**Chapter 1: Introduction to Advanced Replication (page 19)**

**Overview of Replication**

Replication is the process of copying and maintaining database objects, such as tables, in multiple databases that make up a distributed database system. Changes applied at one site are captured and stored locally before being forwarded and applied at each of the remote locations. Advanced Replication is a fully integrated feature of the Oracle server; it is not a separate server.

Reasons for using replication: Availability, Performance, Disconnected computing, Network load reduction, Mass deployment.

**Replication Objects, Groups, and Sites**

Replication Objects: Tables, Indexes, Views and Object Views, Packages and Package Bodies, Procedures and Functions, User-Defined Types and Type Bodies, Triggers, Synonyms, Indextypes, User-Defined Operators.

A replication group is a collection of replication objects that are logically related. A replication group can exist at multiple replication sites.

A replication group can exist at multiple replication sites. Replication environments support two basic types of sites: master sites and materialized view sites.

**Types of Replication Environments**

Multimaster Replication

Materialized View Replication

Multimaster and Materialized View Hybrid Configurations.

**Materialized View Replication**

When it is important for materialized views to be transactionally consistent with each other, you can organize them into refresh groups .

A materialized view log is a table at the materialized view's master site or master materialized view site that records all of the DML changes to the master table or master materialized view.

Deployment templates simplify the task of deploying and maintaining many remote materialized view sites. Using deployment templates, you can define a collection of materialized view definitions at a master site, and you can use parameters in the definitions so that the materialized views can be customized for individual users or types of users.

**Chapter 2: Master Replication Concepts and Architecture (page 39)**

**Master Replication Concepts**

Oracle has two types of master replication: single master replication and multimaster replication. Multimaster replication includes multiple master sites, where each master site operates as an equal peer. In single master replication, a single master site supporting materialized view replication provides the mechanisms to support potentially hundreds or thousands of materialized view sites.

**Multimaster Replication Process**

There are two types of multimaster replication: asynchronous and synchronous.

Asynchronous replication independently propagates any DML or replicated procedure execution to all of the other master sites participating in the multimaster replication environment. Asynchronous replication requires less networking and hardware resources than does synchronous replication, resulting in better availability and performance.

Synchronous replication propagates any changes made at a local site to other synchronously linked masters in a replication environment during the same transaction as the initial change. Synchronous replication also ensures that no data conflicts are introduced into the replication environment.

**Conflict Resolution Concepts**

Any data conflicts introduced by a materialized view site are detected and resolved at the target master site or master materialized view site of the materialized view.

After a data conflict is detected, the following actions occur:  
1. The conflict resolution methods try to resolve the data conflict.  
2. If the conflict is not resolved, then the data conflict is logged in the error queue at the destination site.

**Master Replication Architecture**

Master Site Roles/Users: Replication Administrator, Propagator, Receiver.

There are three modes of operation for a replication environment: Normal, Quiescing, Quiesced. There are only two mechanisms to control these modes: Suspend, Resume.

Administrative Request Queue: each request has a status that displays the state of the request. Here are the possible states: READY, AWAIT\_CALLBACK, ERROR, DO\_CALLBACK.

Organizational Mechanisms:

* Master Group. Corresponding master groups at different sites must contain the same set of replication objects.
* Column Groups: provide the organizational mechanism to group all columns that are involved in a conflict resolution routine.

Asynchronous data replication occurs when an application updates a local replica of a table, stores replication information in a local queue, and then forwards the replication information to other replication sites at a later time. Consequently, asynchronous data replication is also called store-and-forward data replication.

Synchronous data propagation occurs when an application updates a local replica of a table, and within the same transaction also updates at least one other replica of the same table. Consequently, synchronous data replication is also called real-time data replication .

Performance Mechanisms: Parallel Propagation, Minimum Communication, Delay Seconds.

Replication Protection Mechanisms (Data Propagation Dependency Maintenance):

* Parallel Propagation Dependency Tracking
* Use of Row-Level Dependency Tracking to Improve Parallelism
* Minimize Transaction Dependencies to Improve Parallelism

Conflict Resolution Mechanisms: Resolution of Data Conflicts:

* Latest and Earliest Timestamp
* Overwrite and Discard
* Maximum and Minimum
* Additive and Average
* Timestamp
* Priority Group
* Site Priority

**Chapter 3: Materialized View Concepts and Architecture (page 97)**

**Materialized View Concepts**

A materialized view is a replica of a target master from a single point in time. The master can be either a master table at a master site or a master materialized view at a materialized view site.

