites」を選択します。

「Materialized View Site」オブジェクトには次のフォルダが含まれます。

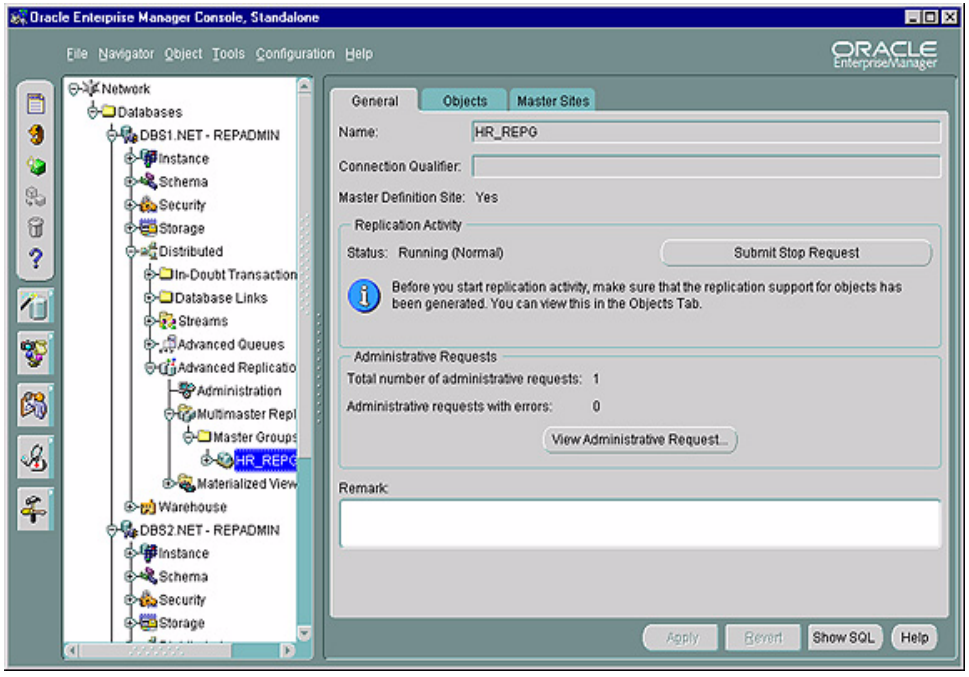
- **「Materialized View Groups」フォルダ :** このフォルダを使用すると、マテリアライズド・ビュー・サイトのマテリアライズド・ビュー・グループを作成および管理できます。「Materialized View Group Wizard」を使用して新規マテリアライズド・ビュー・グループを作成するには、「Materialized View Groups」フォルダを右クリックし、「Create Using Wizard」を選択します。
- **「Materialized Views」フォルダ :** このフォルダを使用すると、マテリアライズド・ビュー・サイトの個別のマテリアライズド・ビューを作成および管理できます。新規マテリアライズド・ビュー・ログを作成するには、「Materialized Views」フォルダを右クリックし、「Create」を選択します。
- **「Refresh Groups」フォルダ :** このフォルダを使用すると、マテリアライズド・ビュー・サイトのリフレッシュ・グループを作成および管理できます。新規リフレッシュ・グ

ループを作成するには、「Refresh Groups」フォルダを右クリックし、「Create」を選択します。

右ペイン

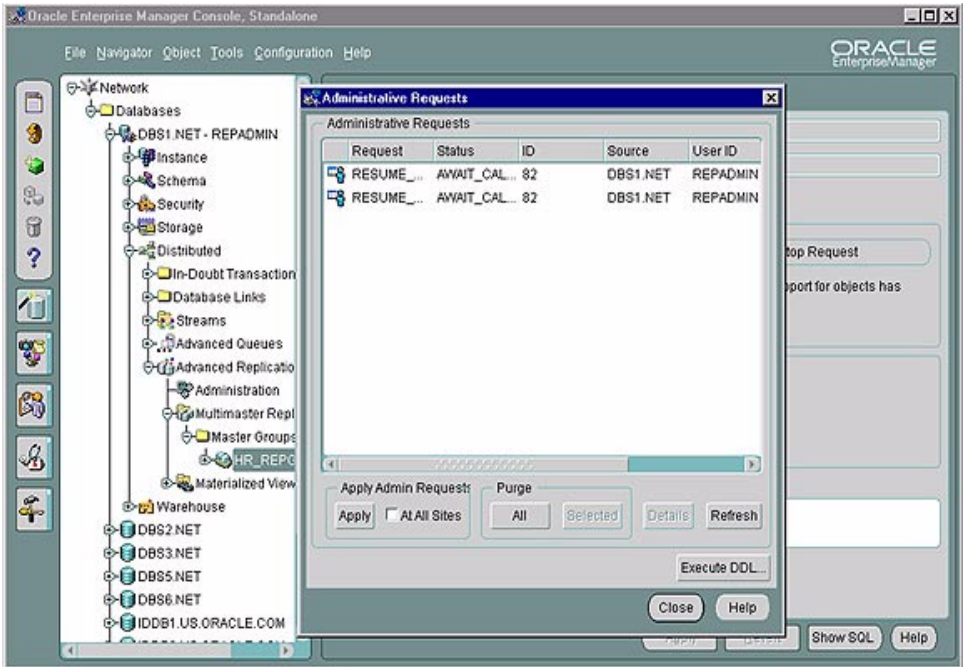
レプリケーション・マネージメント・ツールの右ペインでは、レプリケーション環境を監視および管理できます。図 7-3 に、「Master Group」プロパティ・シートの「General」タブの例を示します。

図 7-3 プロパティ・シートの例



プロパティ・シート上のボタンをクリックすると、ダイアログ・ボックスが開きます。たとえば、マスター・グループの「General」プロパティ・シートの「View Administrative Request」ボタンをクリックすると、図 7-4 のように「Administrative Requests」ダイアログ・ボックスが表示されます。

図 7-4 ダイアログ・ボックスの例



「Topology」タブ

レプリケーション・マネージメント・ツールの右ペインは、レプリケーション環境の監視に使用できます。「Administration」プロパティ・シートの「Topology」タブは、使用可能な監視ツールの例です。このタブには、レプリケーション環境内のノードとノード間のリンクがグラフィカルに表示されます。

「Topology」タブには、次の情報が表示されます。

- 選択したサイトのマスター・グループのレプリケーション・アクティビティに関連するすべてのマスター・サイト
- 各マスター・サイトでの遅延トランザクションの数
- 遅延トランザクションの接続先サイト
- 各マスター・サイトでの管理要求の数
- マスター・サイトでのローカル・エラーの有無とエラー数
- 選択したマスター・サイトに接続されているマテリアライズド・ビューまたはマテリアライズド・ビュー・グループの合計数

- マスター・サイトがマテリアライズド・ビュー・サイトとしても機能しているかどうか

「Topology」タブは、ゼロより大きい値のみを表示します。たとえば、マスター・サイトでの管理要求数がゼロの場合、管理要求数は表示されません。Oracle9*i* 以上が稼働するレプリケーション・サイトに接続している場合は、「Topology」タブに「Statistics」ボタンも表示されます。このボタンをクリックすると、レプリケーション・サイトのパフォーマンス統計情報が表示されます。

注意： 選択したサイトにマスター・グループ、マテリアライズド・ビュー・グループまたはマテリアライズド・ビューを 1 つも作成していない場合、レプリケーションを構成する必要があることを示すメッセージが「Topology」タブに表示されます。

「Topology」タブには次のアイコンとイメージが表示されます。

アイコン / イメージ	説明
	マスター・サイトを示します。
	対応するサイトで処理中の現在の管理要求数を示します。
	遅延トランザクション・エラーまたは管理要求エラー（あるいはその両方）が発生したマスター・サイトを示します。
	対応する矢印が指すサイトに適用する遅延トランザクションの現在の数を示します。
	対応するサイトでの遅延トランザクション・エラーの数を示します。
	マスター・サイトおよびマテリアライズド・ビュー・サイトの両方として機能するサイト（二重サイト）を示します。
	遅延トランザクション・エラーまたは管理要求エラー（あるいはその両方）が発生した二重サイトを示します。

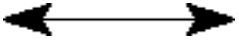
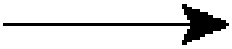
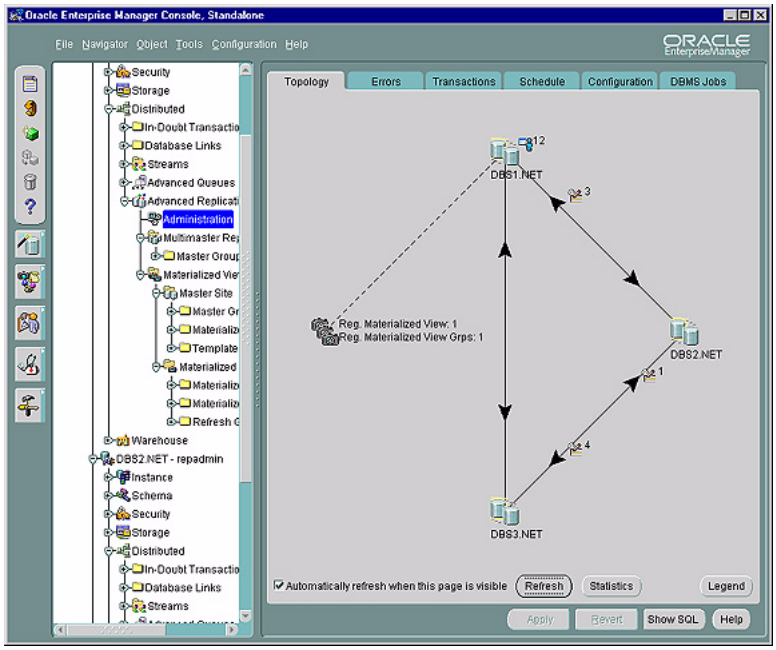
アイコン / イメージ	説明
	マスター・サイトに接続されているときに、そのマスター・サイトにマテリアライズド・ビュー・サイトがあることを示します。マテリアライズド・ビューまたはマテリアライズド・ビュー・グループ（あるいはその両方）は、マスター・サイトに登録できます。
	マテリアライズド・ビュー・サイトに接続されているときに、マテリアライズド・ビュー・サイトを示します。
	エラーのあるマテリアライズド・ビュー・サイトを示します。
 （黒の実線の矢印）	現在のマスター・サイトからリモート・マスター・サイトへのデータベース・リンクが機能していることを示します。 **注意 :** レプリケーション・マネージメント・ツールでは、リモート・サイトのデータベース・リンクのテストは行いません。したがって、リモート・サイトから現在のマスター・サイトへのデータベース・リンクが機能しているかどうかや、障害の有無は判断できません。これを判断するには、Oracle Enterprise Manager でリモート・サイトに接続し、「Topology」タブを表示します。
（赤の破線の矢印）	現在のマスター・サイトからリモート・マスター・サイトへのデータベース・リンクに障害があることを示します。 **注意 :** レプリケーション管理者（通常は `repadmin` ユーザー）としてログインしていない場合は、2 つのサイト間のデータベース・リンクが正常に機能している場合でも、表示上では機能していないように見えることがあります。このように表示されるのは、そのユーザーにサイト間のプライベート・データベース・リンクがないためです。レプリケーション管理者にはプライベート・データベース・リンクがあります。このような場合は、レプリケーション管理者としてサイトにログインし、リンクに障害があるか確認します。
 （1 方向の黒の矢印）	マテリアライズド・ビュー・サイトに接続されているときに、マテリアライズド・ビュー・サイトとマスター・サイト間のデータベース・リンクを示します。2 つのサイト間には現在ネットワーク接続がある場合とない場合があります。
（黒の破線）	マテリアライズド・ビュー・サイトとマスター・サイト間の 1 つ以上のデータベース・リンクを示します。サイト間には現在ネットワーク接続がある場合とない場合があります。

図 7-5 レプリケーション・マネージメント・ツールの「Topology」タブの例



マスター・データベース orc1.world に接続しているとき、この例の「Topology」タブには次の情報が表示されます。

- データベース orc1.world と orc2.world は、同一レプリケーション環境内のマスター・サイトです。
- データベース orc1.world はマスター・サイトとしてのみ機能し、マテリアライズド・ビュー・サイトではないことが、このデータベースのアイコンから判断できます。もう一方のマスター・データベース（orc2.world）は、マスター・サイトとしてのみ機能しているか、二重サイトとして機能しているかのいずれかです。これを判断するには、orc2.world に接続します。
- マスター・データベース orc1.world には、処理対象の管理要求が 2 つあります。
- マスター・データベース orc1.world には、ローカルの遅延トランザクション・エラーとローカルの管理要求エラーはありません。
- マスター・データベース orc2.world には、処理対象の管理要求はありません。
- マスター・データベース orc2.world には、ローカルの遅延トランザクション・エラーとローカルの管理要求エラーはありません。

- マスター・データベース orc1.world の遅延トランザクション・キューには、orc2.world に適用される予定の遅延トランザクションが 8 つあります。
- マスター・データベース orc2.world の遅延トランザクション・キューには、orc1.world に適用される予定の遅延トランザクションが 3 つあります。
- マスター・データベース orc1.world には、マテリアライズド・ビュー・グループが 1 つとマテリアライズド・ビューが 15 個登録されています。これらの登録済みマテリアライズド・ビューは、複数のマテリアライズド・ビュー・サイトに分散している可能性があります。

このレプリケーション環境のその他のマスター・データベースにもマテリアライズド・ビューがありますが、これらのマテリアライズド・ビューが「Topology」タブに表示されるのは、これらのデータベースに接続しているときのみです。たとえば、マスター・データベース orc2.world にマテリアライズド・ビュー・サイトがあるかどうかを判断するには、Oracle Enterprise Manager で orc2.world データベースに接続し、「Topology」タブを開きます。

レプリケーション・マネージメント・ツールのウィザード

レプリケーション・マネージメント・ツールのウィザードは、複数の手順が必要な作業に対して画面ごとに指示を提供します。ウィザードを使用すると、その指示に従って作業を進めることができるので、複雑な作業を簡単に行えます。次の項では、レプリケーション・マネージメント・ツールのウィザードについて説明します。

