Oracle® Database

2 Day + Data Replication and Integration Guide

11*g* Release 1 (11.1)

B28324-03

August 2008

Covers using Oracle Streams, materialized views, and other distributed database functionality

ORACLE®

Oracle Database 2 Day + Data Replication and Integration Guide, 11*g* Release 1 (11.1)

B28324-03

Primary Author: Randy Urbano

Contributing Author: Maria Pratt

Contributor: Janet Blowney, Steve Fogel, Vira Goorah, Thuvan Hoang, Stella Kister, Sushil Kumar, Patricia McElroy, Colin McGregor, Valarie Moore, Ashish Ray, Subbanarasimha Shastry, Sreejesh Srinivasan, Jim Stamos, Mark Townsend, Byron Wang, Lik Wong, Jingwei Wu, Jun Yuan, Ramkumar Venkatesan

Contents

4 Replicating Data Using Oracle Streams

5 Administering an Oracle Streams Replication Environment

6 Extending an Oracle Streams Replication Environment

7 Replicating Data Using Materialized Views

8 Administering a Materialized View Replication Environment

9 Sending Messages Using Oracle Streams Advanced Queuing

Preface

This preface contains these topics:

Audience

Oracle Database 2 Day + Data Replication and Integration Guide is for anyone who wants to perform data replication and integration tasks that involve Oracle databases. Data replication and integration tasks involve using information at two or more databases in a unified way.

This guide recommends best practices and describes efficient ways of performing data replication and integration tasks. This guide describes using Oracle Enterprise Manager to complete tasks whenever possible.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at `http://www.oracle.com/accessibility/`.

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

TTY Access to Oracle Support Services

Oracle provides dedicated Text Telephone (TTY) access to Oracle Support Services within the United States of America 24 hours a day, 7 days a week. For TTY support, call 800.446.2398. Outside the United States, call +1.407.458.2479.

Related Documents

For more information, see these Oracle resources:

- *Oracle Database 2 Day DBA*
- *Oracle Database Administrator's Guide*
- *Oracle Streams Concepts and Administration*
- *Oracle Streams Replication Administrator's Guide*
- *Oracle Streams Advanced Queuing User's Guide*
- *Oracle Database Heterogeneous Connectivity Administrator's Guide*
- *Oracle Database Advanced Replication*
- *Oracle Database Advanced Replication Management API Reference*
- *Oracle Database PL/SQL Packages and Types Reference*
- *Oracle Database Reference*
- The Enterprise Manager online Help
- *Oracle Enterprise Manager Concepts*

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1

Introduction to Data Replication and Integration

As a database administrator with multiple databases to manage, you are responsible for making information available when and where it is needed.

This chapter contains the following sections:

About This Guide

Oracle Database 2 Day + Data Replication and Integration Guide teaches you how to perform common tasks that are necessary to configure and administer several different types of data replication and integration environments. See "About Data Replication and Integration" on page 1-2 and "About Data Replication and Integration Features" on page 1-3 for information about the types of environments.

This guide helps you decide which data replication and integration environments are best for you. This guide also provides basic, task-oriented instructions for configuring, maintaining, monitoring, and troubleshooting common types of data replication and integration environments.

The primary interfaces used in this guide are Oracle Enterprise Manager and SQL*Plus.

This section contains the following topics:

Before Using This Guide

Before using this guide, you must:

- Become familiar with *Oracle Database 2 Day DBA*
- Install Oracle Database software and configure at least two databases

Because this guide describes data replication and integration at multiple databases, more than one database is required for most of the tasks described in this guide.

What This Guide Is Not

The *Oracle Database 2 Day + Data Replication and Integration Guide* is task-oriented. The objective is to describe common data replication and integration tasks. Where appropriate, it describes the concepts necessary for understanding and completing the current task.

Data replication and integration involves several Oracle Database features. These features include distributed SQL, Oracle Database Gateway, Oracle Streams, and materialized views. This guide does not provide exhaustive information about these features. For complete conceptual information about these features and detailed instructions for using them, see the appropriate Oracle documentation:

- *Oracle Database Administrator's Guide* for information about distributed SQL
- *Oracle Database Heterogeneous Connectivity Administrator's Guide* for information about Oracle Database Gateway
- *Oracle Streams Concepts and Administration* for general information about Oracle Streams
- *Oracle Streams Replication Administrator's Guide* for information about using Oracle Streams for replication
- *Oracle Database Advanced Replication* and *Oracle Database Advanced Replication Management API Reference* for more information about materialized views
- *Oracle Streams Advanced Queuing User's Guide* for information about using Oracle Streams for message queuing

Also, this guide describes using some of the data replication and integration features available in Oracle Enterprise Manager, but this guide does not provide exhaustive information about Enterprise Manager. For information about Enterprise Manager, see:

- The Enterprise Manager online Help
- *Oracle Enterprise Manager Concepts*

About Data Replication and Integration

As organizations expand, it becomes increasingly important for them to be able to share information among multiple databases and applications. Data replication and integration enables you to access information when and where you need it in a distributed environment. Oracle Database provides secure and standard mechanisms that enable communication between databases, applications, and users. These mechanisms include queues, data replication, messaging, and distributed access in both homogeneous and heterogeneous environments.

This guide describes using distributed SQL, replication, and message queuing. You can make efficient use of your computing resources by using these features to complete the following types of tasks:

- Replicate data between databases
- Provide easy access to data in distributed databases
- Exchange data between Oracle databases and non-Oracle databases
- Enable communication between applications
- Exchange information with customers, partners, and suppliers
- Provide event notification and workflow

Oracle Database provides the following types of data replication and integration solutions to address your specific requirements:

- **Consolidation:** All data is moved into a single database and managed from a central location. Oracle Real Application Clusters (Oracle RAC), Grid computing, and Virtual Private Database (VPD) can help you consolidate information into a single database that is highly available, scalable, and secure.
- **Federation:** Data appears to be integrated in a single virtual database, while actually remaining in its current distributed locations. Distributed queries, distributed SQL, and Oracle Database Gateway can help you create a federated database.
- **Sharing:** Multiple copies of same information are maintained in multiple databases and application data stores. Data replication and messaging can help you share information at multiple databases.

See Also:

- *Oracle Database 2 Day + Real Application Clusters Guide*
- *Oracle Database 2 Day + Security Guide*

About Data Replication and Integration Features

It is not always possible for an organization to consolidate all of its data into a single database. The data might be spread over several geographic locations, and some remote locations might not have good connectivity with a primary site. In some cases, the data might be consolidated, but the organization might need a method for best of breed applications to communicate with each other. These are only a few reasons why organizations might need to share information between locations or applications.

Oracle Database provides several ways for organizations to achieve their data replication and integration goals. This topic helps you decide which data replication and integration features are best for your organization.

The following topics describe when to use different data replication and integration features:

- When to Access and Modify Information in Multiple Databases
- When to Replicate Data with Oracle Streams
- When to Replicate Data with Materialized Views
- When to Send Messages Between Databases

You can choose to use one of these features or a combination of them to meet your requirements.

Note: In addition to the data replication and integration features described in this guide, Oracle Warehouse Builder is another option that you can use to integrate information. Oracle Warehouse Builder is a flexible tool that enables you to design and deploy various types of data integration strategies. Projects commonly implemented using Warehouse Builder involve mission critical operational systems, migration scenarios, integration of disparate operational systems, and traditional data warehousing. Oracle Warehouse Builder is comprised of a set of graphical user interfaces to assist you in implementing solutions for integrating data. See *Oracle Warehouse Builder User's Guide* for information about using it.

When to Access and Modify Information in Multiple Databases

Despite their best efforts to consolidate information, many organizations find themselves with multiple, distributed databases. Even if these organizations might prefer to centralize this data, at least in the short term, it might not be possible. These organizations must have a method of accessing these distributed data sources as if they were a single, centralized database. Using distributed SQL, applications and users can access and modify information at multiple Oracle or non-Oracle databases as if it resided in a single Oracle database.

Because information does not need to be moved or copied, using distributed SQL to federate their distributed data sources provides organizations with the fastest, and easiest, path to information integration. If information is later moved, then it is not necessary to rewrite an application. This is especially useful for organizations that are transitioning to a consolidated approach, but need a method for accessing the distributed data now.

For example, by using distributed SQL with the appropriate Oracle Database Gateway, applications can access legacy data immediately, without waiting until it can be imported into an Oracle Database. Distributed SQL is also useful to organizations that want to perform ad hoc queries or updates on infrequently accessed data that is more appropriately located elsewhere.

See Also:

- Chapter 3, "Accessing and Modifying Information in Multiple Databases"

When to Replicate Data with Oracle Streams

If connectivity is not an issue, then organizations might prefer to replicate data in a near-real-time manner. Doing so insures that the data is up to date at all locations as soon as possible. Oracle Streams supports near-real-time data replication in a variety of configurations, depending on an organization's specific requirements. In an Oracle Streams replication environment, databases push changes to each other automatically.

Common uses for Oracle Streams replication include:

- Creating a reporting site to offload processing from a primary online transaction processing (OLTP) site.
- Providing load balancing and improved scalability and availability for a call center or similar application.
- Providing site autonomy between locations to satisfy certain common business requirements.

- Transforming and consolidating data from multiple locations, such as regional offices.
- Replicating data between different platforms and Oracle Database releases, and across a wide area network (WAN).

There are two common types of Oracle Streams replication configurations: n-way and hub-and-spoke. Specifically, a multimaster (or n-way) configuration is frequently used by organizations that must provide scalability and availability of data. Often, these applications use a "follow the sun" model, with replicas located around the globe. For example, an organization might have call centers in the United States, Europe, and Asia, each with a complete copy of the customer data. Customer calls can be routed to the appropriate call center depending on the time of day. Each call center has fast, local access to the data. If a site becomes unavailable for any reason, then transactions can be routed to one of the surviving locations. This type of configuration can also be used to provide load balancing between multiple locations.

Another common configuration is hub-and-spoke. For example, an insurance company might use this configuration to share customer data between its headquarters and local sales offices. A networked version of this configuration can be especially useful in cases of limited connectivity between the end spokes and the hub. Suppose local sales offices have direct connectivity to regional offices, which in turn connect to headquarters, but the local offices have no direct connectivity to headquarters. This type of networked routing can eliminate some of the complexity that results when there are direct connections between all locations. The hub-and-spoke configuration is also useful in data warehousing environments, where detailed data is maintained at each store or spoke, and higher-level data can be shared with the data warehouse or hub.

In both n-way and hub-and-spoke configurations, organizations can configure Oracle Streams replication to allow updates to the replicated data at multiple locations. In such replication environments, data conflicts are possible. Oracle Streams provides conflict resolution methods that can resolve these conflicts automatically.

Because Oracle Streams provides a flexible infrastructure for all information sharing requirements, including messaging and replication, it is easy for an organization to change its configuration as its needs change.

See Also:

- Chapter 4, "Replicating Data Using Oracle Streams"

When to Replicate Data with Materialized Views

It might not always be practical for all users to access data that is stored in a single location. For example, field sales personnel might need access to a price list when they are at a customer site, without immediate access to the corporate databases. They might want to process an order, even if they cannot connect to the primary or master database. These users require a replica of the database or a portion of the database.

Unlike Oracle Streams replication, materialized views do not continuously replicate data at all times. A materialized view is a replica of a table or a subset of a table that can be refreshed to a transactionally consistent point in time. During a refresh, only the final values of the changed rows are pulled down and applied to the materialized view, no matter how many updates were applied to the master table. This reduces the amount of time that the remote site must be connected to the master site.

Materialized views are especially useful for locations with limited connectivity to the master site. Updatable materialized views allow these locations to function autonomously, even when connectivity is unavailable. When updates are allowed at

multiple locations, ownership is typically partitioned in some manner between the locations to prevent conflicting updates. When conflicts are possible, Oracle provides conflict resolution methods that can resolve these conflicts automatically.

In addition to supporting disconnected computing, organizations can also use materialized views to improve performance and scalability by providing local access to data and by off loading processing at the primary location. For example, one or more materialized views might be used to off load reporting activity from an order-entry system.

See Also:

- Chapter 7, "Replicating Data Using Materialized Views"

When to Send Messages Between Databases

As organizations grow, they typically develop a variety of applications to automate processes and manage tasks. Although these applications do not share data directly, they might not operate entirely autonomously. These applications need a way to communicate with one another to coordinate tasks and exchange information.

Using Oracle Streams Advanced Queuing (AQ), applications can securely and reliably communicate with one another in an asynchronous manner. Oracle Streams AQ supports all of the standard features of message queuing systems, including multiconsumer queues, publish and subscribe, content-based routing, Internet propagation, and transformations. So, for example, the shipping department can easily notify the billing department when a product has shipped, and the customer can be billed accordingly.

By combining Oracle Streams AQ with the appropriate messaging gateway, applications can even interoperate with other message queuing systems, such as TIBCO Rendezvous or IBM Websphere MQ. This ability can be especially useful when it is necessary to share information with business partners or customers.

See Also:

- Chapter 9, "Sending Messages Using Oracle Streams Advanced Queuing"

2

Common Data Replication and Integration Tasks

This chapter describes how to complete common tasks that are required in many data replication and integration environments.

This chapter contains the following sections:

Setting the GLOBAL_NAMES Initialization Parameter to TRUE

To access data in multiple locations, you must first ensure that each location can be uniquely identified. Next, you must establish a communication path between these locations.