You can use materialized views to achieve one or more of the following goals:

* Ease Network Loads
* Create a Mass Deployment Environment
* Enable Data Subsetting
* Enable Disconnected Computing

**Read-Only, Updatable, and Writeable Materialized Views**

Read-Only, Updatable, and Writeable Materialized Views: A materialized view can be either read-only, updatable, or writeable. Users cannot perform data manipulation language (DML) statements on read-only materialized views, but they can perform DML on updatable and writeable materialized views.

Available Materialized Views:

* Primary Key Materialized Views
* Object Materialized Views
* ROWID Materialized Views
* Complex Materialized Views

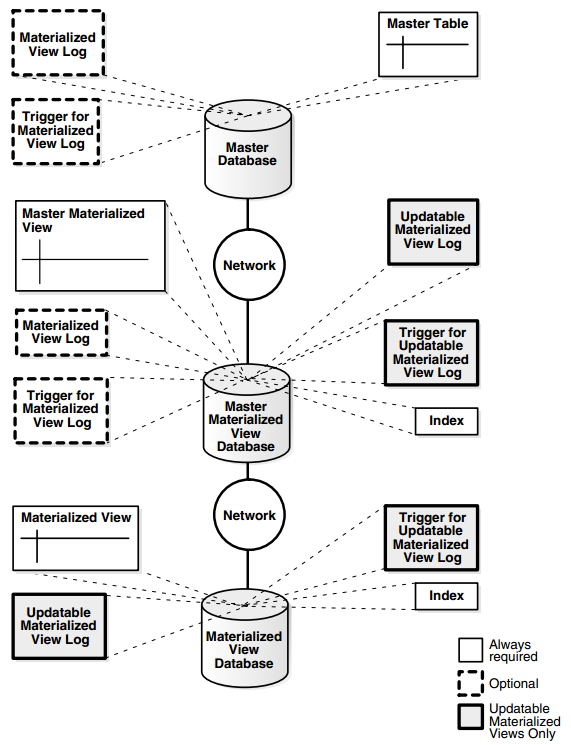
Three distinct types of users perform operations on materialized views:

* Creator: the user who creates the materialized view
* Refresher: the user who refreshes the materialized view
* Owner: the user who owns the materialized view. The materialized view resides in this user's schema.

Some reasons to use data subsetting are to: Reduce Network Traffic, Secure Sensitive Data, Reduce Resource Requirements, Improve Refresh Times.

The ability to create materialized views that are based on other materialized views enables you to create multitier materialized views.

**Materialized View Architecture**



A materialized view group in a replication system maintains a partial or complete copy of the objects at the target replication group at its master site or master materialized view site.

Refresh Groups: to preserve referential integrity and transactional (read) consistency among multiple materialized views, Oracle has the ability to refresh individual materialized views as part of a refresh group. A refresh group can contain up to 400 materialized views.

|  |  |
| --- | --- |
| **Advantages of Large Refresh Groups** | **Advantages of Small Refresh Groups** |
| * Refreshes faster than an equal number of materialized views in multiple refresh groups * Refreshes with single replication management API call | * Materialized views locked for shorter periods of time * Rollback of refresh changes due to loss of connectivity is less likely |

A materialized view refresh is an efficient batch operation that makes a materialized view reflect a more current state of its master table or master materialized view.

Refresh Types: Complete Refresh, Fast Refresh, Force Refresh.

Initiating a Refresh: Scheduled Refresh, On-Demand Refresh

**Chapter 4: Deployment Templates Concepts and Architecture (page 167)**

**Mass Deployment Challenge**

The need to have accurate information at any time and at any place continues to grow rapidly. At the same time, information is becoming decentralized and users are often disconnected from the network, requiring the information to be distributed to the active points-of-usage.