- Setup Wizard
- Materialized View Group Wizard
- Deployment Template Wizard
- Template Script Generation Wizard
- Copy Template Wizard

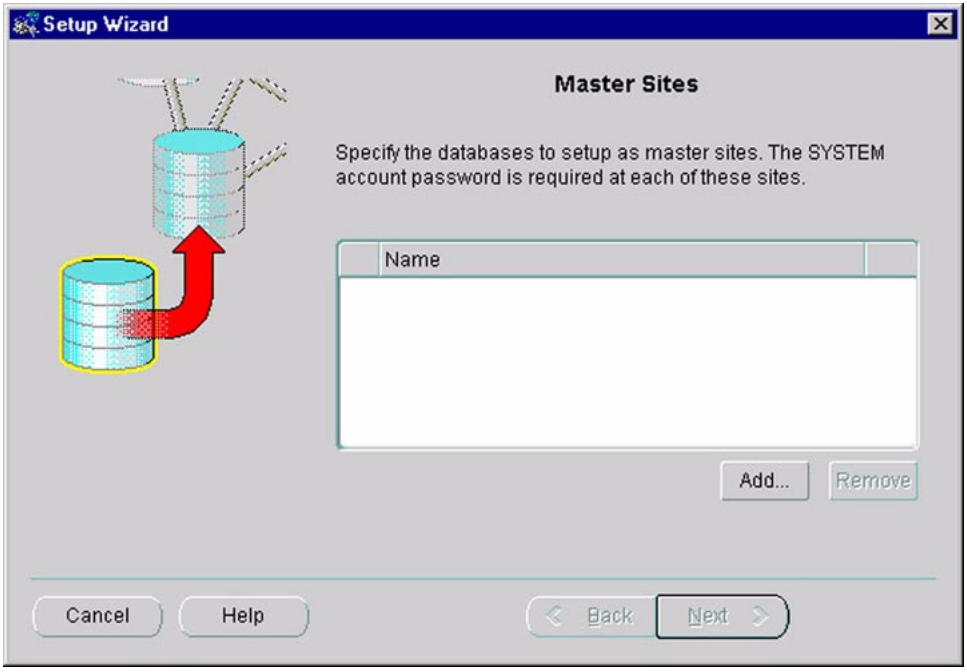
Setup Wizard

「Setup Wizard」の指示に従って操作すると、レプリケーション用のマスター・サイトおよびマテリアライズド・ビュー・サイトを設定できます。「Setup Wizard」を使用することで、レプリケーション・サイトを簡単に準備できます。指定した各マスター・サイトごとに、「Setup Wizard」で次の手順を実行します。

- レプリケーション管理者の役割をするデータベース・アカウントの作成。デフォルトでは「Setup Wizard」により、このアカウントにレプリケーション・プロパゲータおよび受信者も作成されます。マスター・サイトでのレプリケーション管理者のデフォルト・ユーザー名は `repadmin` で、マテリアライズド・ビュー・サイトでのデフォルト・ユーザー名は `mvadmin` です。ただし、これらのユーザー名は変更できます。
- レプリケーション管理者アカウントに対する必要な権限の付与。
- 各レプリケーション・サイトの新規レプリケーション管理者アカウントに対応するデータベース・リンクの作成。
- マスター・サイトおよびマテリアライズド・ビュー・サイト（オプション）に対して、マスター・サイト間で変更をプッシュするジョブのスケジュール。
- システム内のすべてのサイトの遅延トランザクション・キューから、完了したトランザクションをパージするジョブのスケジュール。

「Setup Wizard」は、マスター・サイト用とマテリアライズド・ビュー・サイト用で多少異なります。マスター・サイトの設定用に「Setup Wizard」を開くには、ナビゲータ・ペインの「Multimaster Replication」オブジェクトを右クリックし、「Setup Master Sites」を選択します。

図 7-6 マスター・サイト用「Setup Wizard」の初期画面

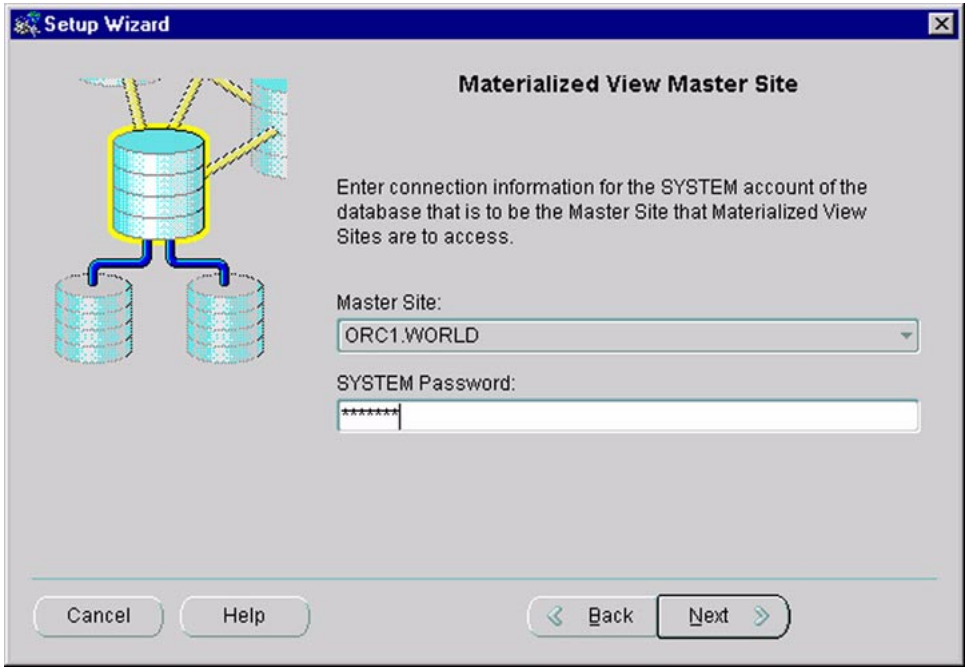


関連項目：「Setup Wizard」を使用してマスター・サイトを設定する方法の詳細は、レプリケーション・マネージメント・ツールのオンライン・ヘルプの「Setup Master Sites: Overview」の項目を参照してください。この項目を参照するには、「Help Contents」で「Replication」、「Set Up Replication Sites」、「Master Site」の順に選択します。

マテリアライズド・ビュー・サイトの設定用に「Setup Wizard」を開くには、ナビゲータ・ペインの「Materialized View Site」オブジェクトを右クリックし、「Setup Materialized View Sites」を選択します。

注意：「Setup Wizard」は、アントラステッド・セキュリティを使用してマテリアライズド・ビュー・サイトを設定します。アントラステッド・セキュリティの詳細は、『Oracle9*i* レプリケーション・マネージメント API リファレンス』を参照してください。

図 7-7 マテリアライズド・ビュー・サイト用「Setup Wizard」の初期画面



関連項目：「Setup Wizard」を使用してマテリアライズド・ビュー・サイトを設定する方法の詳細は、レプリケーション・マネージメント・ツールのオンライン・ヘルプの「Set Up Materialized View Sites: Overview」の項目を参照してください。この項目を参照するには、「Help Contents」で「Replication」、「Set Up Replication Sites」、「Materialized View Site」の順に選択します。

関連項目：

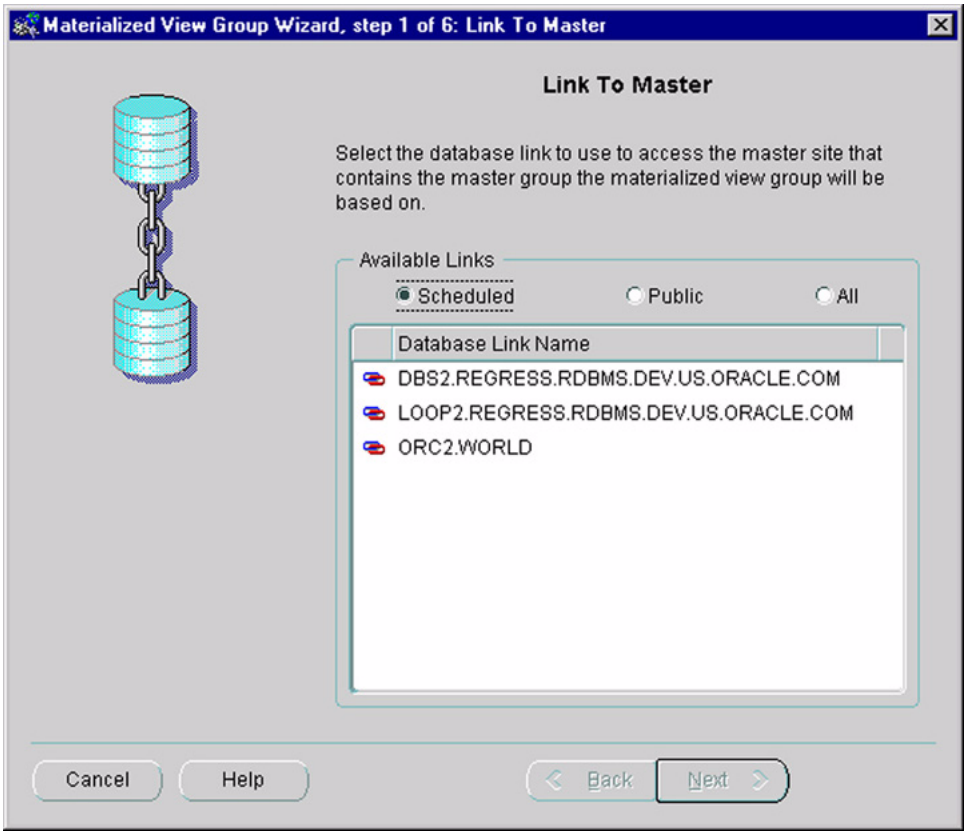
- マスター・サイトの詳細は、第 2 章「マスター・レプリケーションの概要とアーキテクチャ」を参照してください。
- マテリアライズド・ビュー・サイトの詳細は、第 3 章「マテリアライズド・ビューの概要とアーキテクチャ」を参照してください。

Materialized View Group Wizard

「Materialized View Group Wizard」の指示に従って操作すると、マスター・グループに基づいたマテリアライズド・ビュー・グループを作成できます。各マテリアライズド・ビューは、ソースのマスター・グループ内のマスター表の部分的なコピーまたは完全なコピーです。マテリアライズド・ビュー・グループはリモート・マテリアライズド・ビュー・サイトに置かれており、マスター・サイトの 1 つのターゲット・マスター・グループに基づいています。

「Materialized View Group Wizard」は、マテリアライズド・ビュー・グループを作成するマテリアライズド・ビュー・サイトで実行します。「Materialized View Group Wizard」を開くには、ナビゲータ・ペインの「Materialized View Group」フォルダを右クリックし、「Create Using Wizard」を選択します。

図 7-8「Materialized View Group Wizard」の初期画面



関連項目：「Materialized View Group Wizard」を使用してマテリアライズド・ビュー・グループを設定する方法の詳細は、レプリケーション・マネージメント・ツールのオンライン・ヘルプの「Create Materialized View Group: Overview」の項目を参照してください。この項目を参照するには、「Help Contents」で「Replication」、「Materialized View Replication」、「Create」を選択します。

関連項目： マテリアライズド・ビュー・グループの詳細は、3-53 ページの「マテリアライズド・ビュー・グループ」を参照してください。

Deployment Template Wizard

配置テンプレートを使用すると、多くのリモート・マテリアライズド・ビュー・サイトの配置およびメンテナンスの作業が簡単になります。また、配置テンプレートを使用すると、マスター・サイトでマテリアライズド・ビュー定義の集合を定義し、その定義中のパラメータを使用してマテリアライズド・ビューを個々のユーザーまたはユーザーのタイプにあわせてカスタマイズできます。

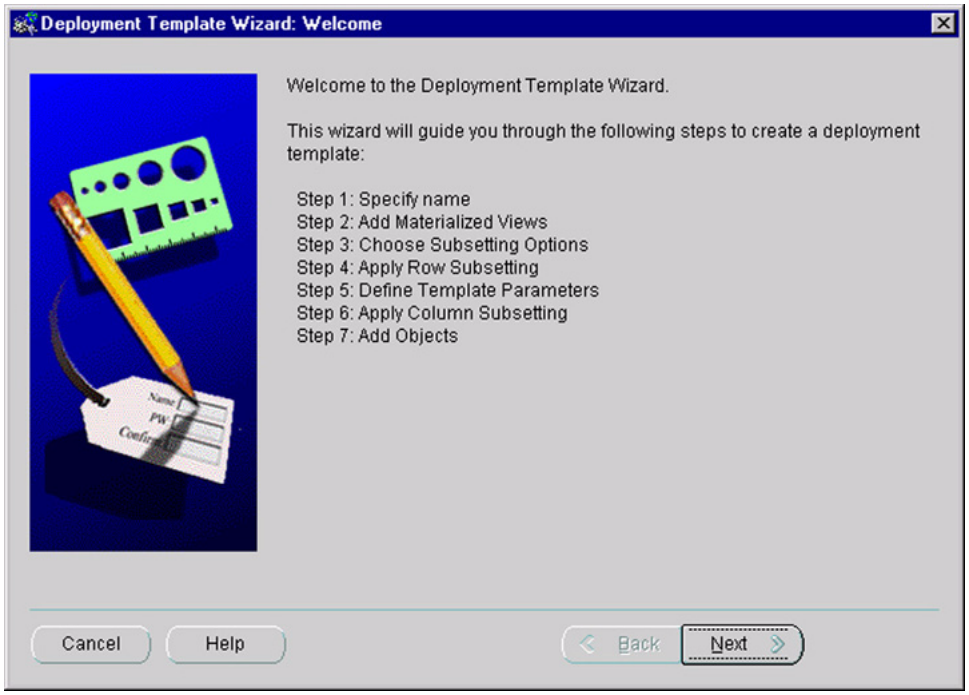
たとえば、営業用に 1 つのテンプレートを作成し、サービス技術員用に別のテンプレートを作成する場合などです。この場合、販売地域または顧客サポート・レベルなどのパラメータ値を使用できます。ユーザーが SQL スクリプトを実行してテンプレートをインスタンス化すると、リモート・サイトで適切なマテリアライズド・ビューが作成され、データが移入されます。