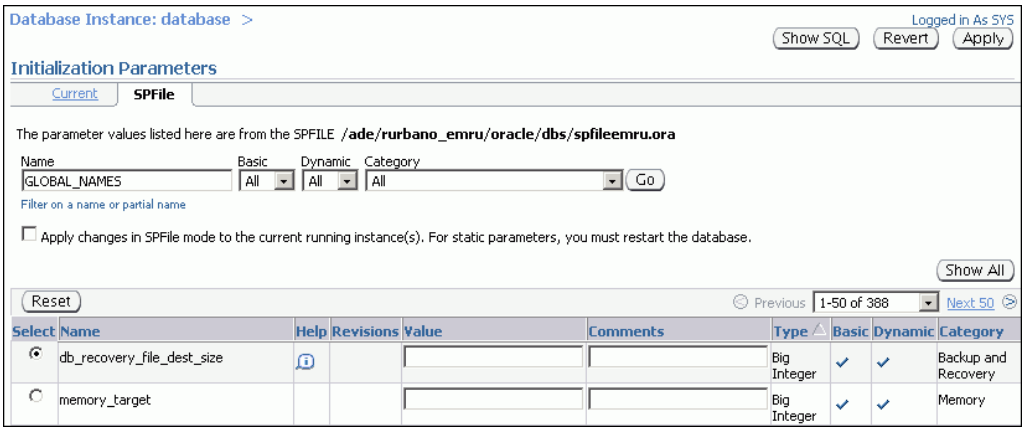
The unique identifier for each database is referred to as its **global database name**. By setting the initialization parameter `GLOBAL_NAMES` to `TRUE`, you guarantee that each database in your distributed database environment can be uniquely identified. A database forms a global database name by prefixing the database network domain, specified by the `DB_DOMAIN` initialization parameter at database creation, with the individual database name, specified by the `DB_NAME` initialization parameter.

The `GLOBAL_NAMES` parameter specifies whether a database link is required to have the same name as the database to which it connects. If you use or plan to use distributed processing, then Oracle recommends that you set this parameter to `TRUE` at each database to ensure the use of consistent naming conventions for databases and links in a networked environment.

To set the GLOBAL_NAMES initialization parameter to TRUE at a database:

1. Log in to Enterprise Manager as an administrative user who can change initialization parameters. For example, you can log in as a user with `SYSDBA` privilege.
2. Go to the Database Home page for the database instance.
3. Click **Server** to open the Server subpage.
4. Click **Initialization Parameters** in the Database Configuration section.
5. If you are using a server parameter file, then click **SPFile**. Otherwise, proceed to the next step.

6. On the Initialization Parameters page, enter `GLOBAL_NAMES` in the search tool.



7. Click **Go**.
8. Set the `GLOBAL_NAMES` initialization parameter to `TRUE`.
9. Click **Apply** to save your changes.

   Ensure that you set the parameter permanently in either the server parameter file or in your initialization parameter file.
10. Complete Steps 1 through 9 for each database in your distributed environment. By default, the `GLOBAL_NAMES` initialization parameter is set to `FALSE`. Therefore, it must be set to `TRUE` explicitly at each database.

If you were directed to this topic from another topic, then go back to the topic now:

- "Preparing to Access and Modify Information in Multiple Oracle Databases" on page 3-3
- "Preparing for Messaging" on page 9-5
- "Preparing for Materialized View Replication" on page 7-3
- "Preparing for Oracle Streams Replication" on page 4-21

See Also:

- "Tutorial: Creating a Database Link" on page 2-8
- *Oracle Database 2 Day DBA*
- *Oracle Database Reference* for more information about the `GLOBAL_NAMES` initialization parameter

Tutorial: Creating an Oracle Streams Administrator

If you plan to use any of the components of Oracle Streams in your environment, then configure an Oracle Streams administrator. Oracle Streams components include:

- Queues
- Queue tables
- Capture processes
- Propagations

- Apply processes
- Rules and rule sets

An Oracle Streams administrator configures and manages these components at each database where they are used. See the following topics in this guide for information about these components:

- Chapter 4, "Replicating Data Using Oracle Streams"
- Chapter 9, "Sending Messages Using Oracle Streams Advanced Queuing"

To configure an Oracle Streams administrator, either create a new user with the appropriate privileges or grant these privileges to an existing user. You should not use the `SYS` or `SYSTEM` user as an Oracle Streams administrator, and the Oracle Streams administrator should not use the `SYSTEM` tablespace as its default tablespace.

To create an Oracle Streams administrator named strmadmin:

1. Tutorial: Creating the Tablespace for the Oracle Streams Administrator
2. Tutorial: Creating the Oracle Streams Administrator

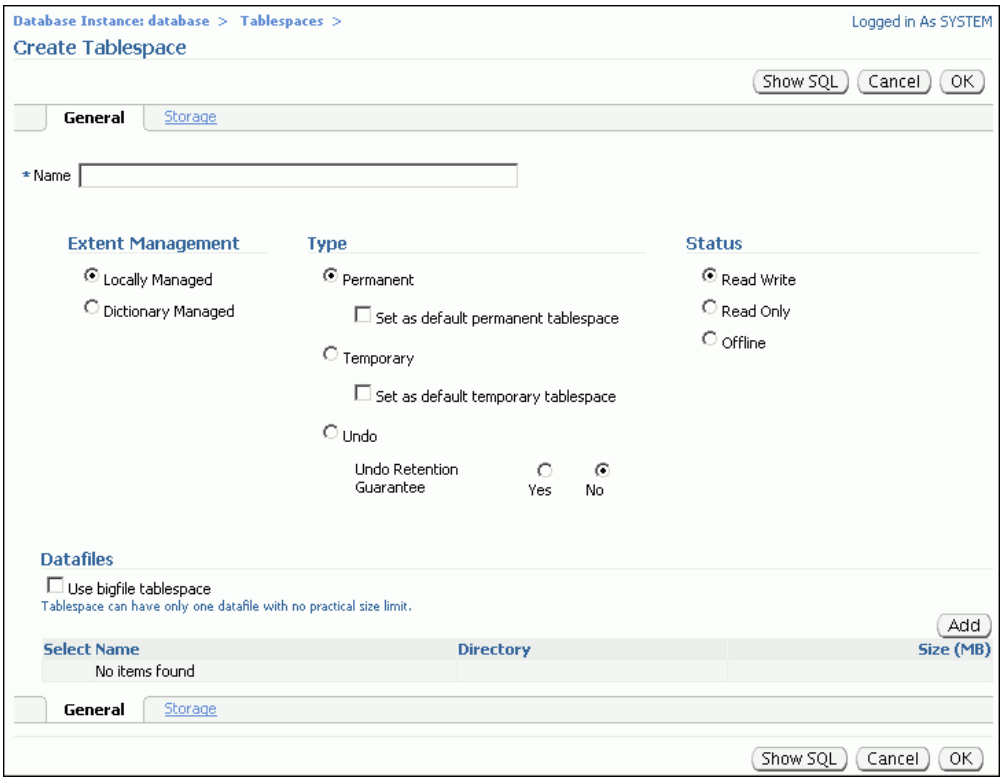
Tutorial: Creating the Tablespace for the Oracle Streams Administrator

The Oracle Streams administrator should use a dedicated tablespace that is not used by any other user. Queue tables and other Oracle Streams components require disk space, and a dedicated tablespace can meet these space requirements efficiently.

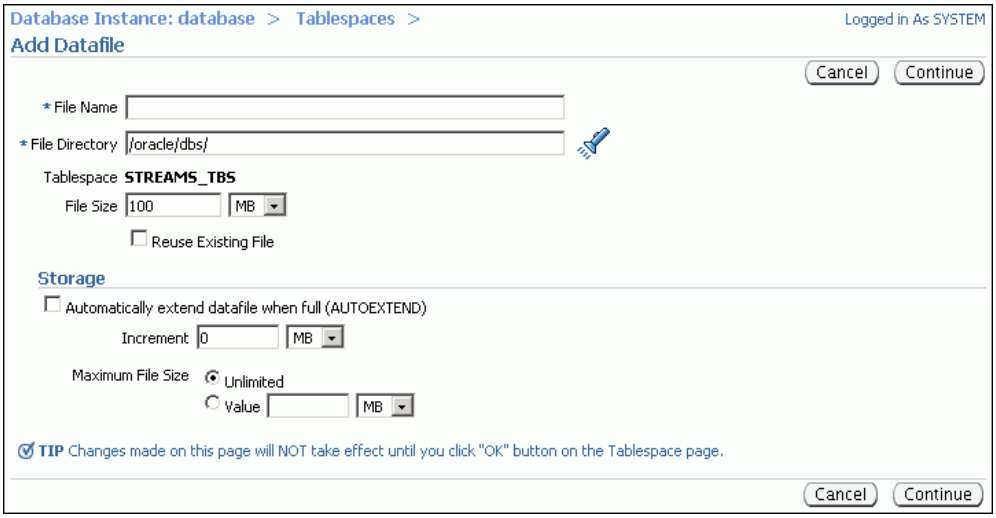
To create a new tablespace for the Oracle Streams administrator:

1. Log in to Enterprise Manager as an administrative user.
2. Go to the Database Home page for the database instance.
3. Click **Server** to open the Server subpage.
4. Click **Tablespaces** in the Storage section.
5. On the Tablespaces page, click **Create**.

   The Create Tablespace page appears, showing the General subpage.



6. Enter `streams_tbs` in the **Name** field.
7. Click **Add** in the Datafiles section to open the Add Datafile page.



8. Enter `streams_tbs.dbf` in the **File Name** field.
9. Check the directory in the **File Directory** field and change it if necessary.
10. Change the size in the **File Size** field to `25` and ensure that the list is set to `MB`.
11. Select **Automatically extend datafile when full (AUTOEXTEND)** in the Storage section.
12. Enter `5` in the **Increment** field and set the list to `MB`.
13. Set the **Maximum File Size**. Typically, it is best to leave it set to `Unlimited`.

14. Click **Continue**.
15. On the Create Tablespace page, click **OK**.
16. Complete the steps in "Tutorial: Creating an Oracle Streams Administrator" on page 2-2 to finish creating the Oracle Streams administrator.

Note: You can also use the `CREATE TABLESPACE` SQL statement to create a tablespace.

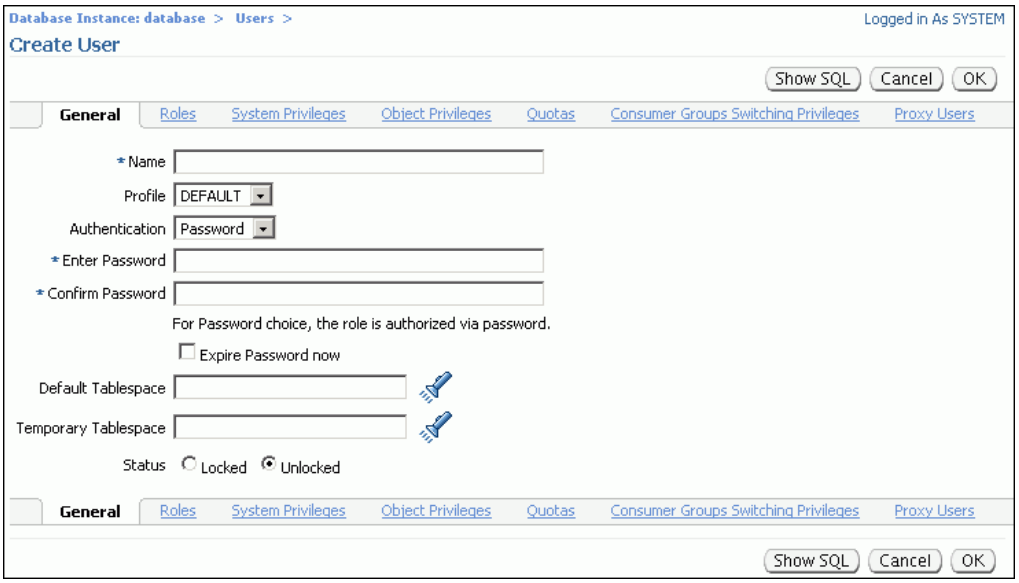
Tutorial: Creating the Oracle Streams Administrator

This topic describes creating an Oracle Streams administrator that uses the tablespace configured in "Tutorial: Creating the Tablespace for the Oracle Streams Administrator" on page 2-3.

To create a new Oracle Streams administrator named strmadmin:

1. Log in to Enterprise Manager as an administrative user.
2. Go to the Database Home page for the database instance.
3. Click **Server** to open the Server subpage.
4. Click **Users** in the Security section.
5. On the Users page, click **Create**.

   The General subpage of the Create User page appears.



6. Enter `strmadmin` in the **Name** field.
7. Enter a password for the new user in the **Enter Password** and **Confirm Password** fields.

   Enter an appropriate password for the administrative user. See *Oracle Database 2 Day + Security Guide* for information about choosing passwords.
8. Click the flashlight icon for the **Default Tablespace** field to select the `streams_tbs` tablespace created in "Tutorial: Creating the Tablespace for the Oracle Streams Administrator" on page 2-3.

9. Click the flashlight icon for the **Temporary Tablespace** field to select a temporary tablespace for the new user.
10. Click **Roles**.

    The Roles subpage of the Create User page appears.
11. Click **Edit List**.

    The Modify Roles page appears.
12. Move **DBA** from the Available Roles list to the Selected Roles list.
13. Click **OK**.
14. On the Create User page, click **OK** to create the user.