**Deployment Templates and the Mass Deployment Goal**

Oracle's deployment templates enable you accomplish the following objectives:

* Define the materialized view environment once: You define the structure of a materialized view environment once using a deployment template so that each user (site) receives the database infrastructure to support the front-end application.
* Customize materialized view sites individually: You use deployment template parameters to customize each materialized view environment so that each user receives the particular data subset needed.

**Oracle Deployment Templates Concepts**

Oracle offers deployment templates to allow the database administrator to package a materialized view environment for easy, custom, and secure deployment. Packaging a deployment template is the process of defining the materialized view environment that will be created by the deployment template. Packaging a deployment template prepares it for instantiation at the remote materialized view site. Instantiation creates the materialized view site objects and populates the materialized views with data.

Deployment template features include the following:

* Centralized control
* Ability to repeatedly deploy a materialized view environment
* Template parameters that allow data subsetting or customization at remote site
* Authorized user lists to control template instantiation and data access

**Deployment Template Elements**

General Template Information

Oracle deployment templates center around the general template information, which consists of the template name, target refresh group, and private/public status.

A deployment template is defined at a single master site. While you cannot have two deployment templates at the master site with the same name, you can copy a deployment template to another site using the same deployment template name.

Template Object Definitions

You cannot use deployment templates to instantiate the following types of objects: User-defined types, User-defined type bodies, User-defined operators, Indextypes.

Nor can you use deployment templates to instantiate any objects based on these types of objects.

Template Parameters

During template instantiation, the individual user values for these parameters are substituted.

Oracle enables you to specify default values and user-specific parameter values for a template. You can enter the parameter values during the creation of the deployment template or after the template is created, but you must enter the parameter values before the template is instantiated. Users cannot enter values for parameters during instantiation.

User Authorization

Deployment templates can be either public or private. You set this when you create the template. If a template is public, then any user with access to the master site can instantiate the template.

If a template has been created for private use, then only authorized users can instantiate the target template.

**Deployment Template Packaging and Instantiation**

Online Instantiation

Online instantiation allows a materialized view site to instantiate a deployment template while connected to the target master site.

Packaging a deployment template for online instantiation means generating a script file that, when run at the materialized view site, creates the materialized view objects and connects to the master site to populate the materialized views with data.

One of the benefits of online instantiation is that the data subset is current as of the instantiation process.

Offline Instantiation

To decrease server loads during peak usage periods and reduce remote connection times, you may choose offline instantiation of the template for your environment.

Packaging a template for offline instantiation means generating a script or a binary file that contains the DDL and data manipulation language (DML) to build the materialized view environment defined in the deployment template and populate the environment with data.

Offline instantiation is an ideal solution for mass deployment situations where many laptops and other disconnected computers are instantiating the target template.

**Deployment Template Architecture**

Template Definitions Stored in System Tables

Template object definitions are created using the same DDL that is used to create the objects locally at the materialized view site.

Packaging and Instantiation Process

Two possible methods can be used to define template parameter values: default parameter values and user parameter values.

In a mass deployment environment, most materialized view environments use the offline instantiation method to create the necessary materialized view environment.

**Deployment Template Design**

The combination of deployment template parameters and subquery subsetting gives the database administrator a powerful tool to administer a widely distributed database environment using subqueries and row-subsetted data.

**Local Control of Materialized View Creation**

One of the major benefits of deployment templates is that control is maintained centrally by the DBA building the deployment template.

Local control may be required if the materialized view site:

* Has an experienced DBA
* Is considered a trusted site
* Is a materialized view instead of a master site because of row subsetting requirements

**Chapter 5: Conflict Resolution Concepts and Architecture (page 197)**

**Conflict Resolution Concepts**

Using several techniques, most system designs can avoid conflicts in all or a large percentage of the data that is replicated. However, many applications require that some percentage of data be updatable at multiple sites at any time. If this is the case, then you must address the possibility of replication conflicts.

**Understanding Your Data and Application Requirements**

Conflict resolution is often not possible in reservation systems where multiple bookings for the same item are not allowed. Conflict resolution is often possible in customer management systems.

**Types of Replication Conflicts**

An update conflict occurs when the replication of an update to a row conflicts with another update to the same row. Update conflicts can happen when two transactions originating from different sites update the same row at nearly the same time.