「Deployment Template Wizard」の指示に従って操作すると、配置テンプレートを作成できます。「Deployment Template Wizard」の各画面では、次の処理を実行できます。

- 配置テンプレートの命名およびパブリックまたはプライベートの指定。プライベート・テンプレートの場合は、認証ユーザーを指定できます。
- 配置テンプレートへのオブジェクトの追加。
- 配置テンプレートのパラメータの指定。

「Deployment Template Wizard」は、配置テンプレートを作成するマスター・サイトから実行します。「Deployment Template Wizard」を開くには、ナビゲータ・ペインの「Templates」フォルダを右クリックし、「Create Using Wizard」を選択します。

図 7-9「Deployment Template Wizard」の初期画面



関連項目： 「Deployment Template Wizard」を使用して配置テンプレートを作成する方法の詳細は、レプリケーション・マネージメント・ツールのオンライン・ヘルプの「Overview of Creating a Deployment Template」の項目を参照してください。この項目を参照するには、「Help Contents」で「Replication」、「Deployment Templates」、「Create」の順に選択します。

関連項目： 配置テンプレートの詳細は、第 4 章「配置テンプレートの概要およびアーキテクチャ」を参照してください。

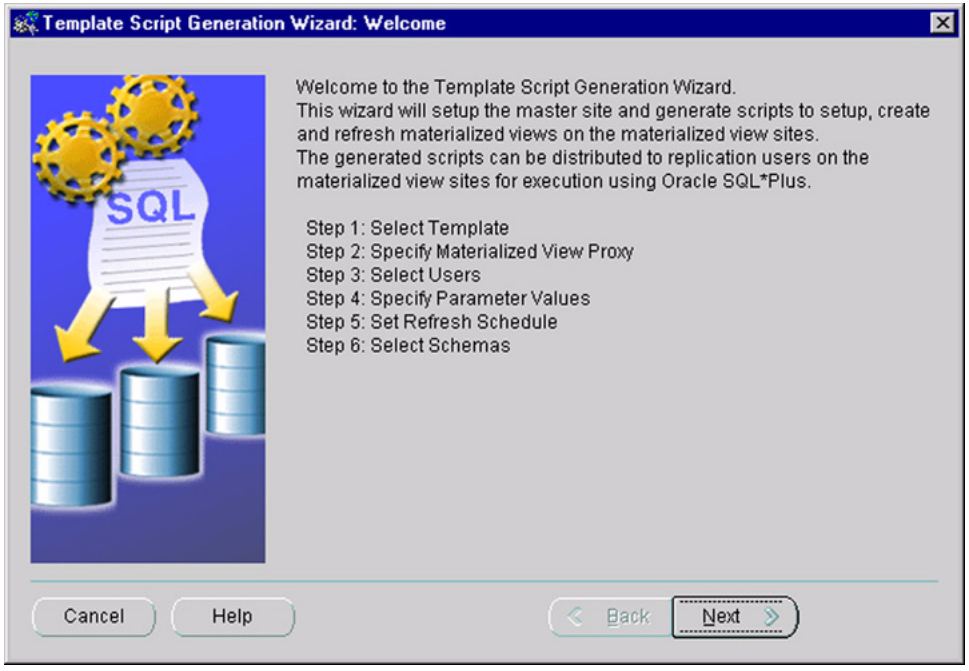
Template Script Generation Wizard

オフライン・インスタンシエーションの場合、エンド・ユーザーは、マスター・サイトにネットワーク接続していない状態で、生成されたファイルを使用してテンプレートをインスタンス化できます。「Template Script Generation Wizard」を使用すると、必要な DDL とデータをすべてファイルに書き込むことができます。そのファイルをマテリアライズド・ビュー・サイトに転送して実行します。これは、低速の WAN 接続を使用しているラップトップ・ユーザー、あるいはネットワーク接続が不安定または低速な場合に最適なソリューションです。

「Template Script Generation Wizard」の指示に従って操作すると、オフライン・インスタンシエーション用の配置テンプレートをパッケージ化できます。また、「Template Script Generation Wizard」で生成するオフライン・インスタンシエーション・ファイルを使用して、マテリアライズド・ビュー・サイトにマテリアライズド・ビューおよびその他のオブジェクトを作成できます。

「Template Script Generation Wizard」は、オフライン・インスタンシエーション・ファイルを生成するテンプレートを含むマスター・サイトで実行します。「Template Script Generation Wizard」を実行するには、ナビゲータ・ペインの「Templates」フォルダを右クリックし、「Template Script Generation」を選択します。

図 7-10「Template Script Generation Wizard」の初期画面



注意： オンライン・インスタンシエーション・ファイルを生成する場合は、レプリケーション・マネージメント API を使用する必要があります。オンライン・インスタンシエーション・ファイルの生成の詳細は、『Oracle9*i* レプリケーション・マネージメント API リファレンス』を参照してください。

関連項目： 「Template Script Generation Wizard」を使用してオフライン・インスタンシエーション用の配置テンプレートをパッケージ化する方法の詳細は、レプリケーション・マネージメント・ツールのオンライン・ヘルプの「Package for Offline Instantiation: Overview」の項目を参照してください。この項目を参照するには、「Help Contents」で「Replication」、「Deployment Templates」、「Packaging and Instantiation」の順に選択します。

関連項目： 配置テンプレートのパッケージ化およびインスタンス化の詳細は、4-15 ページの「パッケージ化およびインスタンス化のプロセス」を参照してください。

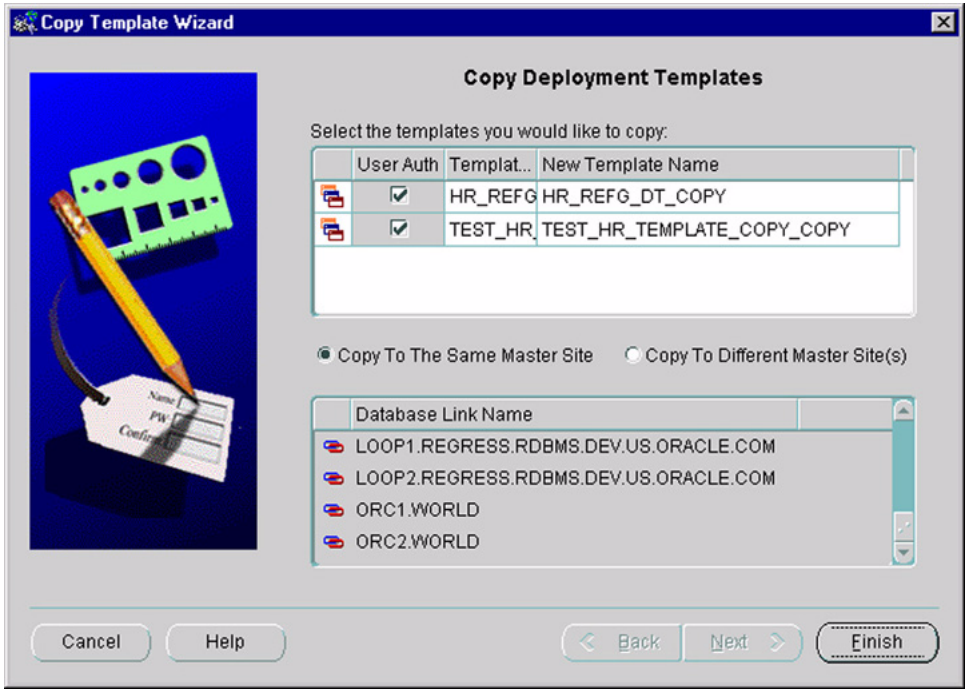
Copy Template Wizard

「Copy Template Wizard」の指示に従って操作すると、配置テンプレートをコピーできます。様々な理由から、配置テンプレートを複数のマスター・サイトにコピーすることが必要な場合があります。たとえば、次の場合です。

- **ネットワーク負荷を分散させる場合 :** ユーザーがテンプレートをインスタンス化する前に、ターゲット・マテリアライズド・ビュー・サイトのマスター・サイトにテンプレートを配置する必要があります。ネットワークの規模が大きい場合、テンプレート定義を複数のマスター・サイトにコピーして、複数のマテリアライズド・ビュー・サイトのネットワーク負荷を分散できます。
- **テンプレートを変更する場合 :** 配置テンプレートを作成した後、そのテンプレートと同じ特性を多数持つ別のテンプレートを作成する必要がある場合があります。新しいテンプレートを作成するかわりにテンプレートをコピーし、これを必要に応じて変更します。

「Copy Template Wizard」は、配置テンプレートを含むマスター・サイトから実行します。「Copy Template Wizard」を開くには、ナビゲータ・ペインの「Templates」フォルダを右クリックし、「Copy」を選択します。

図 7-11「Copy Template Wizard」

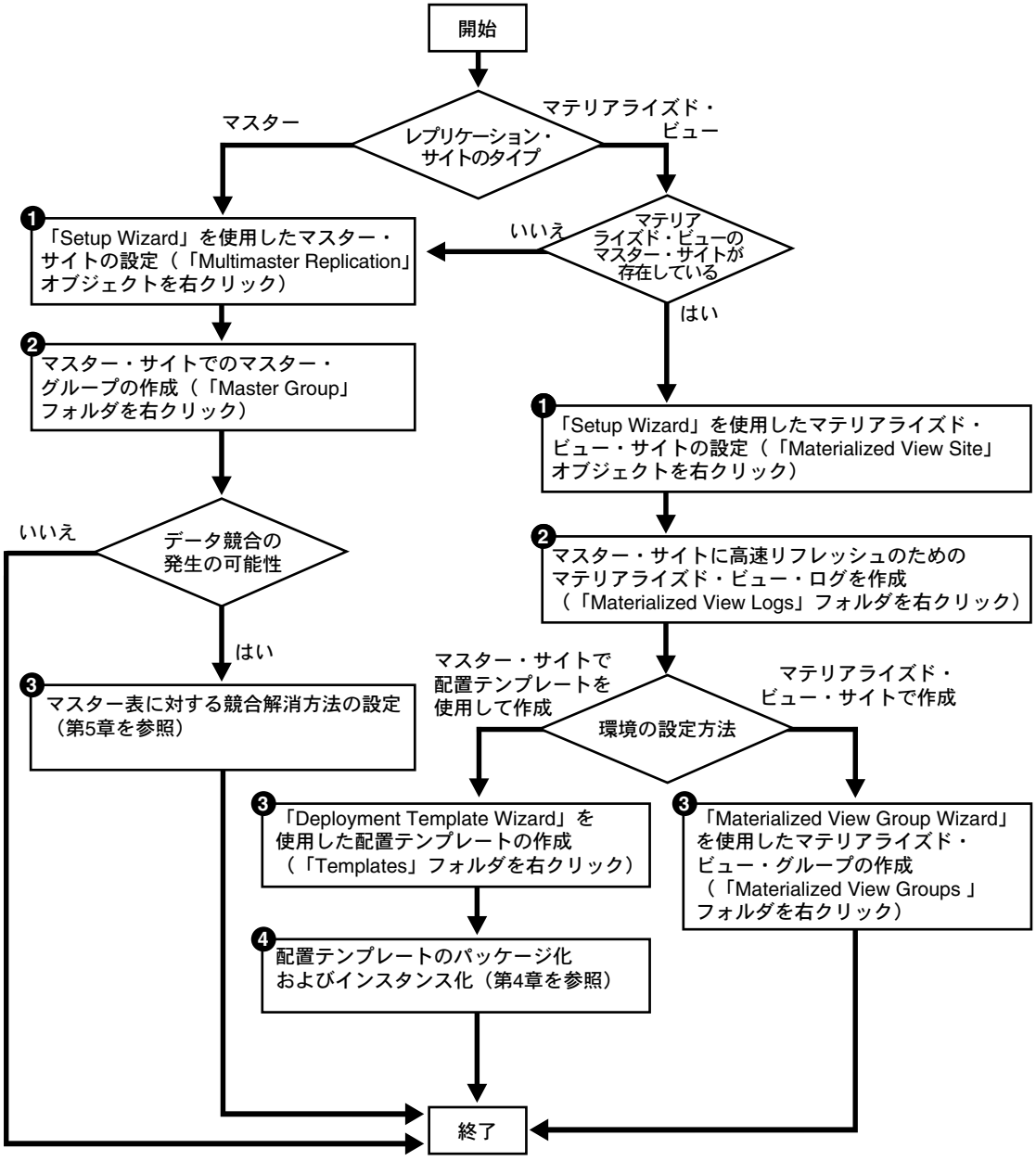


関連項目：「Copy Template Wizard」を使用して配置テンプレートをコピーする方法の詳細は、レプリケーション・マネージメント・ツールのオンライン・ヘルプの「Copying a Template」の項目を参照してください。この項目を参照するには、「Help Contents」で「Replication」、「Deployment Templates」、「Manage」の順に選択します。

レプリケーション環境を作成するためのフローチャート

図 7-12 のフローチャートは、レプリケーション・マネージメント・ツールを使用してレプリケーション環境を作成するときに必要な主な決定事項と作業内容を示しています。このフローチャートでは、マルチマスター環境の場合とマテリアライズド・ビュー環境の場合に分けて表示しています。フローチャートの各手順では、このマニュアルでその作業の詳細を説明している項を示しています。これらの作業を完了する方法の詳細は、レプリケーション・マネージメント・ツールのオンライン・ヘルプを参照してください。