    Note: You can also use the `CREATE USER` SQL statement to create a user and the `GRANT` SQL statement to grant privileges to a user.

15. Grant additional privileges to the user with the `DBMS_STREAMS_AUTH.GRANT_ADMIN_PRIVILEGE` procedure.
    a. On a command line, open SQL*Plus and connect to the database as an administrative user who can grant privileges.

       See *Oracle Database 2 Day DBA* for more information about starting SQL*Plus.
    b. Run the following procedure:

```
BEGIN
  DBMS_STREAMS_AUTH.GRANT_ADMIN_PRIVILEGE(
    grantee          => 'strmadmin',
    grant_privileges => TRUE);
END;
/
```

If you were directed to this topic from another topic, then go back to the topic now:

- "Preparing for Messaging" on page 9-5
- "Tutorial: Configuring Two-Database Replication with Local Capture Processes" on page 4-25
- "Tutorial: Configuring Two-Database Replication with a Downstream Capture Process" on page 4-30
- "Tutorial: Configuring Hub-and-Spoke Replication with Local Capture Processes" on page 4-40
- "Tutorial: Configuring Two-Database Replication with Synchronous Captures" on page 4-46

Creating an ANYDATA Queue

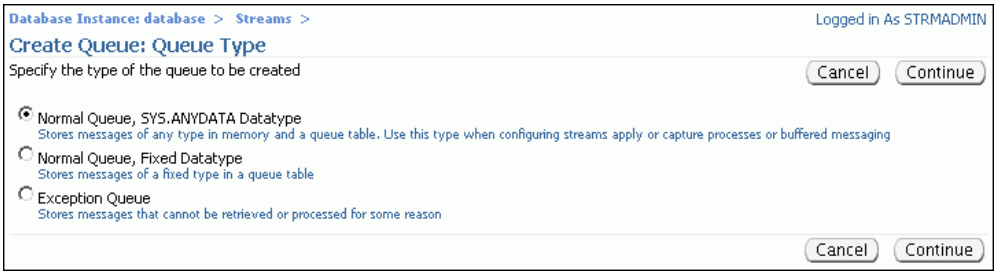
Queues store messages in an Oracle Streams environment. In an Oracle Streams replication environment, queues store messages that contain information about database changes. In an Oracle Streams messaging environment, queues store the messages produced and consumed by applications and users. Typically, each database in an Oracle Streams environment has one or more queues.

ANYDATA queues make it easy to store messages of almost any type. When you use an ANYDATA queue, you can, for example, store several different types of application messages in the same queue. Also, ANYDATA queues must be used to store information about database changes in an Oracle Streams replication environment.

To create an ANYDATA queue and its associated queue table:

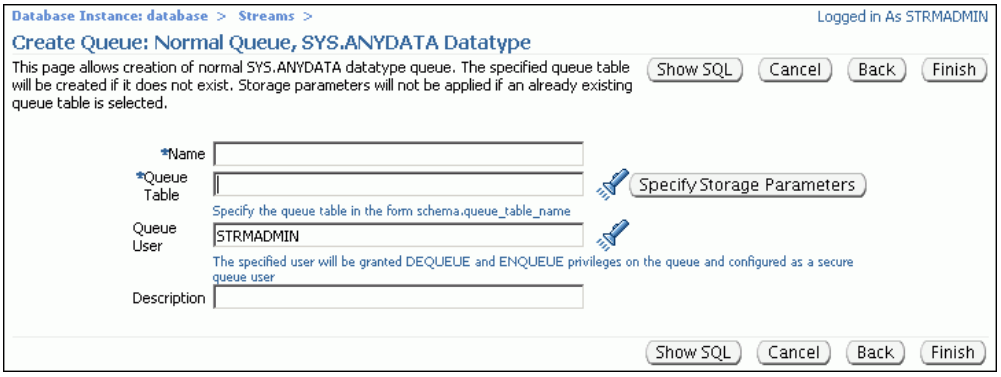
1. In Oracle Enterprise Manager, log in to the database as the Oracle Streams administrator.
2. Go to the Database Home page.
3. Click **Data Movement** to open the Data Movement subpage.
4. Click **Setup** in the Streams section.
5. On the Oracle Streams: Setup Options page, click **Messaging**.

   The Streams page appears, showing the Messaging subpage.
6. Click **Create** to open the Create Queue: Queue Type page.



7. Select **Normal Queue, SYS.ANYDATA Datatype**.

   A queue of the ANYDATA data type enables you to store messages of almost any type in a single queue.
8. Click **Continue** to open the Create Queue: Normal Queue, SYS.ANYDATA Datatype page.



9. Enter the name of the queue in the **Name** field. A typical queue name used in Oracle Streams environments is `streams_queue`, but you can enter a different name.
10. Enter the name of the queue table owner and the queue table name in the **Queue Table** field. Typically, the Oracle Streams administrator owns Oracle Streams queues, and a typical queue table name is `streams_queue_table`. Therefore, you can enter `strmadmin.streams_queue_table`, or you can enter a different owner and name.
11. Ensure that the name of the Oracle Streams administrator is entered in the **Queue User** field.
12. Optionally enter a description for the queue in the **Description** field.
13. Click **Finish** to create the queue table and the queue.

Note: You can also use the `DBMS_STREAMS_ADM.SET_UP_QUEUE` procedure to create an `ANYDATA` queue.

If you were directed to this topic from another topic, then go back to the topic now:

- "Task 2: Configuring the Queues and Propagation Between Them" on page 9-8
- "Task 2: Configuring a Queue and a Messaging Client" on page 9-19
- "Tutorial: Configuring Two-Database Replication with Synchronous Captures" on page 4-46

See Also:

- "Modifying Queues" on page 9-24

Tutorial: Creating a Database Link

To establish a communication path between two locations in a distributed database environment, you must create a database link. A **database link** is a pointer that defines a one-way communication path from one database to another database. An Oracle database uses database links to enable users on one database to access objects in a remote database. A local user can use a database link to a remote database even if the local user is not a user on the remote database.

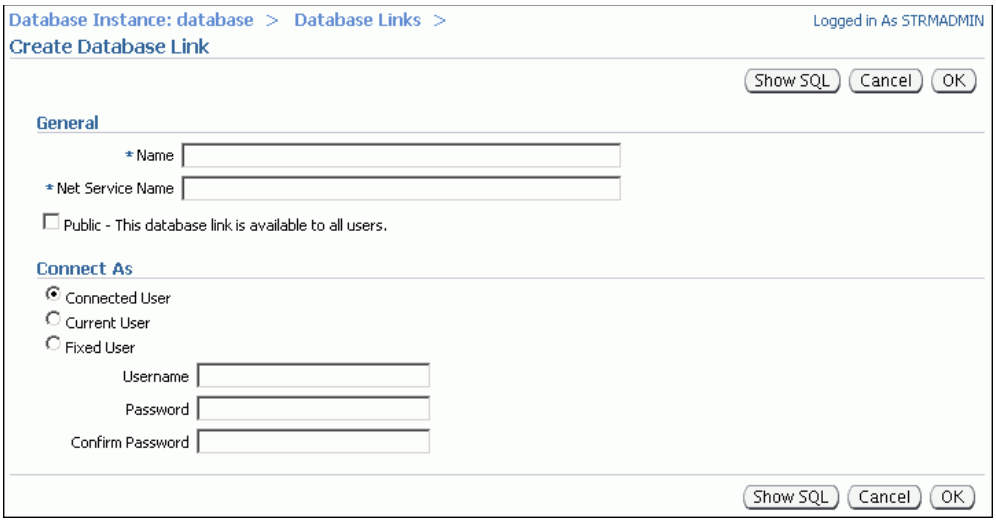
Database links are required in most environments that store data in multiple databases or share information between databases. These environment include those that use distributed SQL, Oracle Streams replication, materialized view replication, and messaging.

Because the `GLOBAL_NAMES` initialization parameter is set to `TRUE` for each database in your distributed environment, you must use a global database name when you establish a link between two databases. Doing so ensures that each database link connects to the correct remote database.

Before you can create a database link between two databases, you must configure network connectivity so that the databases can communicate with each other. See *Oracle Database 2 Day DBA* for information about configuring network connectivity between databases.

To create a database link from the ii1.example.com database to the ii2.example.com database:

1. Log in to Enterprise Manager as an administrative user, such as the Oracle Streams administrator `strmadmin` or `SYSTEM`. The database link is created in the schema of this user.
2. Go to the Database Home page for the `ii1.example.com` database instance.
3. Click **Schema** to open the Schema subpage.
4. Click **Database Links** in the Database Objects section.
5. On the Database Links page, click **Create** to open the Create Database Link page.



6. Enter the name of the database link in the **Name** field. The name must be the global name of the database to which you are linking. In this example, the database link name is `ii2.example.com`.
7. In the **Net Service Name** field, enter the net service name of the database to which you are linking. In this example, the net service name is `ii2.example.com`.
8. Select **Fixed User** in the Connect As section.
9. In the **Username** field, enter the user name of the user who will own the database link. The database link connects to this user on the remote database. In this example, you can enter an administrative user, such as system `SYSTEM`, the Oracle Streams administrator `strmadmin`, or a regular database user, such as `oe`.
10. In the **Password and Confirm Password** fields, enter the password for the specified user on the remote database.
11. Click **OK** to create the database link.

> **Note:** You can also use the `CREATE DATABASE LINK` SQL statement to create a database link.

If you were directed to this topic from another topic, then go back to the topic now:

- "Tutorial: Querying Multiple Oracle Databases" on page 3-4
- "Tutorial: Modifying Data in Multiple Oracle Databases" on page 3-5
- "Tutorial: Running a Stored Procedure in a Remote Oracle Database" on page 3-7

- "Configuring Oracle Databases to Work with Non-Oracle Databases" on page 3-8
- "Task 2: Configuring the Queues and Propagation Between Them" on page 9-8
- "Tutorial: Configuring Two-Database Replication with Local Capture Processes" on page 4-25
- "Tutorial: Configuring Two-Database Replication with a Downstream Capture Process" on page 4-30
- "Tutorial: Configuring Hub-and-Spoke Replication with Local Capture Processes" on page 4-40
- "Tutorial: Configuring Two-Database Replication with Synchronous Captures" on page 4-46
- "Tutorial: Adding Databases to a Replication Environment" on page 6-7
- "Tutorial: Preparing to Compare and Converge Data" on page 10-3

See Also:

- "Setting the GLOBAL_NAMES Initialization Parameter to TRUE" on page 2-1
- "About Data Replication and Integration" on page 1-2
- "About Data Replication and Integration Features" on page 1-3

3

Accessing and Modifying Information in Multiple Databases

This chapter describes how to access and modify information in multiple Oracle and non-Oracle databases.

This chapter contains the following sections:

About Accessing and Modifying Information in Multiple Databases

While connected to an Oracle database, you can access and modify information in other Oracle databases and in non-Oracle databases. When information in two or more databases appears to be in a single database, it is called **federation**. Federation leaves information in its original location, where it is maintained and updated. Multiple data sources appear to be integrated into a single virtual database so that different kinds of databases are presented in one consolidated view. A federated configuration can make all of your databases look like one virtual database to applications and end users, thereby reducing some of the complexity of the distributed system.

The following topics contain more information about accessing and modifying information in multiple databases:

See Also:

- "When to Access and Modify Information in Multiple Databases" on page 1-4

About Distributed SQL

Distributed SQL enables applications and users to query or modify information in multiple databases with a single SQL statement. Because distributed SQL masks the physical location of your data, you can change the location of your data without changing your application. Distributed SQL includes the following: **distributed queries** (which access data) and **distributed transactions** (which modify data). In distributed transactions, the **two-phase commit mechanism** guarantees the integrity of your data by ensuring that all statements in a transaction either commit or roll back as a unit at each database involved in the distributed transaction.

When an application or user tries to commit a distributed transaction, the database to which the application or user is connected is called the **global coordinator**. The global coordinator completes the two-phase commit by initiating the following phases:

- **Prepare Phase:** The global coordinator asks the other databases involved in the distributed transaction to confirm that they can either commit or roll back the transaction, even if there is a failure. If any database cannot complete the prepare phase, then the transaction is rolled back.
- **Commit Phase:** If all of the other databases inform the global coordinator that they are prepared, then the global coordinator commits the transaction and asks all of the other databases to commit the transaction.

See Also:

- *Oracle Database Administrator's Guide*

About Synonyms and Location Transparency

A **synonym** is a database object that acts as an alias for another database object. You can create both public and private synonyms. Every database user can access a public synonym. A private synonym is in the schema of a specific user, and only users who are granted access to the private synonym can use it.

In a distributed environment, synonyms can provide location transparency for database objects. A synonym hides the location of a database object from applications and users. If the database object must be moved or renamed, then you can redefine the synonym, and applications and users can continue to use the synonym without any modifications.

For example, suppose an application must access the `hr.employees` table at a remote database with the global name `ii2.example.com`. A database link exists for the remote database, and the name of the database link is `ii2.example.com`. In this case, you can create a synonym named `employees` in the `hr` schema that points to the table `hr.employees@ii2.example.com`. After the synonym is in place, the application at the local database can use `hr.employees` to access the remote table.