A uniqueness conflict occurs when the replication of a row attempts to violate entity integrity, such as a PRIMARY KEY or UNIQUE constraint. For example, consider what happens when two transactions originate from two different sites, each inserting a row into a respective table replica with the same primary key value. In this case, replication of the transactions causes a uniqueness conflict.

A delete conflict occurs when two transactions originate from different sites, with one transaction deleting a row and another transaction updating or deleting the same row, because in this case the row does not exist to be either updated or deleted.

**How Oracle Detects Different Types of Conflicts**

The receiving master site or master materialized view site in a replication system detects update, uniqueness, and delete conflicts as follows:

* The receiving site detects an update conflict if there is any difference between the old values of the replicated row (the values before the modification) and the current values of the same row at the receiving site.
* The receiving site detects a uniqueness conflict if a uniqueness constraint violation occurs during an INSERT or UPDATE of a replicated row.
* The receiving site detects a delete conflict if it cannot find a row for an UPDATE or DELETE statement because the primary key of the row does not exist.

**Conflict Resolution**

Multitier Materialized Views and Conflict Resolution

When you have a master table and an updatable materialized view based on that master table, a refresh of the materialized view pushes its changes to the master site, where the master site handles any conflicts resulting from the push with its configured conflict resolution methods.

Keep this in mind if you are using multitier materialized views. Because the conflict resolution methods are pulled down from the master site to a master materialized view, the same rules apply to master materialized view sites and updatable materialized views based on them.

Nested Tables and Conflict Resolution

Recommendations for Avoiding Problematic Updates

* + Use a foreign key constraint, initially deferred, on the nested table. This constraint prevents dangling rows in the storage table.
  + Make sure all inserts on the parent table insert an empty nested table. Do not use a null nested table value. This practice helps to create a reusable NESTED\_TABLE\_ID.
  + Make sure all inserts, deletes, and updates are performed directly on the nested table rather than through DML on the parent table.
  + Consider using a trigger on the parent table that prevents inserts and updates that include manipulation of the nested table column.

**Techniques for Avoiding Conflicts**

Use Column Groups

Column groups can help you avoid conflicts even if you do not apply any conflict resolution methods to the column groups. When your replicated table contains multiple column groups, each group is viewed independently when analyzing updates for conflicts.

Use Primary Site and Dynamic Site Ownership Data Models

One way that you can avoid the possibility of replication conflicts is to limit the number of sites in the system with simultaneous update access to the replicated data. Two replicated data ownership models support this approach: primary site ownership and dynamic site ownership.

Primary ownership is the replicated data model that the read-only replication environments support. Primary ownership prevents all replication conflicts, because only a single server permits update access to a set of replicated data.

The dynamic ownership replicated data model is less restrictive than primary site ownership. With dynamic ownership, capability to update a data replica moves from site to site, still ensuring that only one site provides update access to specific data at any given point in time.

Avoiding Specific Types of Conflicts

Avoiding Uniqueness Conflicts: you can create sequences at each site so that each sequence at each site generates a mutually exclusive set of sequence numbers. Alternatively, you can append a unique site identifier as part of a composite primary key. Finally, you can select a globally unique value using the SYS\_GUID function.

Avoiding Delete Conflicts: applications should mark rows for deletion and then configure the system to periodically purge logically deleted rows using procedural replication.

Avoiding Update Conflicts: in a shared ownership data model, update conflicts cannot be avoided in all cases.

Avoiding Ordering Conflicts: Whenever possible, avoid or automatically resolve ordering conflicts.

**Conflict Resolution Architecture**

Support Mechanisms

Column groups have the following characteristics:

* A column can belong only to one column group.
* A column group can consist of one or more columns of a table.
* Conflict resolution is applicable only to columns in a column group.

By default, every replicated table has a shadow column group. The shadow column group of a table contains all columns that are not within a specific column group. You cannot assign conflict resolution methods to a table's shadow group.

An Oracle object based on a user-defined type that occupies a single column in a table is a column object. A column object cannot span column groups.

An object table is a special kind of table in which each row represents an object based on a user-defined type.

A nested table's storage table is treated as an independent table in conflict resolution.