図 7-12 レプリケーション・マネージメント・ツールを使用したレプリケーション環境の作成



A

レプリケーションに関する問題のトラブルシューティング

この付録では、レプリケーション環境を管理するためのトラブルシューティングのガイドラインについて説明します。この付録の内容は、次のとおりです。

- データベース・リンクに関する問題の診断方法
- マスター・サイトに関する問題の診断方法
- 遅延トランザクション・キューに関する問題の診断方法
- マテリアライズド・ビューに関連する問題の診断方法

データベース・リンクに関する問題の診断方法

データベース・リンクが正常に機能していないと考えられる場合、Oracle Enterprise Manager、SQL*Plus または別のツールを使用して、データベース・リンクを削除し、再作成できます。

- データベース・リンク名が、ターゲット・データベースのグローバル名と同じであることを確認します。
- スケジュールした間隔が意図したとおりであることを確認します。
- スケジュールした間隔が、必要な実行時間よりも短くないことを確認します。

指定サイトへのデータベース・リンクで接続修飾子を使用した場合は、そのサイトにリンクする他のサイトでも同じ接続修飾子を使用する必要があります。たとえば、次のようにデータベース・リンクを作成するとします。

```
CREATE DATABASE LINK dbs1.world@myethernet CONNECT TO repadmin
  IDENTIFIED BY secret USING 'connect_string_myethernet';
```

マスター・サイトかマテリアライズド・ビュー・サイトかにかかわらず、`dbs1.world@myethernet` に対応付けられたすべてのサイトで、接続修飾子 `myethernet` を使用する必要があります。

> **関連項目：** データベース・リンクと接続修飾子の詳細は、『Oracle9*i* データベース管理者ガイド』を参照してください。

マスター・サイトに関する問題の診断方法

マルチマスター・レプリケーション・システムで問題が発生する可能性があります。次の各項で、これらの問題とその解決方法について説明します。

- 新しいマスター・サイトでレプリケート・オブジェクトが作成されない場合
- DDL による変更がマスター・サイトに伝播されない場合
- DML による変更が他のサイトに非同期で伝播されない場合
- DML をレプリケート表に適用できない場合
- バルク更新および制約違反
- レプリケート・オブジェクトの再作成
- 表のレプリケーション・サポートを生成できない場合
- レプリケート・プロシージャまたはレプリケート・トリガーに関する問題
- ON DELETE CASCADE および整合性制約に関する問題

新しいマスター・サイトでレプリケート・オブジェクトが作成されない場合

マスター・グループに新しいマスター・サイトを追加しても、新しいマスター・サイトで適切なオブジェクトが作成されない場合は、次の処理を実行します。

- 必要なプライベート・データベース・リンクが新しいマスター・サイトと既存のマスター・サイトの間に存在することを確認します。レプリケーション・マネージメント・ツールの「Setup Wizard」を使用してサイトを設定した場合は、何も問題はありません。リンクは、既存の各サイトから新しいサイトへ、および新しいサイトから既存の各サイトへ相互に存在している必要があります。
- すべてのサイトで管理要求が正常に終了していることを確認します。管理要求がまだ実行されていない場合、保留になっている管理要求を手動で実行し、即時に操作を完了させることができます。

DDL による変更がマスター・サイトに伝播されない場合

新しいマスター・グループ・オブジェクトを作成しても、あるいはマスター定義サイトでマスター・グループ・オブジェクトの定義を変更しても、変更がマスター・サイトに伝播されない場合は、すべてのサイトで管理要求が正常に完了していることを確認します。管理要求が保留になっている場合は、手動で実行して即時に操作を完了させることができます。

レプリケーション API を通して DDL 文を実行すると、DDL を送信するユーザー名において DDL 文が実行されます。発行者のスキーマ以外のスキーマ内のオブジェクトに DDL 文を適用するときは、発行者に文を実行する適切な権限が必要です。また、文の中でスキーマを明示的に指定する必要があります。たとえば、レプリケーション管理者のユーザーが `DBMS_REPCAT.CREATE_MASTER_ REPOBJECT` プロシージャの `ddl_text` パラメータとして次のように指定するとします。

```
CREATE TABLE oe.new_employees AS SELECT * FROM hr.employees WHERE ...;
```

それぞれの表名にはスキーマ名が含まれるので、レプリケーション管理者が `oe` や `hr` または別のユーザーであっても、必要な権限があれば、この文は正しく機能します。

注意： 各スキーマ・オブジェクトの名前を適切なスキーマで修飾してください。

DML による変更が他のサイトに非同期で伝播されない場合

マスター・サイトのデータを更新し、その変更がレプリケーション環境の他のサイトに非同期で伝播されない場合は、次の処理を実行します。

- Oracle Enterprise Manager のレプリケーション・マネージメント・ツールを使用して、対応する遅延トランザクションが接続先にプッシュされているかどうかをチェックします。プッシュされていない場合、スケジュール・リンクがキューを接続先サイトにプッシュするまでの待ち時間をチェックできます。リンクによるスケジュール・プッシュを待機できない場合、遅延トランザクションを手動で実行できます。
- スケジュール・リンクのプッシュ間隔が経過しても対応する遅延トランザクションがプッシュされない場合は、リンクの対応するジョブをチェックします。
- 遅延トランザクションが接続先に伝播されても、エラーのために実行できない場合もあります。接続先サイトの DEFERROR のデータ・ディクショナリ・ビューでエラーの有無を調べます。

関連項目： 変更をレプリケートせずに表を変更する方法の詳細は、『Oracle9*i* レプリケーション・マネージメント API リファレンス 』を参照してください。レプリケート表のデータを手動で同期する際にこの方法が必要になる場合があります。

DML をレプリケート表に適用できない場合

レプリケート表を変更しようとしたときに deferred_rpc_quiesce 例外を受け取った場合は、レプリケート・オブジェクトが属するマスター・グループが静止または静止中になっています。先に進むには、レプリケーション管理者がマスター・グループのレプリケーション・アクティビティを再開する必要があります。

バルク更新および制約違反

1 つの UPDATE 文をレプリケート表に適用すると、ゼロ以上の行を更新できます。UPDATE 文を実行すると、ゼロ以上の更新要求が、更新される行ごとに 1 つずつ遅延実行用のキューに入れられます。Oracle では文末で制約のチェックをまとめて行うので、制約が存在する場合はこの機能が重要になります。たとえば、バルク更新では一意制約違反が発生しなくても、同じ順序で 1 行ずつ更新すると一意制約違反が発生する場合があります。

更新順序が重要である場合は、適切な順序で 1 行ずつ更新します。これによって、遅延トランザクション・キューにおける更新要求の順序を定義できます。

レプリケート・オブジェクトの再作成

パッケージ、プロシージャまたはビューなどのオブジェクトをマスター・グループに追加する場合は、オブジェクトの状態が有効である必要があります。オブジェクトの状態が無効の場合は、マスター・グループにオブジェクトを追加する前に、オブジェクトを再コンパイルするか、オブジェクトを削除して再作成してください。DBA_REPOBJECT のデータ・ディクショナリ・ビューでレプリケーション・オブジェクトの状態を調べます。

表のレプリケーション・サポートを生成できない場合

表のレプリケーション・サポートを生成すると、ローカル・サイトで内部トリガーがアクティブ化されます。レプリケーションに関連するほとんどのパッケージ（DBMS_REPCAT や DBMS_DEFER など）に対する EXECUTE 権限を、レプリケーション管理者およびレプリケート・オブジェクトを所有するユーザーに付与する必要があります。レプリケーション管理者が多くの一般的なレプリケーション・シナリオで必要とする権限は、レプリケーション・マネージメント・ツールの「Setup Wizard」および DBMS_REPCAT_ADMIN パッケージを使用して付与します。ただし、レプリケート・オブジェクトの所有者がレプリケーション管理者でない場合は、DBMS_DEFER に対する EXECUTE 権限をオブジェクトの所有者に対して明示的に付与する必要があります。

レプリケート・プロシージャまたはレプリケート・トリガーに関する問題

予期しない未解消の競合を発見した場合で、プロシージャ・レプリケーションと行レベルのレプリケーションが表に混在しているときは、プロシージャを注意深く検証し、レプリケート・プロシージャが競合の原因かどうかを確認します。次のチェックを実行します。

- ❑ プロシージャでの更新と行レベルの更新の順序競合でないことを確認してください。
- ❑ レプリケート・プロシージャが更新の実行前に表を EXCLUSIVE モードにロックしているかどうか、行レベルの更新の競合を回避するために他のメカニズムを使用しているかどうかをチェックします。
- ❑ レプリケート・プロシージャの最初で行レベルのレプリケーションを使用禁止にし、最後に再び使用可能にしていることをチェックします。
- ❑ プロシージャの実行時に例外が発生した場合も、行レベルのレプリケーションを再び使用可能にしていることを確認します。
- ❑ レプリケート・プロシージャがすべてのマスター・サイトで実行されたことをチェックします。

レプリケート表で定義したレプリケート・トリガーのすべてについて、同様のチェックを行う必要があります。

ON DELETE CASCADE および整合性制約に関する問題

ON DELETE CASCADE は、次のような構成のレプリケート表はサポートしていません。この例では、A、B、C の 3 つの表があると想定します。

- 表 B に、外部キー制約のために表 A のレコードに依存するレコードがあります。
- 表 C に、外部キー制約のために表 B のレコードに依存するレコードがあります。
- 表 C には、外部キー制約のために表 A のレコードに依存するレコードもあります。

このような構成で ON DELETE CASCADE を使用すると、レコードを削除した順序とは異なる順序で伝播される可能性があります。この場合、次のエラーが返されます。

```
ORA-02292: 整合性制約 (HR.DEPT_MGR_FK) に違反しました - 子レコードがあります
```

このようなエラーが発生した場合は、ON DELETE CASCADE を使用せず、レプリケーションを認識してこの機能を実行するトリガーを作成します。

遅延トランザクション・キューに関する問題の診断方法

サイトの遅延トランザクションが接続先にプッシュされない場合、その原因は次の各項で説明している問題である可能性があります。

- スケジュール・リンクのジョブのチェック
- 同期レプリケーション使用時の分散トランザクションの問題
- データベース・リンク指定の不備
- 不適切なレプリケーション・カタログ・ビュー

スケジュール・リンクのジョブのチェック

スケジュール・リンクを作成すると、対応するジョブがサイトのジョブ・キューに追加されます。スケジュール・リンクが定期的な間隔で遅延トランザクションをプッシュするように設定しているときに問題が検出された場合は、まずジョブ・キューに関する問題かどうかを確認する必要があります。

同期レプリケーション使用時の分散トランザクションの問題

同期レプリケーションを使用する場合、Oracle では分散トランザクションを使用して、そのトランザクションがリモート・サイトで正しくコミットされるようにします。分散トランザクションでは 2 フェーズ・コミットが使用されます。非同期レプリケーションでは 2 フェーズ・コミットは使用されません。

関連項目： 分散トランザクション使用時の問題の診断方法の詳細は、『Oracle9*i* データベース管理者ガイド』を参照してください。

データベース・リンク指定の不備

指定したリモート・サイトにトランザクションがプッシュされていない場合、トランザクションの接続先の指定方法に問題がある場合があります。スケジュール・リンクを作成するときは、完全なデータベース・リンク名を指定する必要があります。

不適切なレプリケーション・カタログ・ビュー

ビューの定義が不適切であると、遅延トランザクションが正しく動作しない場合があります。DEFCALLDEST ビューと DEFTRANDEST ビューの定義は、catdefer.sql と catrepc.sql とで異なります。レプリケーションを使用するときは常に、catrepc.sql の定義を使用する必要があります。catdefer.sql を（再）ロードする場合は、その後で catrepc.sql のビュー定義がロードされるようにしてください。

マテリアライズド・ビューに関連する問題の診断方法

レプリケーション・システムのマテリアライズド・ビュー・サイトでは様々な問題が発生します。次の項では、これらの問題とその解決方法をいくつか説明します。

- マテリアライズド・ビュー・サイトでのレプリケート・オブジェクトの作成に関する問題
- リフレッシュに関する問題
- リフレッシュに関する問題の高度なトラブルシューティング

マテリアライズド・ビュー・サイトでのレプリケート・オブジェクトの作成に関する問題

マテリアライズド・ビュー・サイトで新しいオブジェクトの作成に失敗した場合は、次の処理を実行します。

- 更新可能マテリアライズド・ビューの場合は、対応するマスター表またはマスター・マテリアライズド・ビューにマテリアライズド・ビュー・ログがあることを確認します。
- オブジェクトを作成する権限があるかどうかを確認します。マテリアライズド・ビューの場合は、マスター表またはマスター・マテリアライズド・ビューとそのマテリアライズド・ビュー・ログに対する SELECT 権限が必要です。詳細は、6-13 ページの「権限の割当て」を参照してください。
- 既存のマテリアライズド・ビューをマテリアライズド・ビュー・グループに追加する場合は、マテリアライズド・ビューを再作成してグループに追加します。
- 高速リフレッシュを実行する主キーまたは副問合せマテリアライズド・ビューを作成する場合は、マスター表やマスター・マテリアライズド・ビューのマテリアライズド・ビュー・ログに主キーがロギングされていることを確認します。

- 高速リフレッシュを実行する ROWID マテリアライズド・ビューを作成する場合は、マスター表のマテリアライズド・ビュー・ログに ROWID がロギングされていることを確認します。
- 副問合せマテリアライズド・ビューの場合は、必要な列がマテリアライズド・ビュー・ログに追加されていることを確認します。詳細は、6-16 ページの「マテリアライズド・ビュー・ログへの列のロギング」を参照してください。
- 高速リフレッシュを実行するマテリアライズド・ビューと関連するすべての表に対して、マテリアライズド・ビュー・ログが存在することを確認します。マテリアライズド・ビューに副問合せが含まれている場合は、副問合せで参照される表のそれぞれに対してマテリアライズド・ビュー・ログが必要です。

配置テンプレートのオフライン・インスタンシエーションの実行に関する問題

配置テンプレートをオフラインでインスタンス化しようとしたときに、一時表領域でエクステントを初期化できないというエラーを受け取った場合に、一時データベースが自動的に拡張されるように、一時データベースのデータ・ファイルを調整する必要があります。