About Accessing and Modifying Information in Non-Oracle Databases

You can use distributed SQL to federate data not only in an Oracle database, but in non-Oracle databases as well. Oracle Database Gateway enables Oracle databases to access and modify data in a number of non-Oracle databases, including Sybase, DB2, Informix, Microsoft SQL Server, Ingres, and Teradata databases. This access is completely transparent to the end user. That is, you can issue the same SQL statements regardless of whether you are accessing data in an Oracle database or a non-Oracle database.

Note: The capabilities of a specific Oracle Database Gateway are limited by the capabilities of the non-Oracle database being accessed. For information about the limitations of a specific Oracle Database Gateway, see the Oracle documentation for that specific gateway.

See Also:

- "Working with Data in Non-Oracle Databases" on page 3-8

About Stored Procedures

To reduce network traffic when performing complex operations in a federated environment, you can use stored procedures. A **procedure** or **function** is a schema object that is run to solve a specific problem or perform a set of related tasks. Procedures and functions are identical except that functions always return a single value to the caller, while procedures do not. Generally, you use a procedure to perform an action, and you use a function to compute a value. In this guide, the general term **stored procedure** includes both procedures and functions.

Oracle databases support stored procedures that are written in PL/SQL or Java, but this guide discusses only PL/SQL stored procedures. PL/SQL stored procedures consist of a set of SQL statements and other PL/SQL constructs that are grouped together and stored in the database. Stored procedures let you combine the ease and flexibility of SQL with the procedural functionality of a structured programming language.

As with SQL statements, to run a stored procedure, you do not need to be aware of its physical location. Similarly, by using the appropriate Oracle Database Gateway, you can even call a stored procedure that is in a non-Oracle database. In this case, the gateway maps the PL/SQL calls to the non-Oracle database stored procedures.

See Also:

- "Tutorial: Running a Stored Procedure in a Remote Oracle Database" on page 3-7
- *Oracle Database PL/SQL Language Reference* for information about PL/SQL stored procedures
- *Oracle Database Java Developer's Guide* for information about Java stored procedures

Preparing to Access and Modify Information in Multiple Oracle Databases

This topic describes actions that are required to prepare your databases to access and modify information at other databases.

To prepare to access and modify information in multiple databases:

1. Set the `GLOBAL_NAMES` initialization parameter to `TRUE` at each Oracle database in the distributed environment. See "Setting the GLOBAL_NAMES Initialization Parameter to TRUE" on page 2-1 for instructions.
2. Configure network connectivity so that the databases can communicate with each other. See *Oracle Database 2 Day DBA* for information about configuring network connectivity between databases.

Tutorial: Querying Multiple Oracle Databases

A **distributed query** accesses information in two or more databases. In a synonym or in a SELECT statement, you can identify a remote table, view, or materialized view by appending *@dblink* to the end of its name. The *dblink* is a database link to the database that contains the remote database object.

Meet the following conditions before running the distributed query in this topic:

- Satisfy the prerequisites described in "Preparing to Access and Modify Information in Multiple Oracle Databases" on page 3-3.
- Create a database link from the local database to any remote database that contains a database object involved in the query. In the example in this topic, the SYSTEM user at the ii1.example.com database uses a database link that connects to the SYSTEM user at the ii2.example.com database. See "Tutorial: Creating a Database Link" on page 2-8 for information about creating such a database link.
- Ensure that the hr sample schema is installed on the local database, and the oe sample schema is installed on the remote database. These sample schemas are installed by default with Oracle Database.

This topic uses ii1.example.com and ii2.example.com as sample databases. You can substitute any two databases in your environment that meet these conditions.

For this example, assume the following:

- A company keeps its human resources information in the hr schema at the ii1.example.com database and its order entry information in the ii2.example.com database.
- The employee_id in the hr.employees table corresponds with the sales_rep_id in the oe.orders table.
- A manager has a question about an order and wants to contact the sales representative for the order.

In this case, the contact information for the sales representative is in the hr.employees table in the ii1.example.com database, and the order information is in the oe.orders table in the ii2.example.com database.

To run a distributed query that combines the information at the ii1.example.com and ii2.example.com databases to show the contact information for the sales representative:

1. Create a synonym for the remote database object. In this example, create a synonym called ord in the hr schema that points to the oe.orders table at the ii2.example.com database:
   a. Log in to Enterprise Manager as SYSTEM user.
   b. Go to the Database Home page for the ii1.example.com database instance.
   c. Click **Schema** to open the Schema subpage.
   d. Click **Synonyms** in the Database Objects section.
   e. On the Synonyms page, click **Create**.
   f. On the Create Synonym page, enter ord in the **Name** field in the General section.
   g. With Schema selected for the Type, enter hr in the **Schema** field or click the flashlight icon to select the hr schema.

h. In the Database section, select **Remote**.

i. Click the flashlight icon for the **Service Name** field to select the `ii2.example.com` database link for the `SYSTEM` user.

j. In the As Alias For section, enter `oe.orders`.

k. Click **OK** to create the synonym.

Note: You can also use the `CREATE SYNONYM` SQL statement to create a synonym.

---

2. On a command line, open SQL*Plus and connect to the `ii1.example.com` database as the `SYSTEM` user.

   See *Oracle Database 2 Day DBA* for more information about starting SQL*Plus.

3. Run the following query:

```
COLUMN FIRST_NAME HEADING 'First Name' FORMAT A20
COLUMN LAST_NAME HEADING 'Last Name' FORMAT A20
COLUMN PHONE_NUMBER HEADING 'Phone Number' FORMAT A20

SELECT e.first_name, e.last_name, e.phone_number
  FROM hr.employees e, hr.ord o
  WHERE o.order_id    = 2456 AND
        e.employee_id = o.sales_rep_id;
```

   The output will be similar to the following:

```
First Name           Last Name            Phone Number
-------------------- -------------------- --------------------
Danielle             Greene               011.44.1346.229268
```

See Also:

- "When to Access and Modify Information in Multiple Databases" on page 1-4

Tutorial: Modifying Data in Multiple Oracle Databases

A **distributed transaction** includes one or more statements that, individually or as a group, modify data or the structure of database objects in two or more databases. In a synonym or in a statement that modifies data in a remote database, you can identify a database object by appending `@`*`dblink`* to the end of its name. The *`dblink`* is a database link to the database that contains the remote database object.

Statements that modify data in tables are called data manipulation language (DML) statements. Statements that modify the structure of database objects are called data definition language (DDL) statements. Both DML and DDL statements can be part of a distributed transaction.

Meet the following conditions before running the distributed transaction in this topic:

- Satisfy the prerequisites described in "Preparing to Access and Modify Information in Multiple Oracle Databases" on page 3-3.
- Create a database link from the local database to any remote database that contains a database object involved in the transaction. In the example in this topic, the `SYSTEM` user at the `ii1.example.com` database uses a database link that

connects to the SYSTEM user at the ii2.example.com database. See "Tutorial: Creating a Database Link" on page 2-8 for information about creating such a database link.

- Ensure that the hr sample schema is installed on the local database, and the oe sample schema is installed on the remote database. These sample schemas are installed by default with Oracle Database.

This topic uses ii1.example.com and ii2.example.com as sample databases. You can substitute any two databases in your environment that meet these conditions.

For this example, assume the following:

- A company keeps its human resources information in the hr schema at the ii1.example.com database and its order entry information in the ii2.example.com database.
- The employee_id in the hr.employees table corresponds with the sales_rep_id in the oe.orders table.
- A sales representative has been promoted to the job of sales manager. The employee_id of this sales representative in the hr.employees table is 154. The job_id and manager_id data for this employee must change in the hr.employees table.
- The current orders for the promoted sales representative must be transferred to a different sales representative. The employee_id of the sales representative who is now responsible for the orders is 148 in the hr.employees table. All of the orders in the oe.orders table currently assigned to sales_rep_id 154 must be changed to sales_rep_id 148.

The company wants these changes to be committed in a single distributed transaction.

To run a distributed transaction that changes data at both the ii1.example.com and ii2.example.com databases:

1. Create a synonym for the remote database object. In this example, create a synonym called ord in the hr schema that points to the oe.orders table at the ii2.example.com database. Step 1 in "Tutorial: Querying Multiple Oracle Databases" on page 3-4 contains instructions for creating this synonym.
2. On a command line, open SQL*Plus and connect to the ii1.example.com database as the SYSTEM user.

   See *Oracle Database 2 Day DBA* for more information about starting SQL*Plus.
3. Update the data at each database and commit your changes:

```
UPDATE hr.employees SET
  job_id = 'SA_MAN', manager_id = 100
  WHERE employee_id = 154;

UPDATE hr.ord
  SET   sales_rep_id = 148
  WHERE sales_rep_id = 154;

COMMIT;
```

See Also:

- "When to Access and Modify Information in Multiple Databases" on page 1-4

Tutorial: Running a Stored Procedure in a Remote Oracle Database

A **remote procedure call (RPC)** runs a procedure or function at a remote database. An RPC performs any work defined in the remote procedure. To run a remote procedure or function, you can identify the remote procedure or function by appending `@dblink` to the end of its name. The *dblink* is a database link to the database that contains the remote procedure or function. You can also create a synonym that points to the remote procedure or function.

Meet the following conditions before performing the sample RPC in this topic:

- Satisfy the prerequisites described in "Preparing to Access and Modify Information in Multiple Oracle Databases" on page 3-3.
- Create a database link from the local database to any remote database that contains a procedure or function that is being called. In the example in this topic, the `SYSTEM` user at the `ii1.example.com` database uses a database link that connects to the `SYSTEM` user at the `ii2.example.com` database. See "Tutorial: Creating a Database Link" on page 2-8 for information about creating such a database link.
- Ensure that the `hr` sample schema is installed on the remote database. The `hr` sample schema is installed by default with Oracle Database.

This topic uses `ii1.example.com` and `ii2.example.com` as sample databases. You can substitute any two databases in your environment that meet these conditions.

For this example, assume the following:

- A company keeps its human resources information in the `hr` schema at the `ii2.example.com` database.
- The `hr` schema does not exist at the local `ii1.example.com` database.
- When an employee leaves the company, a procedure called `add_job_history` in the `hr` schema inserts a row into the `hr.job_history` table. The row contains information about the history of the employee with the company.
- The employee with an `employee_id` of `127` is leaving the company.
- You want to connect to the `ii1.example.com` database and run the `add_job_history` procedure at the `ii2.example.com` database to record the job history for employee `127`.

To run an RPC to record the job history of the employee:

1. On a command line, open SQL*Plus and connect to the `ii1.example.com` database as the `SYSTEM` user.

   See *Oracle Database 2 Day DBA* for more information about starting SQL*Plus.

2. Run the `add_job_history` procedure at the `ii2.example.com` database to record the job history of the employee:

   ```
   exec hr.add_job_history@ii2.example.com(127,'14-JAN-99','26-JUN-06','ST_
   CLERK',50);
   ```

3. Commit the changes:

   ```
   COMMIT;
   ```

   This step is not necessary if the remote procedure commits.

4. Optionally, query the `hr.job_history` table at the `ii2.example.com` database to see the inserted row:

```
SELECT * FROM hr.job_history@ii2.example.com ORDER BY employee_id;
```

The output will be similar to the following:

```
EMPLOYEE_ID START_DAT END_DATE  JOB_ID     DEPARTMENT_ID
----------- --------- --------- ---------- -------------
        101 21-SEP-89 27-OCT-93 AC_ACCOUNT            110
        101 28-OCT-93 15-MAR-97 AC_MGR                110
        102 13-JAN-93 24-JUL-98 IT_PROG                60
        114 24-MAR-98 31-DEC-99 ST_CLERK               50
        122 01-JAN-99 31-DEC-99 ST_CLERK               50
        127 14-JAN-99 26-JUN-06 ST_CLERK               50
        176 24-MAR-98 31-DEC-98 SA_REP                 80
        176 01-JAN-99 31-DEC-99 SA_MAN                 80
        200 17-SEP-87 17-JUN-93 AD_ASST                90
        200 01-JUL-94 31-DEC-98 AC_ACCOUNT             90
        201 17-FEB-96 19-DEC-99 MK_REP                 20
```

Notice that the job history of the employee with an `employee_id` of `127` is recorded in the table.

See Also:

- "When to Access and Modify Information in Multiple Databases" on page 1-4

Working with Data in Non-Oracle Databases

Oracle Database Gateway enables Oracle databases to access and modify data in a number of non-Oracle databases, including Sybase, DB2, Informix, Microsoft SQL Server, Ingres, and Teradata databases. For the best performance and usability, you should follow established best practices for working with non-Oracle databases.

This section contains these topics:

- Configuring Oracle Databases to Work with Non-Oracle Databases
- Best Practices for Working with Non-Oracle Databases

See Also:

- "When to Access and Modify Information in Multiple Databases" on page 1-4
- "About Accessing and Modifying Information in Non-Oracle Databases" on page 3-2

Configuring Oracle Databases to Work with Non-Oracle Databases

Before you can query data, modify data, or run a stored procedure in a non-Oracle database, you must complete several tasks.

To begin to work with data in non-Oracle databases:

1. Install and configure Oracle Database Gateway software for each non-Oracle database. Oracle Database Gateway software can be installed on the computer system running an Oracle database, on the computer system running a non-Oracle database, or on a third computer system. For information about installing and configuring a specific Oracle Database Gateway, see the Oracle documentation for that specific gateway.

2. Configure Oracle Net Services so that the Oracle database can communicate with the listener for Oracle Database Gateway that was configured in Step 1. See *Oracle Database Net Services Administrator's Guide* for instructions.