If a conflict resolution method fails to resolve a data conflict, or if you have not defined any conflict resolution methods, then the error queue contains information about the data conflict.

Common Update Conflict Resolution Methods

|  |  |
| --- | --- |
| **Resolution Methods** | **Convergence with Multiple Master Sites** |
| Latest timestamp | YES (with backup method) |
| Overwrite | NO |

The latest timestamp method resolves a conflict based on the most recent update, as identified by the timestamp of when the update occurred.

The overwrite method replaces the current value at the destination site with the new value from the originating site, and therefore can never guarantee convergence with more than one master site. This method is designed to be used by a single master site and multiple materialized view sites.

The overwrite method is also useful if:

* Your primary concern is data convergence.
* You have a single master site.
* No particular business rule exists for selecting one update over the other.
* You have multiple master sites and you supply a notification facility to notify the person who ensures that data is correctly applied, instead of logging the conflict in the DEFERROR data dictionary view and leaving the resolution to your local database administrator.

**Additional Update Conflicts Resolution Methods**

|  |  |
| --- | --- |
| **Resolution Methods** | **Convergence with Multiple Master Sites** |
| Additive | YES |
| Average | NO |
| Discard | NO |
| Earliest timestamp | NO |
| Maximum | YES (column values must always increase) |
| Minimum | YES (column values must always decrease) |
| Priority group | YES (with ordered update values) |
| Site priority | NO |

The additive method works with column groups consisting of a single numeric column only. If a conflict arises, instead of choosing one value over another, then the difference of the two values is added to the current value. This method might be useful in a financial environment where deposits and withdrawals happen so frequently that conflicts may arise.

The average method works with column groups consisting of a single numeric column only. Instead of adding the difference to the current value, the average method resolves the conflict by computing the average of the current and the new value. The average method might be useful for scientific applications that would rather average two values than choose one value over another (for example, to compute the average temperature or weight).

The discard method ignores the values from the originating site and therefore can never guarantee convergence with more than one master site. The discard method ignores the new value from the originating site and retains the value at the destination site.

The discard methods is also useful if:

* Your primary concern is data convergence.
* You have a single master site.
* There is no particular business rule for selecting one update over the other.
* You have multiple master sites and you supply a notification facility to notify the person who ensures that data is correctly applied, instead of logging the conflict in the DEFERROR view and leaving the resolution to your local database administrator.

The earliest timestamp methods resolves a conflict based on the earliest (oldest) update, as identified by the timestamp of when the update occurred.

When Advanced Replication detects a conflict with a column group and calls the maximum value conflict resolution method, it compares the new value from the originating site with the current value from the destination site for a designated column in the column group.

When Advanced Replication detects a conflict with a column group and calls the minimum value conflict resolution method, it compares the new value from the originating site with the current value from the destination site for a designated column in the column group.

Priority groups allow you to assign a priority level to each possible value of a particular column. If Oracle detects a conflict, then Oracle updates the table whose "priority" column has a lower value using the data from the table with the higher priority value. Therefore, a higher value means a higher priority.

Site priority is a special kind of priority group. With site priority, the priority column you designate is automatically updated with the global database name of the site where the update originated.

**Uniqueness Conflicts Resolution Methods**

Oracle provides three prebuilt methods for resolving uniqueness conflicts:

* Append the global site name of the originating site to the column value from the originating site.
* Append a generated sequence number to the column value from the originating site.
* Discard the row value from the originating site.

The append site name method works by appending the global database name of the site originating the transaction to the replicated column value that is generating a dup\_val\_on\_index exception.

The append sequence methods works by appending a generated sequence number to the column value that is generating a dup\_val\_on\_index exception.

The discard uniqueness conflict resolution method resolves uniqueness conflicts by simply discarding the row from the originating site that caused the error. This method does not guarantees convergence with multiple master sites and should be used with a notification facility. Unlike the append methods, the discard uniqueness method minimizes the propagation of data until data accuracy can be verified.

**Delete Conflict Resolution Methods**

Send and Compare Old Values: To detect and resolve an update conflict for a row, the propagating site must send a certain amount of data about the new and old versions of the row to the receiving site. Depending on your environment, the amount of data that Oracle propagates to support update conflict detection and resolution can be different.