たとえば、次の文を発行してデータ・ファイルを調整します。

```
ALTER DATABASE TEMPFILE '/u02/oracle/rbdb1/temp.dbf'
    AUTOEXTEND ON
    NEXT 10M;
```

この調整を行った後、マテリアライズド・ビュー・サイトで配置テンプレートをオフラインでインスタンス化します。

リフレッシュに関する問題

次の項では、マテリアライズド・ビューのリフレッシュに関するいくつかの一般的な問題について説明します。

一般的な問題

次のようないくつかの一般的な要因で、マテリアライズド・ビューのグループの自動リフレッシュが実行されない場合があります。

- マテリアライズド・ビュー・データベースでのジョブ・キュー・プロセスの欠落
- ネットワークまたはサーバー障害の影響
- サーバー・シャットダウンの影響

マテリアライズド・ビューのリフレッシュ・グループで問題が発生したときは、前述の状況が原因かどうかを確認します。

自動リフレッシュの再試行

グループの自動リフレッシュが失敗すると、グループはリフレッシュ実行予定のままになります。次の条件に基づいて、グループの自動リフレッシュが再試行されます。

- グループのリフレッシュを初回は 1 分後、次は 2 分後、その次は 4 分後というように、失敗するたびに再試行間隔を倍にします。
- 再試行間隔はリフレッシュ間隔自体より長い時間にはなりません。
- 最高 16 回まで自動リフレッシュを再試行します。

リフレッシュ・グループのリフレッシュを 16 回再試行してもエラーが発生する場合は、グループは破損しているとみなされます。Schema Manager の「Refresh Group」プロパティ・シートの「General」ページを参照すると、いつリフレッシュ・グループが破損したかがわかります。`USER_REFRESH` および `USER_REFRESH_CHILDREN` データ・ディクショナリ・ビューの `BROKEN` 列を問い合せて、リフレッシュ・グループの現在の状態を知ることもできます。

マテリアライズド・ビューのリフレッシュ・グループの破損によるエラーは、トレース・ファイルに記録されます。リフレッシュ・グループが正常にリフレッシュされない原因となった問題を修正し、グループを手動でリフレッシュする必要があります。この結果、破損フラグがリセットされ、自動リフレッシュが実行できるようになります。

> **関連項目：** マテリアライズド・ビュー・トレース・ファイルの名前の形式は *jn* で、*n* はオペレーティング・システム固有です。各システムにおける名前は、オペレーティング・システム固有の Oracle ドキュメントを参照してください。

新規マテリアライズド・ビュー・サイトでの高速リフレッシュ・エラー

新規マテリアライズド・ビュー・サイトでマテリアライズド・ビュー作成中に、マスター表またはマスター・マテリアライズド・ビューのマテリアライズド・ビュー・ログがパージされる場合があります。これが発生すると、次のエラーが検出される場合があります。

`ORA-12004 REFRESH FAST` はマテリアライズド・ビュー *`materialized_view_name`* に使用できません。
`ORA-12034` *`materialized_view_name`* のマテリアライズド・ビュー・ログは最新のリフレッシュよりも新しいものです。

> **関連項目：** この問題を回避する方法については、6-20 ページの「新規マテリアライズド・ビュー・サイトを追加するときの問題の回避」を参照してください。

マテリアライズド・ビューが繰り返しリフレッシュされる場合

マテリアライズド・ビュー・グループのリフレッシュが継続的に繰り返される場合、グループのリフレッシュ間隔をチェックします。グループの自動リフレッシュ間隔は、リフレッシュを開始する前に評価されます。グループのリフレッシュ間隔がグループのすべてのマテリアライズド・ビューのリフレッシュに要する時間よりも短い場合、ジョブ・キュー・プロセスで未処理ジョブのキューがチェックされるたびに、グループ・リフレッシュが繰り返し開始されます。

マテリアライズド・ビュー・ログが大きくなりすぎる場合

マスター・サイトまたはマスター・マテリアライズド・ビュー・サイトでマテリアライズド・ビュー・ログが大きくなりすぎる場合は、ネットワークまたはサイトの障害のためにマテリアライズド・ビューが削除されたことが、マスター・サイトまたはマスター・マテリアライズド・ビュー・サイトで認識されているかどうかをチェックします。マテリアライズド・ビュー・ログの部分的なパージ、あるいは未使用のマテリアライズド・ビュー・サイトの登録の解除が必要になる場合があります。

> **関連項目：** マテリアライズド・ビュー・ログの管理方法の詳細は、『Oracle9*i* レプリケーション・マネージメント API リファレンス』を参照してください。

リフレッシュに関する問題の高度なトラブルシューティング

マテリアライズド・ビューのリフレッシュに関する問題が発生している場合、次の処理を実行します。

- DBA_REFRESH_CHILDREN ビューに表示されている NEXT_DATE 値をチェックして、リフレッシュがスケジュールされているかどうかを判断します。
- リフレッシュ間隔が経過している場合は、DBA_REFRESH ビューで、マテリアライズド・ビューのリフレッシュが対応付けられたジョブ番号をチェックして、ジョブ・キューに関する問題の診断を行います。
- 実行中のジョブ・キュー・プロセスがあるかどうかをチェックします。JOB_QUEUE_PROCESSES 初期化パラメータをチェックし、DBA_JOBS_RUNNING ビューを問い合せ、オペレーティング・システムを使用してジョブ・キュー・プロセスが稼働中かどうかをチェックします。
- 2 つのマテリアライズド・ビュー間のマスター・ディテール関係を定義すると、エラーが発生することもあります。マスター・ディテール関係は、宣言参照整合性の制約を使用して、マスター表に対してのみ定義します。その後、関連するマテリアライズド・ビューを同じリフレッシュ・グループ内に入れて、この関係を保ちます。ただし、遅延（遅延可能）制約はマテリアライズド・ビューに対して定義します。

- マテリアライズド・ビューのマスター・サイトまたはマスター・マテリアライズド・ビュー・サイトに未解決の競合が記録されている場合、REFRESH_AFTER_ERRORS を TRUE に設定しないとマテリアライズド・ビューをリフレッシュできません。このパラメータは、マテリアライズド・ビューのリフレッシュ・グループの作成または変更時に設定できます。レプリケーション・マネージメント・ツールのプロパティ・シートに、これと対応するパラメータがあります。
- 同じリフレッシュ・グループ内のマテリアライズド・ビューでは、行は 1 つのトランザクションで更新されます。このようなトランザクションは非常に大きくなる可能性があるので、大きなロールバック・セグメント（リフレッシュ中に使用するように指定したロールバック・セグメントを含む）がマテリアライズド・ビュー・サイトで必要になります。あるいは、リフレッシュをより頻繁に実行してトランザクションのサイズを小さくする必要があります。
- Oracle エラー ORA-12004 が発生した場合は、マテリアライズド・ビュー・ログを管理するためにマスター・サイトでロールバック・セグメントが不足したか、マテリアライズド・ビュー・ログの情報が古い可能性があります。たとえば、マテリアライズド・ビュー・ログがパージまたは再作成された可能性があります。
- 1 つのマテリアライズド・ビューの完全リフレッシュでは、内部的に TRUNCATE 機能が使用されるので、処理速度が向上し、ロールバック・セグメントの必要量が少なくなります。ただし、マテリアライズド・ビューが完全リフレッシュされるまで、マテリアライズド・ビューには一時的にデータが表示されないことがあります。複数のマテリアライズド・ビュー（リフレッシュ・グループなど）をリフレッシュするときは、TRUNCATE 機能は使用されません。
- マスター表を再編成するとき（たとえば、システム・リソースの再利用などの目的で）は、マスター表に TRUNCATE 機能を実行して、ROWID マテリアライズド・ビューの完全リフレッシュを実行する必要があります。そうしないと、マテリアライズド・ビューがマスター表への誤った ROWID を持つことになります。マスター表の再編成には、DBMS_MVIEW パッケージの BEGIN_TABLE_REORGANIZATION および END_TABLE_REORGANIZATION プロシージャを使用します。詳細は、『Oracle9*i* レプリケーション・マネージメント API リファレンス』を参照してください。
- リフレッシュの実行中にエラー ORA-00942（表またはビューが存在しません）が発生した場合は、データベース・リンクをチェックし、マスター表またはマスター・マテリアライズド・ビューとマテリアライズド・ビュー・ログに対して必要な権限を持っていることを確認します。
- 高速リフレッシュが成功した後で障害が発生した場合、次の点をチェックします。
  - マテリアライズド・ビュー・ログの切捨て、パージまたは削除が実行されたかどうか
  - マテリアライズド・ビュー・ログに対する必要な権限を持っているかどうか
- 強制リフレッシュに非常に長い時間を要する場合は、リフレッシュ時に使用するマテリアライズド・ビュー・ログが削除されていないかチェックします。

- BUILD DEFERRED を使用して作成したマテリアライズド・ビューの最初の高速リフレッシュが失敗した場合は、その他の問題をチェックする前に、前の完全リフレッシュが成功していたことを確認します。

  関連項目： マテリアライズド・ビュー・ログの管理方法の詳細は、『Oracle9*i* レプリケーション・マネージメント API リファレンス』を参照してください。

B

列長セマンティクスと Unicode

この付録には、列長セマンティクスおよび Unicode に対するレプリケーション・サポートに関する情報が含まれています。この付録の内容は、次のとおりです。

- レプリケーション・サイトおよび表の列に関する列長セマンティクス
- マルチマスター環境での列長セマンティクス・サポート
- マテリアライズド・ビューでの列長セマンティクス・サポート
- DDL 伝播と列長セマンティクス
- Unicode に対するレプリケーション・サポート

関連項目： 長さセマンティクスおよび Unicode の詳細は、次のマニュアルを参照してください。

- 『Oracle9*i* Database グローバリゼーション・サポート・ガイド』
- 『Oracle9*i* SQL リファレンス』

レプリケーション・サイトおよび表の列に関する列長セマンティクス

列長セマンティクスにより、列の長さをバイト単位に指定するか文字単位に指定するかが決まります。列の長さをバイト単位に指定する場合は BYTE を、文字単位に指定する場合は CHAR を使用します。CHAR 列セマンティクスは、コードポイント長セマンティクスとも呼ばれます。

キャラクタ・セットによっては 1 文字に複数バイトを必要とするものがあるため、列に 10 BYTE と指定すると、実際に格納される文字数が 10 文字より少ないことがありますが、10 CHAR と指定すると、どのキャラクタ・セットでも列に 10 文字が格納されます。CHAR 列セマンティクスを指定できるのは、Oracle9*i* データベースのみです。

Oracle9*i* データベースの長さセマンティクスは NLS_LENGTH_SEMANTICS 初期化パラメータを使用して設定します。すべての VARCHAR2 列および CHAR 列で、この初期化パラメータにデフォルトとして指定された値が使用されます。この初期化パラメータを設定しない場合、デフォルト設定は BYTE です。

それぞれの列で、データベースの列長セマンティクスを上書きできます。たとえば、サイトの列長セマンティクスが CHAR の場合でも、CREATE TABLE 文または ALTER TABLE 文を使用して、個々の列の長さのセマンティクスに BYTE を指定できます。

次の文では表を作成し、列長をバイト単位に指定します。

```
CREATE TABLE byte_col (a VARCHAR2(10 BYTE));
```

次の文では表を作成し、列長を文字単位に指定します。

```
CREATE TABLE char_col (a VARCHAR2(10 CHAR));
```

注意： CHAR 列セマンティクスを指定するには、データベースの互換性のレベルが 9.0.1 以上の必要があります。互換性のレベルは、COMPATIBLE 初期化パラメータによって制御されます。

マルチマスター環境での列長セマンティクス・サポート

マスター・グループ内のすべてのマスター・サイトの列長セマンティクスは同じにする必要があり、マスター表内の個々の列長セマンティクスはすべてのマスター・サイトで同じにする必要があります。マスター定義サイトのマスター・グループに表があり、その表を新規マスター・サイトにレプリケートする場合は、次のいずれかの方法で新規サイトに表を作成できます。

- 新規マスター・サイトをマスター・グループに追加するときに、Advanced Replication によって新規マスター・サイトにその表を生成することを指定します。
- 新規マスター・サイトをマスター・グループに追加する前に、そのマスター・サイトで表を事前に手動で作成します。

以降の項では、この 2 つの表の作成方法のそれぞれに対する列長セマンティクス・サポートを説明します。

Advanced Replication によって生成された表に対する列長セマンティクス・サポート

Advanced Replication で新規マスター・サイトに表を生成するように指定し、CHAR 列セマンティクスを使用している場合は、マスター定義サイトと新規マスター・サイトの両方で Oracle9*i* が稼働している必要があります。BYTE 列セマンティクスを指定する場合は、これらのサイトで Oracle8*i* 以前のリリースの Oracle が稼働していてもかまいません。

このサポートについて、表 B-1 に要約します。

表 B-1 生成された表に対する列長セマンティクス・サポート

マスター定義サイトのリリース	マスター定義サイトの列セマンティクス	新規マスター・サイトのリリース	新規マスター・サイトの列セマンティクス
9.0 以上	CHAR	9.0 以上	CHAR
9.0 以上	CHAR	8.1 以下	サポートされていません
任意のリリース	BYTE	任意のリリース	BYTE

事前作成された表に対する列長セマンティクス・サポート

新規マスター・サイトに表を事前作成するときに、CHAR 列セマンティクスを使用する場合は、マスター定義サイトと新規マスター・サイトの両方で Oracle9*i* が稼働している必要があります。BYTE 列セマンティクスを指定する場合は、これらのサイトで Oracle8*i* 以前のリリースの Oracle が稼働していてもかまいません。

また、表を手動で事前に作成しているため、新規マスター表内の列には、マスター定義サイトでの表の列とは異なる列長セマンティクスを指定している可能性があります。マスター表内の列は各マスター・サイトで同じ列セマンティクスを使用する必要があるため、その場合はエラーが発生します。