   After communication is established with Oracle Database Gateway, the Oracle database can communicate with the non-Oracle database. An Oracle database can use Heterogeneous Services to communicate with Oracle Database Gateway. See *Oracle Database Heterogeneous Connectivity Administrator's Guide* for information about configuration options for Heterogeneous Services.

3. Create a database link to the non-Oracle database. The database link must be created with an explicit `CONNECT TO` clause. After you complete Step 1 and 2, you can create the database link in the same way that you would to connect to an Oracle database. See "Tutorial: Creating a Database Link" on page 2-8 for instructions.

After completing these steps, you can access data and procedures in the non-Oracle database transparently. To work with the non-Oracle database, follow the instructions in these topics and specify database objects in the non-Oracle database:

- "Tutorial: Querying Multiple Oracle Databases" on page 3-4
- "Tutorial: Modifying Data in Multiple Oracle Databases" on page 3-5
- "Tutorial: Running a Stored Procedure in a Remote Oracle Database" on page 3-7

Best Practices for Working with Non-Oracle Databases

Oracle Database Gateway performance is affected by several factors, including network speed, available memory, amount of data being transferred from one database to the other, and the number of concurrent sessions. Some of these factors can be adjusted for better performance.

You can achieve better performance by following these best practices:

- Reduce Post Processing
- Tune the Non-Oracle Database
- Set the Relevant Initialization Parameters
- Check the Location of the Oracle Database Gateway Installation
- Ensure Adequate Memory
- Consider Case Differences

Reduce Post Processing

Performance can be affected negatively if there is a large amount of post processing. Ensure that as much of each SQL statement as possible is processed on the non-Oracle database to achieve better performance. If parts of the `WHERE` clause are missing from the SQL that is sent to the non-Oracle database or if joins are split, then the query will be post processed.

Follow these best practices to reduce post processing:

- Avoid SQL functions in the `WHERE` clause, if possible. SQL functions in `WHERE` clauses affect the performance of the gateway. If you are using Oracle functions that do not have an equivalent in the non-Oracle database, then the gateway compensates for it. The data is retrieved from the non-Oracle database and the function is applied on the Oracle database.

- Use hints to improve the query plan. Look at the gateway trace file or explain plan to determine the SQL being sent to the non-Oracle database. For example, if the SQL statement includes joins of tables on the non-Oracle database, and they are being post processed, then you can use hints to cause the joins to be performed on the non-Oracle database.

Tune the Non-Oracle Database

Performance can be affected negatively if the optimizer does not have enough information to generate an optimal plan. The Oracle optimizer uses table and index statistics of the non-Oracle database to determine the most optimal path to access the data in the non-Oracle database. If this information is missing or inaccurate, then the access path is not optimal. Defining indexes on the non-Oracle database improves the performance of the gateway. See *Oracle Database 2 Day + Performance Tuning Guide* for more information about the Oracle optimizer.

Set the Relevant Initialization Parameters

Customizing the following initialization parameters can improve performance:

- `HS_RPC_FETCH_SIZE`
- `HS_FDS_FETCH_ROWS`
- `HS_LANGUAGE`

For example if you are sure that the data you are accessing in the non-Oracle database is in the same character set as the one used by the Oracle database, then set the `HS_LANGUAGE` initialization parameter to the character set of the Oracle database.

Check the Location of the Oracle Database Gateway Installation

The location of the Oracle Database Gateway installation might affect performance. For example, if CPU is a constraint, then do not install the Oracle Database Gateway on the same computer systems where Oracle databases are running.

Ensure Adequate Memory

Ensure that there is enough memory on the computer system where the Oracle Database Gateway is running. There are several factors that affect memory requirements. These include the SQL statement being processed, the number of concurrent sessions, the number of open cursors, and the number of columns in the table being accessed.

Consider Case Differences

When you are working with non-Oracle databases, remember that an Oracle database defaults characters to uppercase unless you surround identifiers with double quotation marks. For example, to refer to the Sybase table named `emp`, enter the name with double quotation marks, as in the following example:

```
SELECT * FROM "emp"@SYBS;
```

However, to refer to the Sybase table called `emp` owned by user `Smith` from an Oracle application, enter the following:

```
SELECT * FROM "Smith"."emp"@SYBS;
```

If the Sybase table named `emp` is owned by `SMITH`, a table owner name in uppercase letters, then you can enter the owner name without double quotation marks, as in the following examples:

```
SQL> SELECT * FROM SMITH."emp"@SYBS;

SQL> SELECT * FROM smith."emp"@SYBS;

SQL> SELECT * FROM SmItH."emp"@SYBS;
```

See Also:

- The Oracle documentation for your specific Oracle Database Gateway
- *Oracle Database Heterogeneous Connectivity Administrator's Guide*

4

Replicating Data Using Oracle Streams

This chapter contains conceptual information about Oracle Streams replication and describes how to replicate data continuously between databases.

This chapter contains the following sections:

- About Oracle Streams Replication
- Preparing for Oracle Streams Replication
- Configuring Oracle Streams Replication: Examples

See Also:

- Chapter 5, "Administering an Oracle Streams Replication Environment"
- Chapter 6, "Extending an Oracle Streams Replication Environment"
- *Oracle Streams Replication Administrator's Guide*
- Chapter 7, "Replicating Data Using Materialized Views"

About Oracle Streams Replication

Replication is the process of sharing database objects and data at multiple databases. To maintain the database objects and data at multiple databases, a change to one of these database objects at a database is shared with the other databases. In this way, the database objects and data are kept synchronized at all of the databases in the replication environment.

Some replication environments must continually replicate the changes made to shared database objects. **Oracle Streams** is the Oracle Database feature for continuous replication. Typically, in such environments, the databases that contain the shared database objects are connected to the network nearly all the time and continually push database changes over these network connections.

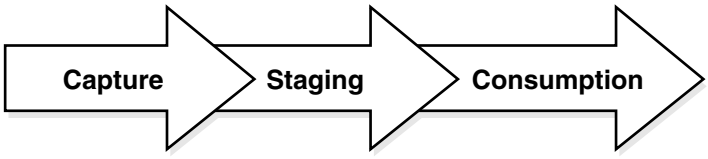
When a change is made to one shared database object, Oracle Streams performs the following actions to ensure that the same change is made to the corresponding shared database object at each of the other databases:

1. Oracle Streams automatically captures the change and stages it in a queue.
2. Oracle Streams automatically pushes the change to a queue in each of the other databases that contain the shared database object.

3. Oracle Streams automatically consumes the change at each of the other databases. During consumption, Oracle Streams dequeues the change and applies the change to the shared database object.

Figure 4–1 shows the Oracle Streams information flow:

Figure 4–1 Oracle Streams Information Flow



You can use Oracle Streams replication to share data at multiple databases and efficiently keep the data current at these databases. For example, a company with several call centers throughout the world might want to store customer information in a local database at each call center. In such an environment, continuous replication with Oracle Streams can ensure that a change made to customer data at one location is pushed to all of the other locations as soon as possible.

When you use Oracle Streams to capture changes to database objects, the changes are formatted into **logical change records (LCRs)**. An LCR is a message with a specific format that describes a database change. If the change was a data manipulation language (DML) operation, then a **row LCR** encapsulates each row change resulting from the DML operation. One DML operation might result in multiple row changes, and so one DML operation might result in multiple row LCRs. If the change was a data definition language (DDL) operation, then a single **DDL LCR** encapsulates the DDL change.

The following topics describe Oracle Streams replication in more detail:

- About Change Capture
- About Change Propagation Between Databases
- About Change Apply
- About Rules for Controlling the Behavior of Capture, Propagation, and Apply
- About Rule-Based Transformations for Nonidentical Copies
- About Supplemental Logging
- About Conflicts and Conflict Resolution
- About Tags for Avoiding Change Cycling
- About the Common Types of Oracle Streams Replication Environments
- About the Oracle Streams Replication Configuration Procedures
- About Key Oracle Streams Supplied PL/SQL Packages and Data Dictionary Views

See Also:

- "When to Replicate Data with Oracle Streams" on page 1-4

About Change Capture

Oracle Streams provides two ways to capture database changes automatically:

- A **capture process** should be used to capture data manipulation language (DML) changes to a relatively large number of tables, an entire schema, or a database. Also, a capture process must be used to capture data definition language (DDL) changes to tables and other database objects. See "About Change Capture with a Capture Process" on page 4-3.
- A **synchronous capture** should be used to capture DML changes to a relatively small number of tables. See "About Change Capture with a Synchronous Capture" on page 4-4.

A single capture process or a single synchronous capture can capture changes made to only one database. The database where a change originated is called the **source database** for the change.

Note: The examples in this guide replicate DML changes only. You should understand the implications of replicating DDL changes before doing so. See *Oracle Streams Replication Administrator's Guide* and *Oracle Database PL/SQL Packages and Types Reference* for information about replicating DDL changes.

See Also:

- "Managing Capture Processes" on page 5-2
- "Preparing for Oracle Streams Replication" on page 4-21

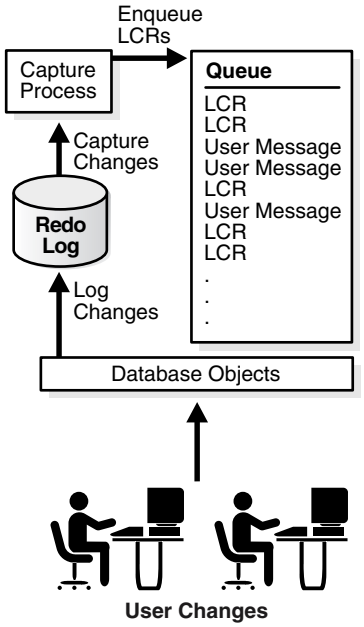
About Change Capture with a Capture Process

A **capture process** is an optional Oracle Database background process that asynchronously captures changes recorded in the redo log. When a capture process captures a database change, it converts it into a logical change record (LCR) and enqueues the LCR.

A capture process is always associated with a single queue, and it enqueues LCRs into this queue only. For improved performance, captured LCRs are always stored in a **buffered queue**, which is System Global Area (SGA) memory associated with a queue.

Figure 4–2 shows how a capture process works.

Figure 4–2 Capture Process



A capture process can run on the source database or on a remote database. When a capture process runs on the source database, the capture process is called a **local capture process**. When a capture process runs on a remote database, the capture process is called a **downstream capture process**.

With downstream capture, redo transport services use the log writer process (LGWR) at the source database to send redo data to the database that runs the downstream capture process. A downstream capture process requires fewer resources at the source database because a different database captures the changes. A local capture process, however, is easier to configure and manage than a downstream capture process. Local capture processes also provide more flexibility in replication environments with different platforms or different versions of Oracle Database.

See Also:

- "Tutorial: Configuring Two-Database Replication with a Downstream Capture Process" on page 4-30
- "Tutorial: Configuring Hub-and-Spoke Replication with Local Capture Processes" on page 4-40
- "Managing Capture Processes" on page 5-2
- "Monitoring Capture Processes" on page 5-12
- *Oracle Streams Concepts and Administration*
- *Oracle Database 2 Day DBA*

About Change Capture with a Synchronous Capture

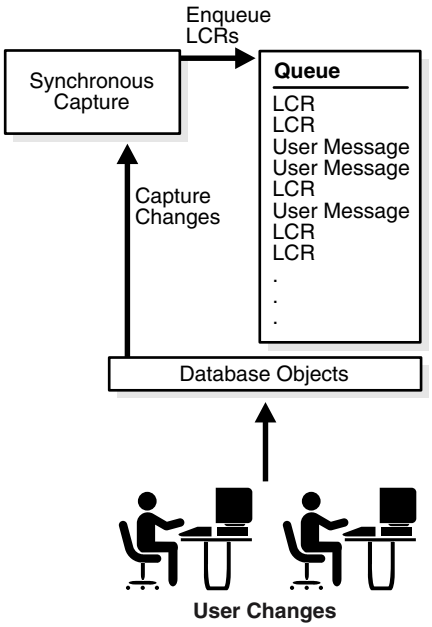
Instead of asynchronously mining the redo log, a **synchronous capture** uses an internal mechanism to capture data manipulation language (DML) changes when they are made to tables. A single DML change can result in changes to one or more rows in the table. A synchronous capture captures each row change, converts it into a row logical change record (LCR), and enqueues it.

A synchronous capture is always associated with a single queue, and it enqueues row LCRs into this queue only. Synchronous capture always enqueues row LCRs into the **persistent queue**. The persistent queue is the portion of a queue that stores messages on hard disk in a queue table, not in memory.

It is usually best to use synchronous capture in a replication environment that captures changes to a relatively small number of tables. If you must capture changes to many tables, to an entire schema, or to an entire database, then you should use a capture process instead of a synchronous capture.

Figure 4–3 shows how a synchronous capture works.

Figure 4–3 Synchronous Capture



Note: If you are using Oracle Database 11*g* Standard Edition, then synchronous capture is the only Oracle Streams component that can capture database changes automatically. To use capture processes, you must have Oracle Database 11*g* Enterprise Edition.