このサポートについて、表 B-2 に要約します。

表 B-2 事前作成された表に対する列長セマンティクス・サポート

マスター定義サイトのリリース	マスター定義サイトの列セマンティクス	新規マスター・サイトのリリース	新規マスター・サイトの列セマンティクス	サポート
9.0 以上	CHAR	9.0 以上	CHAR	あり
9.0 以上	CHAR	9.0 以上	BYTE	なし
9.0 以上	BYTE	9.0 以上	CHAR	なし
9.0 以上 (マルチバイト・キャラクタ・セット)	CHAR	8.1 以下	BYTE	なし
9.0 以上 (シングルバイト・キャラクタ・セット)	CHAR	8.1 以下	BYTE	あり
8.1 以下	BYTE	9.0 以上 (マルチバイト・キャラクタ・セット)	CHAR	なし
8.1 以下	BYTE	9.0 以上 (シングルバイト・キャラクタ・セット)	CHAR	あり
任意のリリース	BYTE	任意のリリース	BYTE	あり

マテリアライズド・ビューでの列長セマンティクス・サポート

マテリアライズド・ビューを作成するとき、マテリアライズド・ビュー内の列長セマンティクスは次のように決定されます。

- マスター列が BYTE または CHAR として明示的に指定されている場合は、マテリアライズド・ビュー内の列はその明示的な指定を保持します。次の例では、a 列に対して CHAR 列セマンティクスが明示的に指定されています。

  ```
  CREATE TABLE char_col (a VARCHAR2(10 CHAR));
  ```

- マスター表内の列が明示的に指定されない場合は、マテリアライズド・ビュー内の列はマテリアライズド・ビュー・サイトのデフォルトの列長セマンティクスを使用します。次の例では、a 列の列長セマンティクスが明示的に指定されていません。

  ```
  CREATE TABLE char_col (a VARCHAR2(10));
  ```

Oracle9*i* のマスターに明示的な CHAR 指定の列があり、Oracle8*i* 以前のリリースを実行しているマテリアライズド・ビュー・サイトがこのマスターに基づいてマテリアライズド・ビューを作成しようとすると、マテリアライズド・ビューの作成は失敗します。

ビルトインのコンテナ表を含むマテリアライズド・ビュー

マテリアライズド・ビュー・サイトでマテリアライズド・ビューを作成する前にコンテナ表を事前に作成した場合、コンテナ表の列長セマンティクスはマスターの列長セマンティクスと一致する必要があります。列長セマンティクスが一致しない場合は、マテリアライズド・ビューの作成中に ORA-12060 エラーが返されます。マテリアライズド・ビューの表を事前に作成するには、CREATE MATERIALIZED VIEW 文の ON PREBUILT TABLE 句を使用します。

> **関連項目：** CREATE MATERIALIZED VIEW 文の ON PREBUILT TABLE 句の詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

更新可能マテリアライズド・ビューに対する列長セマンティクス・サポート

更新可能マテリアライズド・ビューの列長セマンティクスがマテリアライズド・ビューのマスターの列長セマンティクスに一致する場合は、次の操作が常にサポートされます。

- 更新可能マテリアライズド・ビューのリフレッシュ
- マテリアライズド・ビューに加えられた DML 変更のマスターへのプッシュ

ただし、列長セマンティクスが一致せず、マスターが Oracle9*i* の場合は、マテリアライズド・ビューをマテリアライズド・ビュー・グループに追加しようとしたときにエラーが発生します。マテリアライズド・ビューが更新可能マテリアライズド・ビューになるには、マテリアライズド・ビュー・グループに属している必要があります。レプリケーション・マネージメント API を使用する場合は、DBMS_REPCAT パッケージ内の CREATE_MVIEW_

REPOBJECT プロシージャを実行して、マテリアライズド・ビューをマテリアライズド・ビュー・グループに追加します。

表 B-3 に、更新可能マテリアライズド・ビューに対する列長セマンティクスのサポートについて要約します。

表 B-3 更新可能マテリアライズド・ビューに対する列長セマンティクス・サポート

マスター・サイトのリリース	マスター・サイトの列セマンティクス	マテリアライズド・ビュー・サイトのリリース	マテリアライズド・ビュー・サイトの列セマンティクス	更新可能マテリアライズド・ビューのサポート
9.0 以上	CHAR	9.0 以上	CHAR	あり
9.0 以上	CHAR	9.0 以上	BYTE	なし
9.0 以上	BYTE	9.0 以上	CHAR	なし
9.0 以上 (マルチバイト・キャラクタ・セット)	CHAR	8.1 以下	BYTE	なし
9.0 以上 (シングルバイト・キャラクタ・セット)	CHAR	8.1 以下	BYTE	あり
8.1 以下	BYTE	9.0 以上	CHAR	あり
任意のリリース	BYTE	任意のリリース	BYTE	あり

注意： 表 B-3 に示されているマスター・サイトは、マルチマスター・レプリケーション環境のマスター・サイトか、マスター・マテリアライズド・ビュー・サイトのいずれかです。

DDL 伝播と列長セマンティクス

DBMS_REPCAT パッケージを使用すると、新規レプリケート表を作成するデータ定義言語（DDL）文、あるいは既存のレプリケート表に列を追加する DDL 文を伝播できます。このような DDL 文により作成された新規列で CHAR 列セマンティクスを使用する場合は、CHAR 列セマンティクスを明示的に指定するようにします。指定しない場合、レプリケーション・サイト自体でデフォルトとして CHAR 列セマンティクスを設定していても、列は常に BYTE 列セマンティクスになります。

DDL 文を伝播するには、DBMS_REPCAT パッケージ内の次のプロシージャを使用します。

- ALTER_MASTER_REPOBJECT
- CREATE_MASTER_REPOBJECT
- EXECUTE_DDL

関連項目： CHAR 列セマンティクスの明示的な指定方法の詳細は、B-2 ページの「レプリケーション・サイトおよび表の列に関する列長セマンティクス」を参照してください。

Unicode に対するレプリケーション・サポート

Unicode は、任意の言語からの情報を 1 つのキャラクタ・セットを使用して格納できる汎用のエンコードされたキャラクタ・セットです。Unicode は使用されるプラットフォーム、プログラムまたは言語に関係なく、すべてのキャラクタにそれぞれ一意のコード値を提供します。Unicode はマルチマスター・レプリケーション環境およびマテリアライズド・ビュー・レプリケーション環境の両方でサポートされます。Oracle9*i* では、NCHAR または NVARCHAR2 データ型として指定される列は、すべて Unicode 形式で格納されます。

マスター・サイトとマテリアライズド・ビュー・サイトの両方で、NCHAR または NVARCHAR2 データ型を使用すると、異なるリリースの Oracle が稼働する環境でのレプリケーションが可能になります。ただし、レプリケーション・サイトの一方が Oracle8*i* 以前のリリースで、可変幅キャラクタ・セットを使用している場合は、レプリケーションをお薦めしません。この場合はデータが失われる可能性があります。

表 B-4 に、レプリケーションが推奨される場合について要約します。

表 B-4 グローバリゼーション・サポート・キャラクタ・セットに対するレプリケーション・サポート

NCHAR 列または NVARCHAR2 列を含むローカル・データベースのリリース	NCHAR 列および NVARCHAR2 列を含むリモート・データベースのリリース	レプリケーションの推奨
9.0 以上 （Unicode 形式の格納）	9.0 以上 （Unicode 形式の格納）	推奨します
8.1 以下 （固定幅または可変幅の各国語キャラクタ・セット形式）	8.1 以下 （固定幅または可変幅の各国語キャラクタ・セット形式）	推奨します
9.0 以上 （Unicode 形式の格納）	8.1 以下 （可変幅の各国語キャラクタ・セット形式）	推奨しません
9.0 以上 （Unicode 形式の格納）	8.1 以下 （固定幅の各国語キャラクタ・セット形式）	推奨します
8.1 以下 （可変幅の各国語キャラクタ・セット形式）	9.0 以上 （Unicode 形式の格納）	推奨しません
8.1 以下 （固定幅の各国語キャラクタ・セット形式）	9.0 以上 （Unicode 形式の格納）	推奨します

注意： 表 B-4 でレプリケーションを推奨しないと示されている場合、2 つのサイト間でレプリケーションを設定してもエラーは検出されません。ただし、後でデータが失われる可能性があります。データが失われた場合は、エラーが発生します。

NCLOB データ型列のレプリケーション

NCLOB データ型は常に固定幅です。したがって、NCLOB データ型の列のレプリケーションは制限なしでサポートされます。

索引

数字

1 対多副問合せ
　マテリアライズド・ビュー，3-20

A

Advanced Replication
　Oracle Real Application Clusters との比較，2-6
AND 演算子
　単純副問合せマテリアライズド・ビュー，3-24

B

BYTE キャラクタ・セマンティクス，B-2

C

CHAR キャラクタ・セマンティクス，B-2
COMPATIBLE 初期化パラメータ，6-5
Copy Template Wizard，7-23
CREATE TYPE 文
　OID 句，2-14，3-34

D

DBA_MVIEW_REFRESH_TIMES ビュー，3-43
DBA_REGISTERED_MVIEWS ビュー，3-43
DBA_REPCATLOG ビュー，2-32
DBA_TYPE_VERSIONS
　レプリケーション，2-13，3-33
DBMS_DEFER_SYS パッケージ
　SCHEDULE_PUSH プロシージャ，6-22，6-23
DBMS_MVIEW パッケージ
　EXPLAIN_MVIEW プロシージャ，3-25
　REGISTER_MVIEW プロシージャ，3-44
　UNREGISTER_MVIEW プロシージャ，3-44
DBMS_REFRESH パッケージ
　CHANGE プロシージャ，4-20
DBMS_REPCAT パッケージ，2-32，2-34
　COMPARE_OLD_VALUES プロシージャ
　　競合解消，5-36
　CREATE_MVIEW_REPGROUP プロシージャ，3-5
　CREATE_MVIEW_REPOBJECT プロシージャ，3-6
　DO_DEFERRED_REPCAT_ADMIN プロシージャ，
　　2-32，2-34
　REPCAT_IMPORT_CHECK プロシージャ，6-36
　SEND_OLD_VALUES プロシージャ
　　競合解消，5-36
DDL 文
　レプリケーション，1-17
　　問題のトラブルシューティング，A-3
DELAY_SECONDS パラメータ，2-47
Deployment Template Wizard，7-19
DML 文
　レプリケーション
　　問題のトラブルシューティング，A-4

E

EXISTS 演算子
　副問合せを使用するマテリアライズド・ビュー，
　　3-24
EXPLAIN_MVIEW プロシージャ，3-25

G

GLOBAL_NAMES 初期化パラメータ，6-5

I

INIT.ORA パラメータ，「初期化パラメータ」を参照

J

JOB_QUEUE_PROCESSES 初期化パラメータ，2-32，6-5，6-13

L

LOB
レプリケーション・サポート，6-3
レプリケート表で許可されている LOB，6-2

M

Materialized View Group Wizard，7-18

N

NCHAR データ型
Unicode，B-7
NCLOB データ型
長さセマンティクス，B-8
NLS_LENGTH_SEMANTICS 初期化パラメータ，B-2
NOROWDEPENDENCIES 句，2-51
NVARCHAR データ型
Unicode，B-7
n-way レプリケーション，「マルチマスター・レプリケーション」を参照，1-6

O

OF object_type 句
オブジェクト・マテリアライズド・ビュー，3-36
OID 句
CREATE TYPE 文，2-14，3-34
ON COMMIT REFRESH 句
CREATE MATERIALIZED VIEW，3-32
ON DELETE CASCADE 句
レプリケーションに関する問題のトラブルシューティング，A-6
ON PREBUILT TABLE 句，3-24
長さセマンティクス，B-5
OPEN_LINKS 初期化パラメータ，2-22，6-6
OPS，「Oracle Real Application Clusters」を参照
Oracle Parallel Server「Oracle Real Application Clusters」を参照，2-6
Oracle Real Application Clusters
Advanced Replication との比較，2-6
レプリケーションとの比較，6-33
Oracle レプリケーション・マネージメント・ツール，1-16，7-1
「Topology」タブ，7-10
インタフェース，7-4
ウィザード，7-14
Copy Template，7-23
Deployment Template，7-19
Materialized View Group，7-18
Setup，7-15
Template Script Generation Wizard，7-21
最初のログイン，7-3
使用例，7-2
ナビゲータ・ペイン，7-5
右ペイン，7-9
OR 演算子
副問合せを使用するマテリアライズド・ビュー，3-24

P

Parallel Server，「Oracle Real Application Clusters」を参照，2-6
PARALLEL_AUTOMATIC_TUNING 初期化パラメータ，2-45，6-6
PARALLEL_MAX_SERVERS 初期化パラメータ，2-45，6-6
PARALLEL_MIN_SERVERS，6-6
PARALLEL_MIN_SERVERS 初期化パラメータ，2-45，6-6
PCTFREE パラメータ
複合マテリアライズド・ビューの値，3-59
PCTUSED パラメータ
複合マテリアライズド・ビューの値，3-59
peer-to-peer レプリケーション，「マルチマスター・レプリケーション」を参照
PROCESSES 初期化パラメータ，6-6

R

REF
マテリアライズド・ビュー，3-40
レプリケーション，2-17，3-40

REPLICATION_DEPENDENCY_TRACKING 初期化パラメータ，6-7
ROWDEPENDENCIES 句，2-51
表の作成，6-4
ROWID
ROWID マテリアライズド・ビュー，3-9

S

Setup Wizard，7-15
SHARED_POOL_SIZE 初期化パラメータ，6-7

T

Template Script Generation Wizard，7-21

U

Unicode
NCHAR データ型，B-7
NVARCHAR データ型，B-7
レプリケーション，B-7
USER_REFRESH_CHILDREN ビュー，A-9
USER_REFRESH ビュー，A-9
USLOG$_，3-52
UTL_FILE_DIR 初期化パラメータ，6-7