See Also:

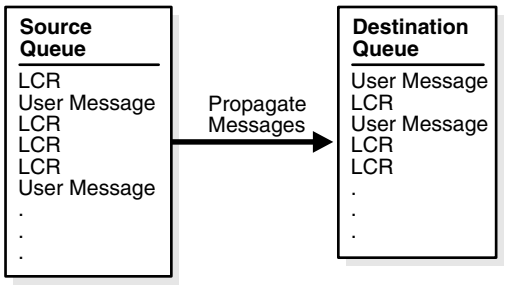
- "Tutorial: Configuring Two-Database Replication with Synchronous Captures" on page 4-46
- *Oracle Streams Concepts and Administration*

About Change Propagation Between Databases

A **propagation** sends messages from one queue to another. You can use Oracle Streams to configure message propagation between two queues in the same database or in different databases. Oracle Streams uses a database link and Oracle Scheduler jobs to send messages. A propagation is always between a **source queue** and a **destination queue**. In an Oracle Streams replication environment, a propagation typically sends messages that describe database changes (in the form of LCRs) from a source queue in the local database to a destination queue in a remote database.

Figure 4–4 shows a propagation.

Figure 4–4 Propagation



See Also:

- "Enabling and Disabling a Propagation" on page 5-4
- "Monitoring Propagations" on page 5-16
- "Tutorial: Creating a Database Link" on page 2-8
- *Oracle Database Administrator's Guide* for information about Oracle Scheduler

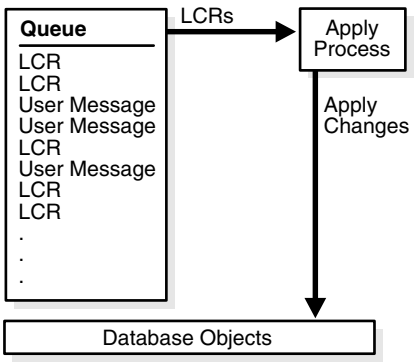
About Change Apply

After database changes have been captured and propagated, they reside in a queue and are ready to be applied to complete the replication process. An **apply process** is an optional Oracle Database background process that dequeues logical change records (LCRs) and other types of messages from a specific queue. In a simple Oracle Streams replication environment, an apply process typically applies the changes in the LCRs that it dequeues directly to the database objects in the local database.

An apply process is always associated with a single queue, and it dequeues messages from this queue only. A single apply process either can dequeue messages from the buffered queue or from the persistent queue, but not both. Therefore, if an apply process applies changes that were captured by a capture process, then the apply process must be configured to dequeue LCRs from the buffered queue. However, if an apply process applies changes that were captured by a synchronous capture, then the apply process must be configured to dequeue LCRs from the persistent queue.

Figure 4–5 shows how an apply process works.

Figure 4–5 Apply Process



When an apply process cannot apply an LCR successfully, it moves the LCR, and all of the other LCRs in the transaction, to a special queue called the **error queue**. The error queue contains all of the current apply errors for a database. If there are multiple apply processes in a database, then the error queue contains the apply errors for each apply process. You can correct the condition that caused an error and then reexecute the corresponding transaction in the error queue to apply its changes. For example, you might modify a row in a table to correct the condition that caused an error in a transaction and then reexecute the transaction.

For an apply process to apply changes to a database object, an **instantiation system change number (SCN)** must be set for the database object. An instantiation SCN is the SCN for a database object that specifies that only changes that were committed after the SCN at the source database are applied by an apply process. The instantiation SCN for a table assumes that the table is consistent at the source and destination database at the specified SCN. Typically, the instantiation SCN is set automatically when you configure the Oracle Streams replication environment.

Note: An apply process can also pass a message that it dequeues as a parameter to a user-defined procedure called an **apply handler** for custom processing. Apply handlers are beyond the scope of this guide. See *Oracle Streams Concepts and Administration* for more information.

See Also:

- "Managing Apply Processes" on page 5-6
- "Monitoring Apply Processes" on page 5-19

About Rules for Controlling the Behavior of Capture, Propagation, and Apply

An Oracle Streams replication configuration must identify what to replicate. Capture processes, synchronous captures, propagations, and apply processes are called **Oracle Streams clients**. **Rules** determine what Oracle Streams clients replicate. You can configure rules for each Oracle Streams client independently, and the rules for different Oracle Streams clients do not need to match.

Rules can be organized into **rule sets**, and the behavior of each Oracle Streams client is determined by the rules in the rule sets that are associated with it. You can associate a **positive rule set** and a **negative rule set** with a capture process, a propagation, and an apply process, but a synchronous capture can only have a positive rule set.

In a replication environment, an Oracle Streams client performs its task if a database change satisfies its rule sets. In general, a change satisfies the rule sets for an Oracle Streams client if *no rules* in the negative rule set evaluate to `TRUE` for the change, and at least one rule in the positive rule set evaluates to `TRUE` for the change. The negative rule set is always evaluated first.

Specifically, you use rule sets in an Oracle Streams replication environment to do the following:

- Specify the changes that a capture process captures from the redo log or discards. If a change found in the redo log satisfies the rule sets for a capture process, then the capture process captures the change. If a change found in the redo log does not satisfy the rule sets for a capture process, then the capture process discards the change.

- Specify the changes that a synchronous capture captures. If a data manipulation language (DML) change satisfies the rule set for a synchronous capture, then the synchronous capture captures the change immediately after the change is committed. If a DML change made to a table does not satisfy the rule set for a synchronous capture, then the synchronous capture does not capture the change.
- Specify the changes (encapsulated in LCRs) that a propagation sends from one queue to another or discards. If an LCR in a queue satisfies the rule sets for a propagation, then the propagation sends the LCR. If an LCR in a queue does not satisfy the rule sets for a propagation, then the propagation discards the LCR.
- Specify the LCRs that an apply process dequeues or discards. If an LCR in a queue satisfies the rule sets for an apply process, then the apply process dequeues and processes the LCR. If an LCR in a queue does not satisfy the rule sets for an apply process, then the apply process discards the LCR.

See Also:

- *Oracle Streams Concepts and Administration* for detailed information about rules

About Rule-Based Transformations for Nonidentical Copies

A **rule-based transformation** is an additional configuration option that provides flexibility when a database object is not identical at the different databases. Rule-based transformations modify changes to a database object so that the changes can be applied successfully at each database. Specifically, a rule-based transformation is any modification to a message when a rule in a positive rule set evaluates to `TRUE`.

For example, suppose a table has five columns at the database where changes originated, but the shared table at a different database only has four of the five columns. When a data manipulation language (DML) operation is performed on the table at the source database, the row changes are captured and formatted as row LCRs. A rule-based transformation can delete the extra column in these row LCRs so that they can be applied successfully at the other database. If the row LCRs are not transformed, then the apply process at the other database will raise errors because the row LCRs have an extra column.

There are two types of rule-based transformations: **declarative** and **custom**. Declarative rule-based transformations include a set of common transformation scenarios for row changes resulting from DML changes (row LCRs). Custom rule-based transformations require a user-defined PL/SQL function to perform the transformation. This guide discusses only declarative rule-based transformations.

The following declarative rule-based transformations are available:

- An add column transformation adds a column to a row LCR.
- A delete column transformation deletes a column from a row LCR.
- A rename column transformation renames a column in a row LCR.
- A rename schema transformation renames the schema in a row LCR.
- A rename table transformation renames the table in a row LCR.

When you add one of these declarative rule-based transformations, you specify the rule to associate with the transformation. When the specified rule evaluates to `TRUE` for a row LCR, Oracle Streams performs the declarative transformation internally on the row LCR. Typically, rules and rule sets are created automatically when you configure your Oracle Streams replication environment.

You can configure multiple declarative rule-based transformations for a single rule, but you can only configure one custom rule-based transformation for a single rule. In addition, you can configure declarative rule-based transformations and a custom rule-based transformation for a single rule.

A transformation can occur at any stage in the Oracle Streams information flow: during capture, propagation, or apply. When a transformation occurs depends on the rule with which the transformation is associated. For example, to perform a transformation during propagation, associate the transformation with a rule in the positive rule set for a propagation.

See Also:

- *Oracle Streams Concepts and Administration* for detailed information about rule-based transformations

About Supplemental Logging

Supplemental logging is the process of adding additional column data to the redo log whenever an operation is performed (such as a row update). A capture process captures this additional information and places it in logical change records (LCRs). Apply processes that apply these LCRs might need this additional information to apply database changes properly.

See Also:

- *Oracle Streams Replication Administrator's Guide* for detailed information about supplemental logging

About Conflicts and Conflict Resolution

Conflicts occur when two different databases that are sharing data in a table modify the same row in the table at nearly the same time. When these changes are captured at one of these databases and sent to the other database, an apply process detects the conflict when it attempts to apply a row LCR to the table. By default, apply errors are placed in the error queue, where they can be resolved manually. To avoid apply errors, you must configure conflict resolution so that apply processes handle conflicts in the best way for your environment.

Oracle Database supplies prebuilt conflict handlers that provide conflict resolution when a conflict results from an `UPDATE` on a row. These handlers are called **prebuilt update conflict handlers**.

When an apply process encounters an update conflict for a row LCR that it has dequeued, it must either apply the row LCR or discard it to keep the data in the two databases consistent. The most common way to resolve update conflicts is to keep the change with the most recent time stamp and discard the older change. See "Tutorial: Configuring Latest Time Conflict Resolution for a Table" on page 4-56 for instructions about configuring latest time conflict resolution for a table.

The following topics discuss how to configure conflict resolution in a particular type of replication environment:

- Tutorial: Configuring Two-Database Replication with Local Capture Processes
- Tutorial: Configuring Hub-and-Spoke Replication with Local Capture Processes
- Tutorial: Configuring Two-Database Replication with Synchronous Captures

Note: Conflicts are not possible in a replication environment when changes to only one database are captured. In these replication environments, typically the replicas at the other databases are read-only.

See Also:

- "Tutorial: Configuring Latest Time Conflict Resolution for a Table" on page 4-56
- "Displaying the Configured Update Conflict Handlers" on page 5-23
- *Oracle Streams Replication Administrator's Guide* for detailed information about conflicts and conflict resolution

About Tags for Avoiding Change Cycling

Change cycling means sending a change back to the database where it originated. Typically, change cycling should be avoided because it can result in each database change going through endless loops to the database where it originated. Such loops can result in unintended data in the database and tax the networking and computer resources of an environment. By default, Oracle Streams is designed to avoid change cycling.

A **tag** is additional information in a change record. Each redo entry that records a database change and each logical change record (LCR) that encapsulates a database change includes a tag. The data type of the tag is `RAW`.

By default, change records have the following tag values:

- When a user or application generates database changes, the value of the tag is `NULL` for each change. This default can be changed for a particular database session.
- When an apply process generates database changes by applying them to database objects, the tag value for each change is the hexadecimal equivalent of `'00'` (double zero). This default can be changed for a particular apply process.

The tag value in an LCR depends on how the LCR was captured:

- An LCR captured by a capture process has the tag value of the redo record that was captured.
- An LCR captured by a synchronous capture has the tag value of the database session that made the change.

Rules for Oracle Streams clients can include conditions for tag values. For example, the rules for a capture process can determine whether a change in the redo log is captured based on the tag value of the redo record. In an Oracle Streams replication environment, Oracle Streams clients use tags and rules to avoid change cycling.

The following topics discuss how change cycling is avoided in a particular type of replication environment:

- Tutorial: Configuring Two-Database Replication with Local Capture Processes
- Tutorial: Configuring Hub-and-Spoke Replication with Local Capture Processes
- Tutorial: Configuring Two-Database Replication with Synchronous Captures

Note:

- Change cycling is not possible in a replication environment when changes to only one database are captured.
- You can also use tags to avoid replicating the changes made by a particular session. Use the `DBMS_STREAMS.SET_TAG` procedure to set the tag for a session. See "Correcting Apply Errors in Database Objects" on page 5-31 for an example.

See Also:

- "Managing Apply Errors" on page 5-30
- *Oracle Streams Replication Administrator's Guide* for detailed information about tags

About the Common Types of Oracle Streams Replication Environments

Oracle Streams enables you to configure many different types of custom replication environments. However, three types of replication environments are the most common: two-database, hub-and-spoke, and n-way.

The following topics describe these common types of replication environments and help you decide which one is best for you:

- About Two-Database Replication Environments
- About Hub-And-Spoke Replication Environments
- About N-Way Replication Environments

About Two-Database Replication Environments

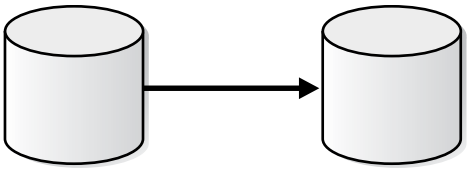
A **two-database replication environment** is one in which only two databases share the replicated database objects. The changes made to replicated database objects at one database are captured and sent directly to the other database, where they are applied. In a two-database replication environment, only one database might allow changes to the database objects, or both databases might allow changes to them.

If only one database allows changes to the replicated database objects, then the other database contains read-only replicas of these database objects. This is a **one-way replication environment** and typically has the following basic components:

- The first database has a capture process or synchronous capture to capture changes to the replicated database objects.
- The first database has a propagation that sends the captured changes to the other database.
- The second database has an apply process to apply changes from the first database.
- For the best performance, each capture process and apply process has its own queue.

Figure 4–6 shows a two-database replication environment configured for one-way replication.