V

VARRAY
マテリアライズド・ビュー，3-38
制限事項，3-40
レプリケーション，2-16，3-38

W

WHERE 句
配置テンプレート，4-8
WITH ROWID 句
REF，3-42

あ

アカウント
マテリアライズド・ビューのための作成，6-11

い

依存性
最小化，2-52
順序付け
レプリケート・トランザクション，2-50
追跡
パラレル伝播，2-50
依存性の追跡
行レベル，2-51，6-4
インスタンス化
配置テンプレート，1-13，4-9
インポート
マテリアライズド・ビュー，3-50
マテリアライズド・ビュー・ログ，3-50
レプリケーションのチェック，6-36

う

上書きによる競合解消方法，5-23

え

エラー
エラー・キュー，2-29
競合，5-20

お

オブジェクト
レプリケート
再作成，A-5
オブジェクト識別子
レプリケーションのための一致，2-12，3-33
オブジェクト表
マテリアライズド・ビュー，3-36
マテリアライズド・ビュー・ログ，3-49
列グループ，5-20
レプリケーション，2-15
オブジェクト・マテリアライズド・ビュー，3-36
OID の保持，3-37
オブジェクト・リレーショナル・モデル
マテリアライズド・ビュー，3-32
レプリケーション，2-12，3-32

か

外部キー
　レプリケート表，6-2
書込み可能マテリアライズド・ビュー，3-7
拡張、可用性，5-35
加算による競合解消方法，5-24
可用性
　拡張，5-35
完全リフレッシュ，1-12，3-59
管理要求，2-31
　状態，2-34
　　AWAIT_CALLBACK，2-34
　　DO_CALLBACK，2-34
　　ERROR，2-34
　　READY，2-34
管理要求キュー，2-34

き

機能、新機能，xv
キャラクタ・セット
　レプリケーション，B-1
行
　競合検出時の識別，2-53
競合
　一意性，5-3
　　回避，5-16
　エラー・キュー，5-20
　回避，5-15
　検出，2-42，2-53，5-4
　　行の識別，2-53，5-5
　更新，5-3
　　回避，5-17
　削除，5-3
　　回避，5-16
　順序
　　回避，5-17
　プロシージャ・レプリケーション，1-20
競合解消，2-54
　アーキテクチャ，5-19
　一意性競合，5-3
　一意性競合の解消方法，5-32
　上書きによる方法，5-23
　エラー・キュー，5-20
　解消方法のデータ収束プロパティ，5-24
　概念，5-2
　加算による方法，5-24
　競合の回避，5-15
　競合の検出，5-4
　競合のタイプ，5-3
　更新可能マテリアライズド・ビュー
　　列のサブセット化，5-9
　更新の競合，5-3
　更新の競合の解消方法，5-21
　最小値による方法，5-28
　最新のタイムスタンプによる方法，5-21
　最大値による方法，5-27
　サイト名の追加による方法，5-33
　サイト優先順位による方法，5-31
　　予備の方法，5-22，5-27
　削除の競合，5-3
　削除の競合解消方法，5-34
　順序の追加による方法，5-33
　静止なしの設定，5-35
　多重化マテリアライズド・ビュー，5-6
　データ要件，5-2
　同期伝播，2-42
　動的サイト所有権，5-15
　トランザクションの順序，5-4
　ネストした表，5-10
　廃棄による方法，5-26，5-34
　パフォーマンス，5-34
　　最小限の通信，5-34
　　古い値の送信，5-36
　　古い値の比較，5-36
　ビルトインの方法，5-5
　プライマリ・サイト所有権，5-15
　平均による方法，5-25
　最も古いタイムスタンプによる方法，5-27
　優先グループによる方法，5-29
　列グループ，5-19
　　オブジェクト表，5-20
　　シャドウ，5-20
　　データ整合性，5-19
　　ネストした表，5-20
　　列オブジェクト，5-20
　列のサブセット化
　　更新可能マテリアライズド・ビュー，5-9
　レプリケーション，1-18，2-10
強制リフレッシュ，1-12，3-61
行のサブセット化，1-11
　配置テンプレート，4-22
　マテリアライズド・ビュー，3-18

く

グループ所有者
　マテリアライズド・ビュー・グループ，3-55

け

計画
　レプリケーション用，6-1
権限
　マテリアライズド・ビュー，3-14，6-13

こ

更新可能マテリアライズド・ビュー，1-9，3-5
　競合解消，5-9
　長さセマンティクス，B-5
　マテリアライズド・ビュー・グループ，3-54
　列のサブセット化，5-9
更新可能マテリアライズド・ビュー・ログ，3-52
　トリガー，3-53
高速リフレッシュ，1-12，3-59
　使用可能かどうかの判断，3-25
　ダイレクト・パス・ロード，3-59
　多重化マテリアライズド・ビュー，3-60
　問題の回避，6-20
コレクション
　マテリアライズド・ビュー，3-38
　　制限事項，3-40
　レプリケーション，2-16，3-38

さ

最小限の通信，2-47
　競合解消，5-34
　　静止，5-35
最小値による競合解消方法，5-28
最新のタイムスタンプ
　競合解消方法，5-21
最大値による競合解消方法，5-27
サイト名の追加による競合解消方法，5-33
サイト優先順位による競合解消方法，5-31
　予備の方法としての，5-22，5-27
索引
　外部キーに対する，6-2
　マテリアライズド・ビュー・サイト，3-52
　レプリケーション，2-24
索引タイプ
　レプリケーション，2-27
サブセット化
　マテリアライズド・ビュー，3-18
　　列オブジェクト，3-35
参照整合性
　自己参照型制約，2-24

し

システム変更番号
　行レベル，2-51，6-4
シノニム，2-27
　レプリケーション，2-27
シャドウ列グループ，5-20
主キー制約
　マテリアライズド・ビュー，3-7，3-24
　レプリケート表，6-2
受信者
　レプリケーション，2-19
順序
　レプリケーション，2-28
順序の追加による競合解消方法，5-33
初期化パラメータ
　COMPATIBLE，6-5
　GLOBAL_NAMES，6-5
　JOB_QUEUE_PROCESSES，2-32，6-5
　NLS_LENGTH_SEMANTICS，B-2
　OPEN_LINKS，2-22，6-6
　PARALLEL_AUTOMATIC_TUNING，2-45，6-6
　PARALLEL_MAX_SERVERS，2-45，6-6
　PARALLEL_MIN_SERVERS，2-45
　PROCESSES，6-6
　REPLICATION_DEPENDENCY_TRACKING，6-7
　SHARED_POOL_SIZE，6-7
　UTL_FILE_DIR，6-7
　編集，6-13
　レプリケーション，6-4
ジョブ
　スケジュール・リンクのチェック，A-6
　レプリケーション，2-32
ジョブ・キュー，2-30
ジョブ・キュー・プロセス
　レプリケーション，6-13
新機能，xv

す

垂直パーティション化，「列のサブセット化」を参照
水平パーティション化，「行のサブセット化」を参照
スキーマ
　マテリアライズド・ビューのための作成，6-11
スケジュール・パージ
　ガイドライン，6-24
　定期的なパージ，6-25
スケジュール・リンク，1-15，6-13
　ガイドライン，6-21
　シリアル伝播，6-27
　定期的なプッシュ，6-22
　パラレル伝播，6-27
　連続的なプッシュ，6-23
ストア / フォワード・レプリケーション，「非同期レプリケーション」を参照
スナップショット，「マテリアライズド・ビュー」を参照

せ

制限事項
　プロシージャ・レプリケーション，2-25
静止，1-7，2-30
　最小限の通信，5-35
　静止なしの競合解消方法の設定，5-35
制約
　参照
　　自己参照型，2-24
制約違反，A-4
セキュリティ
　配置テンプレート，4-8
接続修飾子，2-22
　問題の診断方法，A-2

た

耐障害性，6-32
　Oracle Real Application Clusters，6-33
　実装，6-35
　設計上の考慮事項，6-34
大量配布，3-4，4-2
ダイレクト・パス・ロード
　高速リフレッシュ，3-59
多重化マテリアライズド・ビュー，1-8，3-26
　競合解消，5-6
　高速リフレッシュ，3-60
　制限事項，3-31
　マスター・マテリアライズド・ビュー，3-27
　　制限事項，3-31
　マテリアライズド・ビュー・グループ
　　所有権，3-32
　レベル，3-26
多対 1 副問合せ
　マテリアライズド・ビュー，3-19
多対多副問合せ
　マテリアライズド・ビュー，3-22
単純マテリアライズド・ビュー，3-12

ち

遅延トランザクション・キュー，1-6，2-29
　スケジュール・パージ，6-13
　スケジュール・プッシュ，6-13
　伝播済みトランザクションのパージ，6-24
　プッシュ，2-30
　問題の診断方法，A-6

て

定期的なパージ
　スケジューリング，6-25
定期的なプッシュ
　スケジューリング，6-22
データ型
　レプリケート表で許可されているデータ型，6-2
データ整合性
　シリアル伝播，2-51
　パラレル伝播，2-51
データ・ディクショナリ
　レプリケーション，1-17
データ伝播
　依存性の維持，2-50
　同期，2-40
　非同期，2-38
データベース・リンク
　指定の不備，A-7
　スケジュール・リンク，1-15
　接続修飾子，2-22
　マテリアライズド・ビュー・サイト，6-11，6-30
　問題の診断方法，A-2
　レプリケーション，2-19
　レプリケーション・マネージメント・ツール，2-21

デッドロック
解消
同期伝播，2-42
伝播，2-38
開始，2-44
パラレル，2-44
実装，2-45
チューニング，2-46
プロパゲータのセキュリティ・コンテキスト，2-41
モード，2-42

と

同期レプリケーション，1-19，2-40
トランザクションの接続先，2-42
動的サイト
所有権，5-15
ドメイン索引
レプリケーション，2-24
トラブルシューティング
レプリケーションの問題，A-1
トランザクション
伝播
保護メカニズム，2-49
トリガー
更新可能マテリアライズド・ビュー・ログ，3-53
マテリアライズド・ビュー・ログ，3-47
レプリケート，2-26
トレース・ファイル
マテリアライズド・ビュー，A-9

な

長さセマンティクス
レプリケーション，B-2

ね

ネストした表
競合解消，5-10
マテリアライズド・ビュー，3-38
制限事項，3-40
列グループ，5-20
レプリケーション，2-16，3-38
ネットワーク
FAILOVER オプション，6-35

は

パージ
遅延トランザクション・キュー
定期的なスケジューリング，6-25
連続的なスケジューリング，6-26
廃棄による競合解消方法，5-26，5-34
配置テンプレート，1-13，4-1
DDL 文，4-14
WHERE 句，4-8
アーキテクチャ，4-13
インスタンス化，1-13，4-9
オプション，4-18
オフライン，4-10，4-17
オンライン，4-9，4-16
使用例，4-11
トラブルシューティング，A-8
プロセス，4-15
インスタンス化の結果，4-19
オブジェクト
定義，4-6，4-14
概念，4-3
行のサブセット化，4-22
設計，4-20
定義の格納，4-14
データ・セット，4-26
テンプレート一般情報，4-4
パッケージ化，4-9
オプション，4-18
オフライン・インスタンシエーション用，4-16
オンライン・インスタンシエーション用，4-15
プロシージャ，4-12
プロセス，4-15
パラメータ，4-7
セキュリティ，4-8
マテリアライズド・ビュー・グループ，4-19
マテリアライズド・ビュー・サイト，4-9
マテリアライズド・ビュー・サイトの準備，6-28
マテリアライズド・ビューの追加，4-14
マテリアライズド・ビュー・ログ，4-13
ユーザー定義型，4-7
ユーザー認証，4-9
要素，4-4
リフレッシュ・グループ，4-19
列のサブセット化，4-21
ローカル制御，4-28
配置テンプレートのパッケージ化，4-9

バックアップ
　レプリケーション用，6-35
パッケージ
　レプリケーション，2-25
パフォーマンス
　レプリケーション，2-44
パラメータ
　配置テンプレート，4-7
パラレル伝播，2-44
　依存性
　　追跡，2-50
　行レベル SCN，2-51
　計画，6-27
　実装，2-45
　チューニング，2-46
　レプリケーション環境，6-27
　レプリケーション環境用の構成，2-44
バルク更新，A-4

ひ

日付式，3-62，4-19
非同期レプリケーション，1-6，2-38
ビュー，2-27
　マテリアライズド・ビュー・サイト，3-52
　レプリケーション，2-27
表
　オンラインでの再定義
　　レプリケーション，3-49
　事前作成
　　長さセマンティクス，B-4
　長さセマンティクス，B-3
　レプリケーション・サポートの生成に関する問題，A-5
　レプリケート，2-23，6-2
　　DML の非互換性，A-4
　　依存性の追跡，6-4
　　許可されているデータ型，6-2
表のオンライン再定義，3-49
表の再定義
　オンライン
　　レプリケーション，3-49

ふ

ファンクション
　レプリケート，2-26
フィルタ列，6-16
フェイルオーバー・サイト
　FAILOVER オプションを使用した実装，6-35
複合マテリアライズド・ビュー，3-10，3-12
　PCTFREE の値，3-59
　PCTUSED の値，3-59
副問合せ
　マテリアライズド・ビュー，3-19
　　AND 演算子，3-24
　　EXISTS 演算子，3-24
　　OR 演算子，3-24
　　1 対多，3-20
　　結合，3-24
　　主キー，3-24
　　制限事項，3-24
　　多対 1，3-19
　　多対多，3-22
　　列のロギング，3-24
プッシュ
　遅延トランザクション・キュー
　　定期的なスケジューリング，6-22
　　連続的なスケジューリング，6-23
プライマリ・サイト
　所有権，5-15
プロシージャ
　レプリケート，2-26
プロシージャ・レプリケーション，1-19
　競合の検出，1-20
　制限事項，2-25
　ラッパー，1-20
プロパゲータ
　レプリケーション，2-19
分散スキーマ管理，1-17
分散トランザクション
　問題，A-6