Figure 4–6 One-Way Replication in a Two-Database Replication Environment

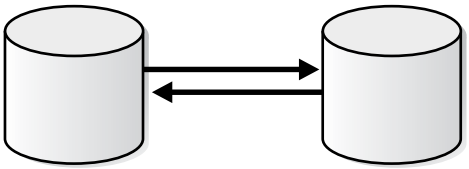


In a two-database replication environment, both databases can allow changes to the replicated database objects. In this case, both databases capture changes to these database objects and send the changes to the other database, where they are applied. This is a **bi-directional replication environment** and typically has the following basic components:

- Each database has a capture process or synchronous capture to capture changes to the replicated database objects.
- Each database has a propagation that sends the captured changes to the other database.
- Each database has an apply process to apply changes from the other database.
- For the best performance, each capture process and apply process has its own queue.

Figure 4–7 show a two-database replication environment configured for bi-directional replication.

Figure 4–7 Bi-Directional Replication in a Two-Database Replication Environment



Typically, in a bi-directional replication environment, you should configure conflict resolution to keep the replicated database objects synchronized. You can configure a two-database replication environment using the configuration procedures in the `DBMS_STREAMS_ADM` package. See "About the Oracle Streams Replication Configuration Procedures" on page 4-16.

See Also:

- "Tutorial: Configuring Two-Database Replication with Local Capture Processes" on page 4-25
- "Tutorial: Configuring Two-Database Replication with a Downstream Capture Process" on page 4-30
- "Tutorial: Configuring Two-Database Replication with Synchronous Captures" on page 4-46
- "About Conflicts and Conflict Resolution" on page 4-9

About Hub-And-Spoke Replication Environments

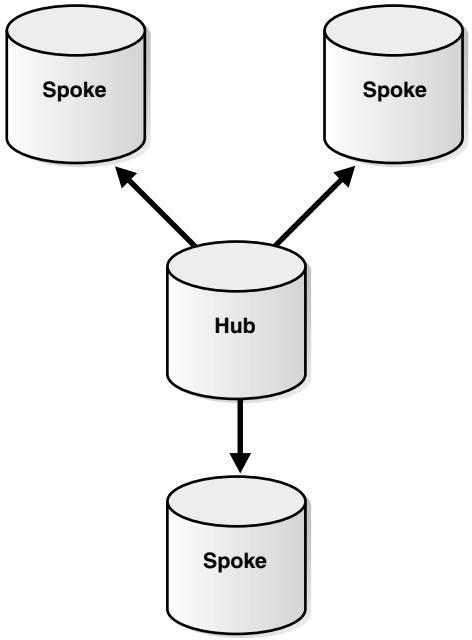
A **hub-and-spoke replication environment** is one in which a central database, or hub, communicates with secondary databases, or spokes. The spokes do not communicate directly with each other. In a hub-and-spoke replication environment, the spokes might or might not allow changes to the replicated database objects.

If the spokes do not allow changes, then they contain read-only replicas of the database objects at the hub. This type of hub-and-spoke replication environment typically has the following basic components:

- The hub has a capture process or synchronous capture to capture changes to the replicated database objects.
- The hub has propagations that send the captured changes to each of the spokes.
- Each spoke has an apply process to apply changes from the hub.
- For the best performance, each capture process and apply process has its own queue.

Figure 4–8 shows a hub-and-spoke replication environment with read-only spokes.

Figure 4–8 Hub-and-Spoke Replication Environment with Read-Only Spokes



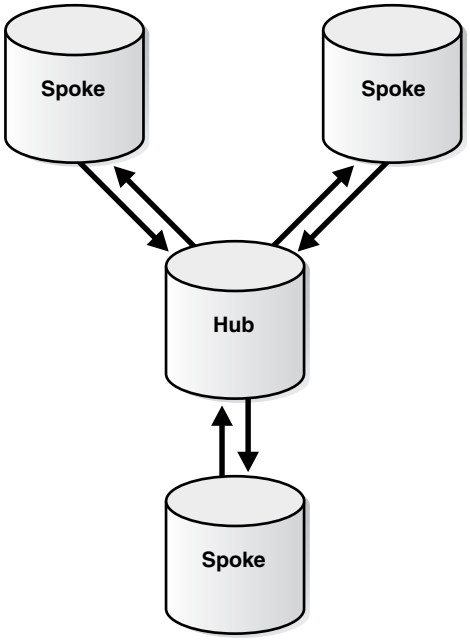
If the spokes allow changes to the database objects, then typically the changes are captured and sent back to the hub, and the hub replicates the changes with the other spokes. This type of hub-and-spoke replication environment typically has the following basic components:

- The hub has a capture process or synchronous capture to capture changes to the replicated database objects.
- The hub has propagations that send the captured changes to each of the spokes.
- Each spoke has a capture process or synchronous capture to capture changes to the replicated database objects.
- Each spoke has a propagation that sends changes made at the spoke back to the hub.

- Each spoke has an apply process to apply changes from the hub and from the other spokes.
- The hub has a separate apply process to apply changes from each spoke. A different apply process must apply changes from each spoke.
- For the best performance, each capture process and apply process has its own queue.

Figure 4–9 shows a hub-and-spoke replication environment with read/write spokes.

Figure 4–9 Hub-and-Spoke Replication Environment with Read/Write Spokes



Typically, in a hub-and-spoke replication environment that allows changes at spoke databases, you should configure conflict resolution to keep the replicated database objects synchronized. Some hub-and-spoke replication environments allow changes to the replicated database objects at some spokes but not at others.

You can configure a hub-and-spoke replication environment using the configuration procedures in the `DBMS_STREAMS_ADM` package. See "About the Oracle Streams Replication Configuration Procedures" on page 4-16.

See "When to Replicate Data with Oracle Streams" on page 1-4 for information about when hub-and-spoke replication is useful.

See Also:

- "Tutorial: Configuring Hub-and-Spoke Replication with Local Capture Processes" on page 4-40
- "About Conflicts and Conflict Resolution" on page 4-9

About N-Way Replication Environments

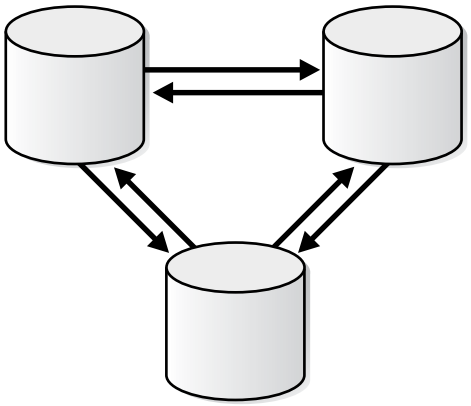
An **n-way replication environment** is one in which each database communicates directly with each other database in the environment. The changes made to replicated database objects at one database are captured and sent directly to each of the other databases in the environment, where they are applied.

An n-way replication environment typically has the following basic components:

- Each database has one or more capture processes or synchronous captures to capture changes to the replicated database objects.
- Each database has propagations that send the captured changes to each of the other databases.
- Each database has apply processes that apply changes from each of the other databases. A different apply process must apply changes from each source database.
- For the best performance, each capture process and apply process has its own queue.

Figure 4–10 shows an n-way replication environment.

Figure 4–10 N-Way Replication Environment



You can configure an n-way replication environment by using the following Oracle-supplied packages:

- `DBMS_STREAMS_ADM` can be used to perform most of the configuration actions, including setting up queues, creating capture processes or synchronous captures, creating propagations, creating apply processes, and configuring rules and rule sets for the replication environment.
- `DBMS_CAPTURE_ADM` can be used to start any capture processes you configured in the replication environment.
- `DBMS_APPLY_ADM` can be used to configure apply processes, configure conflict resolution, and start apply processes, as well as other configuration tasks.

See "When to Replicate Data with Oracle Streams" on page 1-4 for information about when n-way replication is useful.

Typically, in an n-way replication environment, you should configure conflict resolution to keep the replicated database objects synchronized.

Configuring an n-way replication environment is beyond the scope of this guide. See *Oracle Streams Replication Administrator's Guide* for a detailed example that configures an n-way replication environment.

About the Oracle Streams Replication Configuration Procedures

The easiest way to configure an Oracle Streams replication environment is by running one of the following configuration procedures in the `DBMS_STREAMS_ADM` package:

- The `MAINTAIN_GLOBAL` procedure configures an Oracle Streams environment that replicates changes at the database level between two databases.
- The `MAINTAIN_SCHEMAS` procedure configures an Oracle Streams environment that replicates changes to specified schemas between two databases.
- The `MAINTAIN_SIMPLE_TTS` procedure clones a simple tablespace from a source database at a destination database and uses Oracle Streams to maintain this tablespace at both databases.
- The `MAINTAIN_TABLES` procedure configures an Oracle Streams environment that replicates changes to specified tables between two databases.
- The `MAINTAIN_TTS` procedure clones a set of tablespaces from a source database at a destination database and uses Oracle Streams to maintain these tablespaces at both databases.

These procedures configure two databases at a time, and they require you to specify one database as the source database and the other database as the destination database. They can be used to configure a replication environment with more than two databases by running them multiple times.

Table 4–1 describes the required parameters for these procedures.

Table 4–1 Required Parameters for the Oracle Streams Replication Configuration Procedures

Parameter	Procedure	Description
`source_directory_object`	All procedures	The directory object for the directory on the computer system running the source database into which the generated Data Pump export dump file is placed.
`destination_directory_object`	All procedures	The directory object for the directory on the computer system running the destination database into which the generated Data Pump export dump file is transferred. The dump file is used to instantiate the replicated database objects at the destination database.
`source_database`	All procedures	The global name of the source database. The specified database must contain the database objects to be replicated.
`destination_database`	All procedures	The global name of the destination database. The database objects to be replicated are optional at the destination database. If they do not exist at the destination database, then they are instantiated by Data Pump export/import. If the local database is not the destination database, then a database link from the local database to the destination database, with the same name as the global name of the destination database, must exist and must be accessible to the user who runs the procedure.

Table 4–1 (Cont.) Required Parameters for the Oracle Streams Replication Configuration Procedures

Parameter	Procedure	Description
schema_names	MAINTAIN_SCHEMAS only	The schemas to be configured for replication.
tablespace_name	MAINTAIN_SIMPLE_TTS only	The tablespace to be configured for replication.
table_names	MAINTAIN_TABLES only	The tables to be configured for replication.
tablespace_names	MAINTAIN_TTS only	The tablespaces to be configured for replication.

In addition, each of these procedures has several optional parameters. The bi_directional parameter is an important optional parameter. If you want changes to the replicated database objects to be captured at each database and sent to the other database, then the bi_directional parameter must be set to TRUE. The default setting for this parameter is FALSE. When the bi_directional parameter is set to FALSE, the procedures configure a one-way replication environment, where the changes made at the destination database are not captured.

These procedures perform several tasks to configure an Oracle Streams replication environment. These tasks include:

- Configure supplemental logging for the replicated database objects at the source database. See "About Supplemental Logging" on page 4-9.
- If the bi_directional parameter is set to TRUE, then configure supplemental logging for the replicated database objects at the destination database.
- Instantiate the database objects at the destination database. If the database objects do not exist at the destination database, then the procedures use Data Pump export/import to instantiate them at the destination database.
- Configure a capture process to capture changes to the replicated database objects at the source database. This capture process can be a local capture process or a downstream capture process. If the procedure is run at the database specified in the source_database parameter, then the procedure configures a local capture process on this database. If the procedure is run at a database other than the database specified in the source_database parameter, then the procedure configures a downstream capture process on the database that runs the procedure. See "About Change Capture with a Capture Process" on page 4-3.
- If the bi_directional parameter is set to TRUE, then configure a capture process to capture changes to the replicated database objects at the destination database. This capture process must be a local capture process.
- Configure one or more queues at each database to store captured changes.
- Configure a propagation to send changes made to the database objects at the source database to the destination database. See "About Change Propagation Between Databases" on page 4-5.
- If the bi_directional parameter is set to TRUE, then configure a propagation to send changes made to the database objects at the destination database to the source database
- Configure an apply process at the destination database to apply changes from the source database. See "About Change Apply" on page 4-6.

- If the `bi_directional` parameter is set to `TRUE`, then configure an apply process at the source database to apply changes from the destination database.
- Configure rule sets and rules for each capture process, propagation, and apply process. The rules instruct the Oracle Streams clients to replicate changes to the specified database objects. See "About Rules for Controlling the Behavior of Capture, Propagation, and Apply" on page 4-7.
- Set the instantiation SCN for the replicated database objects at the destination database. See "About Change Apply" on page 4-6.
- If the `bi_directional` parameter is set to `TRUE`, then set the instantiation SCN for the replicated database objects at the source database.

> **Tip:** To view all of the actions performed by one of these procedures in detail, use the procedure to generate a script, and view the script in a text editor. You can use the `perform_actions`, `script_name`, and `script_directory_object` parameters to generate a script.

These procedures always configure tags for a hub-and-spoke replication environment. The following are important considerations about these procedures and tags:

- If you are configuring a two-database replication environment, then you can use these procedures to configure it. These procedures configure tags in a two-database environment to avoid change cycling. If you plan to expand the replication environment beyond two databases in the future, then it is important to understand how the tags are configured. If the expanded database environment is not a hub-and-spoke environment, then you might need to modify the tags to avoid change cycling.
- If you are configuring a replication environment that involves three or more databases, then these procedures can only be used to configure a hub-and-spoke replication environment. These procedures configure tags in a hub-and-spoke environment to avoid change cycling.
- If you are configuring an n-way replication environment, then do not use these procedures to configure it. Change cycling might result if you do so.

See "About Tags for Avoiding Change Cycling" on page 4-10 and "About the Common Types of Oracle Streams Replication Environments" on page 4-11 for more information about tags and the different types of replication environments.