へ

平均による競合解消方法，5-25
別名
　列
　　更新可能マテリアライズド・ビュー，3-6

ま

マスター・グループ，1-5，2-35
マスター・サイト，1-5
　DDL による変更が伝播されない場合，A-3
　DML による変更が伝播されない場合，A-4
　新しいマスター・サイトでレプリケート・オブジェクトが作成されない場合，A-3
　スケジュール・パージ，6-13
　　ガイドライン，6-24
　スケジュール・リンク
　　ガイドライン，6-21
　制約
　　違反，A-4
　追加
　　自己参照型制約，2-24
　　循環依存性，2-24
　内部トリガー，3-47
　長さセマンティクス，B-3
　　事前作成された表，B-4
　バルク更新，A-4
　マテリアライズド・ビュー，3-46
　マテリアライズド・ビュー・サイトとの比較，6-8
　マテリアライズド・ビューの登録，3-43
　問題の診断方法，A-2
　ユーザー，2-18
　利点，6-9
　レプリケーション，2-18
　ロール，2-18
マスター定義サイト，1-5，2-4
マスター表
　オンラインでの再定義，3-49
　再編成，3-48
　マテリアライズド・ビュー，3-46
　マテリアライズド・ビュー・ログ，3-47
マスター・マテリアライズド・ビュー・サイト，3-46
マスター・マテリアライズド・ビュー，1-8，3-27，3-46
　マテリアライズド・ビュー・ログ，3-47
マテリアライズド・ビュー，1-2，1-8，3-1
　BUILD DEFERRED
　　トラブルシューティング，A-12
　LOB，6-3
　REF，3-40
　　WITH ROWID 句，3-42
　　有効範囲付，3-40
　　有効範囲なし，3-42
　　ロギング，3-42
ROWID，3-9
VARRAY，3-38
　制限事項，3-40
アーキテクチャ，3-44
インポート，3-50
オブジェクト表，3-36
オブジェクト・マテリアライズド・ビュー，3-36
　OID の保持，3-37
概念，3-2
書込み可能，3-7
行のサブセット化，1-11，3-18
権限，3-14，6-13
更新可能，1-9，3-5
　長さセマンティクス，B-5
　マテリアライズド・ビュー・グループ，3-54
　列別名，3-6
更新可能マテリアライズド・ビュー・ログ，3-52
　トリガー，3-53
コレクション列，3-38
　制限事項，3-40
再編成，3-48
索引，3-52
作成，6-18
作成者，3-14
実表，3-50
主キー，3-7，3-24
準備，6-10
使用目的，3-3
所有者，3-14
スキーマの作成，6-11
性能，3-25
タイプ，3-7
大量配布，3-4
多重化，1-8，3-26，3-32
　競合解消，5-6
　高速リフレッシュ，3-60
　制限事項，3-31
　マスター・マテリアライズド・ビュー，3-27
　レベル，3-26
単純，3-12
単純副問合せ
　AND 演算子，3-24
データのサブセット化，3-4，3-18
登録，3-43
トラブルシューティング，A-7

トレース・ファイル，A-9
長さセマンティクス，B-5
　ビルトインのコンテナ表，B-5
ネストした表，3-38
　制限事項，3-40
ネットワーク負荷，3-3
配置テンプレート，1-13，4-1
　ユーザー定義型，4-7
ビュー，3-52
複合，3-10，3-12
　PCTFREE の値，3-59
　PCTUSED の値，3-59
副問合せ，3-19
　EXISTS 演算子，3-24
　OR 演算子，3-24
　1 対多，3-20
　結合，3-24
　主キー・マテリアライズド・ビュー，3-24
　制限事項，3-24
　多対 1，3-19
　多対多，3-22
　列のロギング，3-24
マスター・サイト，3-46
マスター表，3-46
マスター・マテリアライズド・ビュー・サイト，3-46
マスター・マテリアライズド・ビュー，3-46
マテリアライズド・ビュー・グループ，3-53
マテリアライズド・ビュー・ログ，1-12，3-47
モバイル・コンピューティング，3-4
ユーザー定義データ型，3-32
　ON COMMIT REFRESH 句，3-32
読取り専用，1-8，3-5
　登録，3-44
　登録解除，3-44
リフレッシャ，3-14
リフレッシュ，1-12，3-58
　開始，3-62
　間隔，3-62
　完全，3-59
　強制，3-61
　高速，3-25，3-59
　再試行，A-9
　最終リフレッシュ時刻の問合せ，3-43
　障害，A-9
　トラブルシューティング，A-8，A-10
　必要に応じた，3-63
　リフレッシュ・グループ，1-12，3-56
　　サイズ，3-57
　列オブジェクト
　　列のサブセット化，3-35
　列のサブセット化，1-11
　　列オブジェクト，3-35
　ローカル制御，4-28
マテリアライズド・ビュー・グループ，1-5，3-32，3-53
　グループ所有者，3-55
　更新可能マテリアライズド・ビュー，3-54
　所有権，3-32
　配置テンプレート，4-19
マテリアライズド・ビュー・サイト，1-5
　スキーマ，6-30
　スケジュール・パージ
　　ガイドライン，6-24
　スケジュール・リンク
　　ガイドライン，6-21
　設定，6-29
　遅延トランザクション・キュー
　　スケジュール・プッシュ，6-13
　追加
　　問題の回避，6-20
　データベースのバージョン，6-29
　データベース・リンク，6-11，6-30
　長さセマンティクス，B-5
　　ビルトインのコンテナ表，B-5
　ネットワーク接続性，6-29
　配置テンプレート，4-9
　配置テンプレートの準備，6-28
　マスター・サイトとの比較，6-8
　利点，6-9
　ローカルでの作成，4-28
　ロールバック・セグメント，6-31
マテリアライズド・ビュー・ログ，1-12，3-47
　REF，3-42
　ROWID，3-48
　インポート，3-50
　オブジェクト ID，3-48
　オブジェクト表，3-49
　基礎となる表，3-47
　組合せ，3-48
　結合列，6-16
　作成，6-16
　作成に必要な権限，6-16
　主キー，3-48

トラブルシューティング，A-10
トリガー，3-47
配置テンプレート，4-13
フィルタ列，6-16
列の追加，6-17
列のロギング，3-24，6-16
ON PREBUILT TABLE 句，3-24
1 対多副問合せ，3-24
多対 1 副問合せ，3-24
多対多副問合せ，3-24
マルチマスター・レプリケーション，1-6，2-1
アーキテクチャ，2-18
概念，2-2
使用目的，2-4
同期，2-7，2-9
トランザクション伝播保護，2-49
非接続マテリアライズド・ビュー，2-6
非同期，2-7
フェイルオーバー，2-4
ロード・バランシング，2-5

も

最も古いタイムスタンプによる競合解消方法，5-27

ゆ

ユーザー定義データ型
マテリアライズド・ビュー，3-32
ON COMMIT REFRESH 句，3-32
REF，3-40
オブジェクト表，3-36
型の一致，3-33
コレクション，3-38
列オブジェクト，3-32，3-35
レプリケーション，2-12，3-32
REF，2-17，3-40
オブジェクト表，2-15
型の一致，2-12，3-33
コレクション，2-16，3-38
列オブジェクト，2-12，3-32
優先グループによる競合解消方法，5-29

よ

読取り専用マテリアライズド・ビュー，1-8，3-5
登録
手動，3-44
登録解除，3-44

ら

ラッパー
プロシージャ・レプリケーション，1-20

り

リカバリ
レプリケーション用，6-35
リフレッシュ
DBMS_REFRESH パッケージ
CHANGE プロシージャ，4-20
開始，3-62
間隔，3-62
完全，3-59
強制，3-61
グループ，3-62
高速，3-59
使用可能かどうかの判断，3-25
再試行，A-9
自動，3-62
手動，3-63
障害，A-9
スケジューリング，3-62
トラブルシューティング，A-10
トラブルシューティング
ORA-00942 エラー，A-11
ORA-12004 エラー，A-11
必要に応じた，3-63
マテリアライズド・ビュー，1-12，3-58
マテリアライズド・ビューの切捨て
トラブルシューティング，A-11
ロールバック・セグメント
トラブルシューティング，A-11
リフレッシュ・グループ，1-5，1-12，3-56
サイズに関する考慮事項，3-57
トラブルシューティング，A-8
配置テンプレート，4-19

れ

列
　キャラクタ・セマンティクス，B-2
　列グループ，2-37，5-19
　　オブジェクト表，5-20
　　シャドウ，5-20
　　ネストした表，5-20
　　複数の列グループでのデータ整合性の保証，5-19
　　列オブジェクト，5-20
列オブジェクト
　競合解消
　　古い値の送信，5-39
　　古い値の比較，5-39
　マテリアライズド・ビュー，3-32
　　列のサブセット化，3-35
　列グループ，5-20
　レプリケーション，2-12，3-32
列グループ，5-19
　オブジェクト表，5-20
　シャドウ，5-20
　ネストした表，5-20
　列オブジェクト，5-20
列のサブセット化，1-11
　更新可能マテリアライズド・ビュー
　　競合解消，5-9
　配置テンプレート，4-21
　マテリアライズド・ビュー
　　列オブジェクト，3-35
レプリケーション
　DDL 文，1-17
　DELAY_SECONDS パラメータ，2-47
　LOB，6-3
　Oracle Real Application Clusters との比較，6-33
　Unicode，B-7
　アプリケーションでの使用，1-3
　依存性
　　最小化，2-52
　依存性の追跡，2-51
　インポートされたデータのチェック，6-36
　インポートのチェック，6-36
　エラー・キュー，2-29
　オブジェクト，2-23
　概要，1-2
　可用性，6-32
　環境の作成，7-24
　環境の作成のためのフローチャート，7-24
　管理，1-15，2-30
　管理要求，2-31
　　状態，2-34
　管理要求キュー，2-34
　キャラクタ・セット，B-1
　競合
　　検出，2-53
　　プロシージャ・レプリケーション，1-20
　競合解消，1-18，2-10，2-54
　行のサブセット化，1-11
　グループ，1-4，2-35
　計画，6-1
　再開，2-31
　最小限の通信，2-47
　サイト，1-5
　　選択，6-8
　索引タイプ，2-27
　サポートされていないデータ型
　　BFILE，6-2
　　LONG，6-2
　受信者，2-19
　使用方法，1-2
　初期化パラメータ，6-4
　ジョブ，2-32
　ジョブ・キュー，2-30
　ジョブ・キュー・プロセス，6-13
　シングル・マスター，2-3
　スケジュール・リンク，1-15
　静止，1-7，2-30
　制約
　　トラブルシューティング，A-6
　接続修飾子，2-22
　耐障害性，6-32
　大量配布，4-2
　多重化マテリアライズド・ビュー，1-8
　遅延トランザクション，1-6，2-29
　遅延トランザクション・キュー，2-29
　　問題の診断方法，A-6
　中断，2-31
　定義，1-2
　データベース・リンク，2-19
　　CONNECT TO 句，2-20
　　USING 句，2-20
　　レプリケーション・マネージメント・ツール，2-21
　データ要件，5-2
　伝播，2-38

同期，1-19，2-40
トラブルシューティング，A-1
トランザクション
　依存性の順序，2-50
トランザクション伝播保護，2-49
トリガー
　トラブルシューティング，A-5
内部トリガー，2-28
内部プロシージャ，2-29
配置テンプレート，1-13，4-1
　ユーザー定義型，4-7
ハイブリッド構成，1-14
バックアップおよびリカバリ，6-35
パフォーマンス，2-44
非同期伝播，1-6，2-38
表，6-2
　DML の非互換性，A-4
　依存性の追跡，6-4
　許可されているデータ型，6-2
フィルタ列，6-16
フェイルオーバー，6-35
プロシージャ
　トラブルシューティング，A-5
プロシージャ・レプリケーション，1-19
プロパゲータ，2-19
分散スキーマ管理，1-17
マスター，1-8
マスター・グループ，1-5，2-35
マスター・サイト，1-5，2-18
　利点，6-9
マスター定義サイト，1-5
マスター・マテリアライズド・ビュー，1-8
マテリアライズド・ビュー，1-8，3-1
マテリアライズド・ビュー・グループ，1-5
マテリアライズド・ビュー・サイト
　利点，6-9
マテリアライズド・ビュー・ログ，1-12
マルチマスター，1-6，2-1
モード，2-30
問題
　トラブルシューティング，A-1
ユーザー定義データ型，2-12，3-32
リアルタイム・レプリケーション，「同期レプリケーション」を参照
リフレッシュ，1-12
リフレッシュ・グループ，1-5，1-12
列グループ，2-37
　列のサブセット化，1-11
　レプリケーション管理者（REPADMIN ユーザー），2-18
　レプリケーション・マネージメント API，1-17
　割当て表，4-22
レプリケーション・オブジェクト，1-4，2-27
　再作成，A-5
　索引，2-24
　　外部キーに対する，6-2
　順序，2-28
　トリガー，2-26
　パッケージ，2-25
　表，2-23，6-2
　　DML の非互換性，A-4
　　依存性の追跡，6-4
　　外部キー，6-2
　　許可されているデータ型，6-2
　　サポートを生成できない場合，A-5
　　主キー，6-2
　ファンクション，2-26
　プロシージャ，2-26
　マテリアライズド・ビュー・サイト
　　作成に関する問題，A-7
レプリケーション・カタログ，1-17
　DBA_MVIEW_REFRESH_TIMES，3-43
　DBA_REGISTERED_MVIEWS，3-43
　DBA_REPCATLOG，2-32
　USER_REFRESH，A-9
　USER_REFRESH_CHILDREN，A-9
　不適切なビュー，A-7
レプリケーション・マネージメント API，1-17，2-31
レプリケーション・マネージメント・ツール，「Oracle レプリケーション・マネージメント・ツール」を参照
レプリケーション，
　「マテリアライズド・ビュー」も参照
レベル
　多重化マテリアライズド・ビュー，3-26
連続的なパージ
　スケジューリング，6-26
連続的なプッシュ
　スケジューリング，6-23

ろ

ロールバック・セグメント
　マテリアライズド・ビュー・サイト，6-31

わ

割当て表，4-22