Note: Currently, these configuration procedures configure only capture processes to capture changes. You cannot use these procedures to configure a replication environment that uses synchronous captures. You can configure a synchronous capture using the `ADD_TABLE_RULES` and `ADD_SUBSET_RULES` procedures in the `DBMS_STREAMS_ADM` package.

See Also:

- "Tutorial: Configuring Two-Database Replication with Local Capture Processes" on page 4-25
- "Tutorial: Configuring Two-Database Replication with a Downstream Capture Process" on page 4-30
- "Tutorial: Configuring Hub-and-Spoke Replication with Local Capture Processes" on page 4-40
- *Oracle Database PL/SQL Packages and Types Reference* for more information about the procedures in the `DBMS_STREAMS_ADM` package

About Key Oracle Streams Supplied PL/SQL Packages and Data Dictionary Views

In addition to Oracle Enterprise Manager, you can use several supplied PL/SQL packages to configure and administer an Oracle Streams replication environment. You can also use several data dictionary views to monitor an Oracle Streams replication environment.

The following topics describe the key Oracle Streams supplied PL/SQL packages and data dictionary views:

- About Key Oracle Streams Supplied PL/SQL Packages
- About Key Oracle Streams Data Dictionary Views

About Key Oracle Streams Supplied PL/SQL Packages

Table 4–2 describes the supplied PL/SQL packages that are important for configuring and administering an Oracle Streams replication environment.

Table 4–2 Key Oracle Streams Supplied PL/SQL Packages

Package	Description
`DBMS_STREAMS_ADM`	This package provides an easy way to complete common tasks in an Oracle Streams environment. This package contains procedures that enable you to configure and maintain an Oracle Streams replication environment. This package also provides an administrative interface for adding and removing simple rules for capture processes, synchronous captures, propagations, and apply processes at the table, schema, and database level. This package also contains procedures for creating queues and for managing Oracle Streams metadata, such as data dictionary information.
`DBMS_CAPTURE_ADM`	This package provides an administrative interface for starting, stopping, and configuring a capture process. It also provides an administrative interface for configuring a synchronous capture. This package also provides administrative procedures that prepare database objects at the source database for instantiation at a destination database.
`DBMS_PROPAGATION_ADM`	This package provides an administrative interface for starting, stopping, and configuring a propagation.

Table 4–2 (Cont.) Key Oracle Streams Supplied PL/SQL Packages

Package	Description
DBMS_APPLY_ADM	This package provides an administrative interface for starting, stopping, and configuring an apply process. This package also includes subprograms for configuring conflict detection and resolution and for managing apply errors. This package also includes procedures that enable you to configure apply handlers. This package also provides administrative procedures that set the instantiation SCN for objects at a destination database.
DBMS_STREAMS_AUTH	This package provides subprograms for granting privileges to Oracle Streams administrators and revoking privileges from Oracle Streams administrators.
DBMS_STREAMS_ADVISOR_ADM	This package provides an interface for gathering information about an Oracle Streams environment and advising database administrators based on the information gathered. This package is part of the Oracle Streams Performance Advisor.

See Also:

- *Oracle Streams Concepts and Administration* for more information about these Oracle Streams packages and other Oracle Streams packages
- "About Hub-And-Spoke Replication Environments" on page 4-13

About Key Oracle Streams Data Dictionary Views

Table 4–3 describes the data dictionary views that are important for monitoring an Oracle Streams replication environment.

Table 4–3 Key Oracle Streams Data Dictionary Views

Data Dictionary View	Description
ALL_APPLY DBA_APPLY	Displays information about apply processes.
ALL_APPLY_CONFLICT_COLUMNS DBA_APPLY_CONFLICT_COLUMNS	Displays information about conflict handlers.
ALL_APPLY_ERROR DBA_APPLY_ERROR	Displays information about the error transactions generated by apply processes.
ALL_CAPTURE DBA_CAPTURE	Displays information about capture processes.
ALL_PROPAGATION DBA_PROPAGATION	Displays information about propagations.
ALL_STREAMS_COLUMNS DBA_STREAMS_COLUMNS	Displays information about the columns that are not supported by synchronous captures and apply processes.
ALL_STREAMS_RULES DBA_STREAMS_RULES	Displays information about the rules used by capture processes, synchronous captures, propagations, and apply processes.
ALL_STREAMS_UNSUPPORTED DBA_STREAMS_UNSUPPORTED	Displays information about the database objects that are not supported by capture processes.

Table 4–3 (Cont.) Key Oracle Streams Data Dictionary Views

Data Dictionary View	Description
ALL_SYNC_CAPTURE DBA_SYNC_CAPTURE	Displays information about synchronous captures.
V$BUFFERED_QUEUES	Displays information about buffered queues and the messages in the buffered queues.
V$PROPAGATION_RECEIVER	Displays information about the messages received into buffered queues by propagations.
V$PROPAGATION_SENDER	Displays information about the messages sent from buffered queues by propagations.
V$STREAMS_APPLY_COORDINATOR	Displays information about apply process coordinators for enabled apply processes.
V$STREAMS_APPLY_READER	Displays information about apply process reader servers for enabled apply processes.
V$STREAMS_APPLY_SERVER	Displays information about apply process apply servers for enabled apply processes.
V$STREAMS_CAPTURE	Displays information about enabled capture processes.
V$STREAMS_TRANSACTION	Displays information about transactions that are being processed by capture processes or apply processes. This view can be used to identify long running transactions and to determine how many logical change records (LCRs) are being processed in each transaction. This view only contains information about LCRs that were captured by a capture process.

See Also:

- *Oracle Streams Concepts and Administration* for more information about these Oracle Streams data dictionary views and other Oracle Streams data dictionary views
- "Monitoring an Oracle Streams Replication Environment" on page 5-9

Preparing for Oracle Streams Replication

Before configuring Oracle Streams replication, prepare the databases that will participate in the replication environment.

To prepare for Oracle Streams replication:

1. Set initialization parameters properly before you configure a replication environment with Oracle Streams:
   - **Global Names:** Set the GLOBAL_NAMES initialization parameter to TRUE at each database that will participate in the Oracle Streams replication environment. See "Setting the GLOBAL_NAMES Initialization Parameter to TRUE" on page 2-1.
   - **Compatibility:** To use the latest features of Oracle Streams, it is best to set the COMPATIBLE initialization parameter as high as you can. If possible, then set this parameter to 11.1.0 or higher.

- **System Global Area (SGA) and the Oracle Streams pool:** Ensure that the Oracle Streams pool is large enough to accommodate the Oracle Streams components created for the replication environment. The Oracle Streams pool is part of the System Global Area (SGA). You can manage the Oracle Streams pool by setting the `MEMORY_TARGET` initialization parameter (Automatic Memory Management), the `SGA_TARGET` initialization parameter (Automatic Shared Memory Management), or the `STREAMS_POOL_SIZE` initialization parameter. See *Oracle Streams Concepts and Administration* for more information about the Oracle Streams pool.

  The memory requirements for Oracle Streams components are:

  - Each queue requires at least 10 MB of memory.
  - Each capture process parallelism requires at least 10 MB of memory. The `parallelism` capture process parameter controls the number of processes used by the capture process to capture changes. You might be able to improve capture process performance by adjusting capture process parallelism.
  - Each propagation requires at least 1 MB of memory.
  - Each apply process parallelism requires at least 1 MB of memory. The `parallelism` apply process parameter controls the number of processes used by the apply process to apply changes. You might be able to improve apply process performance by adjusting apply process parallelism.

- **Processes and Sessions:** Oracle Streams capture processes, propagations, and apply processes use processes that run in the background. You might need to increase the value of the `PROCESSES` and `SESSIONS` initialization parameters to accommodate these processes.

2. Review the best practices for Oracle Streams replication environments and follow the best practices when you configure the environment. See *Oracle Streams Replication Administrator's Guide* for information about best practices.

   Following the best practices ensures that your environment performs optimally and avoids problems. The configuration procedures in the `DBMS_STREAMS_ADM` package follow the best practices automatically. However, if you plan to configure an Oracle Streams replication environment without using a configuration procedure, then learn about the best practices and follow them whenever possible.

   The following are some of the important best practices to follow during Oracle Streams configuration:

   - Configure a separate tablespace for the Oracle Streams administrator. The instructions in "Tutorial: Creating an Oracle Streams Administrator" follow this best practice.
   - Use separate queues for capture processes, synchronous captures, and apply processes. For the best performance, these components typically should not share a queue.
   - Use queue-to-queue propagations.

   After the Oracle Streams environment is configured, the following are some of the important best practices to follow for operation of the Oracle Streams environment:

   - Monitor performance and make adjustments when necessary.
   - Monitor queues for size.

- Follow the Oracle Streams best practices for backups.
- Check the alert log for Oracle Streams information.
- Set capture process parallelism for best performance.
- Set apply process parallelism for best performance.
- Check for apply errors and manage them if they occur.

See *Oracle Streams Replication Administrator's Guide* for detailed information about these best practices, and for information about other Oracle Streams best practices.

See Also:

- "About Oracle Streams Replication" on page 4-1
- "Administering an Oracle Streams Replication Environment" on page 5-1
- *Oracle Database 2 Day DBA*
- *Oracle Streams Concepts and Administration* for information about initialization parameters that are important in an Oracle Streams environment

Configuring Oracle Streams Replication: Examples

This section uses examples to show you how to configure Oracle Streams replication environments. The examples configure the most common types of Oracle Streams replication environments.

The following are descriptions of the examples:

- Tutorial: Configuring Two-Database Replication with Local Capture Processes

  This example configures an Oracle Streams replication environment that replicates data manipulation language (DML) changes to all of the tables in the `hr` schema at two databases. The configuration uses capture processes to capture the changes to the replicated database objects.

  The example shows you how to configure the environment for one-way replication or bi-directional replication. With one-way replication, only one database allows changes to the database objects. With bi-directional replication, both databases allow changes to the database objects.

  Use this configuration when you want to configure a relatively simple replication environment that only involves two databases. You might configure a one-way replication environment if you want to use a second database for reporting or data analysis. You might configure a two-way replication environment if the data must be read/write at both databases. For example, two-way replication can be used to improve scalability or increase the availability of the data.

  See "About Two-Database Replication Environments" on page 4-11 for more information about two-database replication environments.

- Tutorial: Configuring Two-Database Replication with a Downstream Capture Process

  This example configures an Oracle Streams replication environment that replicates data manipulation language (DML) changes to all of the tables in the `hr` schema at two databases. The example shows you how to configure the environment for one-way replication, where changes to the replicated database objects are only allowed at one source database. The configuration uses a downstream capture

process at a destination database to capture the changes that were made to these database objects at the source database. The database objects are read-only at the destination database.

Use this configuration when you want to offload reporting or data analysis from a primary database to another database. In this example, the capture process runs on the second database to reduce the load on the primary database.

See "About Two-Database Replication Environments" on page 4-11 for more information about two-database replication environments.

- Tutorial: Configuring Hub-and-Spoke Replication with Local Capture Processes

  This example configures an Oracle Streams replication environment that replicates data manipulation language (DML) changes to all of the tables in the `hr` schema at three databases. This example configures a hub-and-spoke replication environment with one hub and two spokes. A hub-and-spoke replication environment is one in which a central database, or hub, communicates with secondary databases, or spokes. In this example, the spoke databases allow changes to the replicated database objects. The configuration uses capture processes to capture the changes to the replicated database objects.

  Use this configuration when you have a central database that must replicate data with several secondary databases. For example, a business with a central database at headquarters and several secondary databases at sales offices might choose this configuration.

  See "About Hub-And-Spoke Replication Environments" on page 4-13 for more information about hub-and-spoke replication environments.

- Tutorial: Configuring Two-Database Replication with Synchronous Captures

  This example configures an Oracle Streams replication environment that replicates data manipulation language (DML) changes to the `employees` and `departments` tables in the `hr` schema at two databases. This example configures bi-directional replication between the two databases. So, changes to the databases objects are allowed at each database. The configuration uses synchronous captures to capture the changes to the replicated database objects.

  Use this configuration when you want to configure a relatively simple replication environment that only involves two databases and a small number of tables. You might also choose to use synchronous captures if you have Oracle Database 11*g* Standard Edition. To use capture processes, you must have Oracle Database 11*g* Enterprise Edition.

  See "About Two-Database Replication Environments" on page 4-11 for more information about two-database replication environments.

This section also includes an example that configures conflict resolution for a table. See "Tutorial: Configuring Latest Time Conflict Resolution for a Table" on page 4-56. Use conflict resolution in a replication environment that allows more than one database to perform DML changes on replicated tables. This example configures latest time conflict resolution. Therefore, when a conflict occurs for a row change to a table, the most recent change is retained, and the older change is discarded. See "About Conflicts and Conflict Resolution" on page 4-9 for more information about conflict resolution.

Note: Another common Oracle Streams replication environment is the n-way environment. See "About N-Way Replication Environments" on page 4-14.

See Also:

- "About the Common Types of Oracle Streams Replication Environments" on page 4-11

Tutorial: Configuring Two-Database Replication with Local Capture Processes

This example configures an Oracle Streams replication environment that replicates data manipulation language (DML) changes to all of the tables in the `hr` schema. This example configures a two-database replication environment with local capture processes to capture changes. This example uses the global database names `db1.example.com` and `db2.example.com`. However, you can substitute databases in your environment to complete the example. See "About Two-Database Replication Environments" on page 4-11 for more information about two-database replication environments.

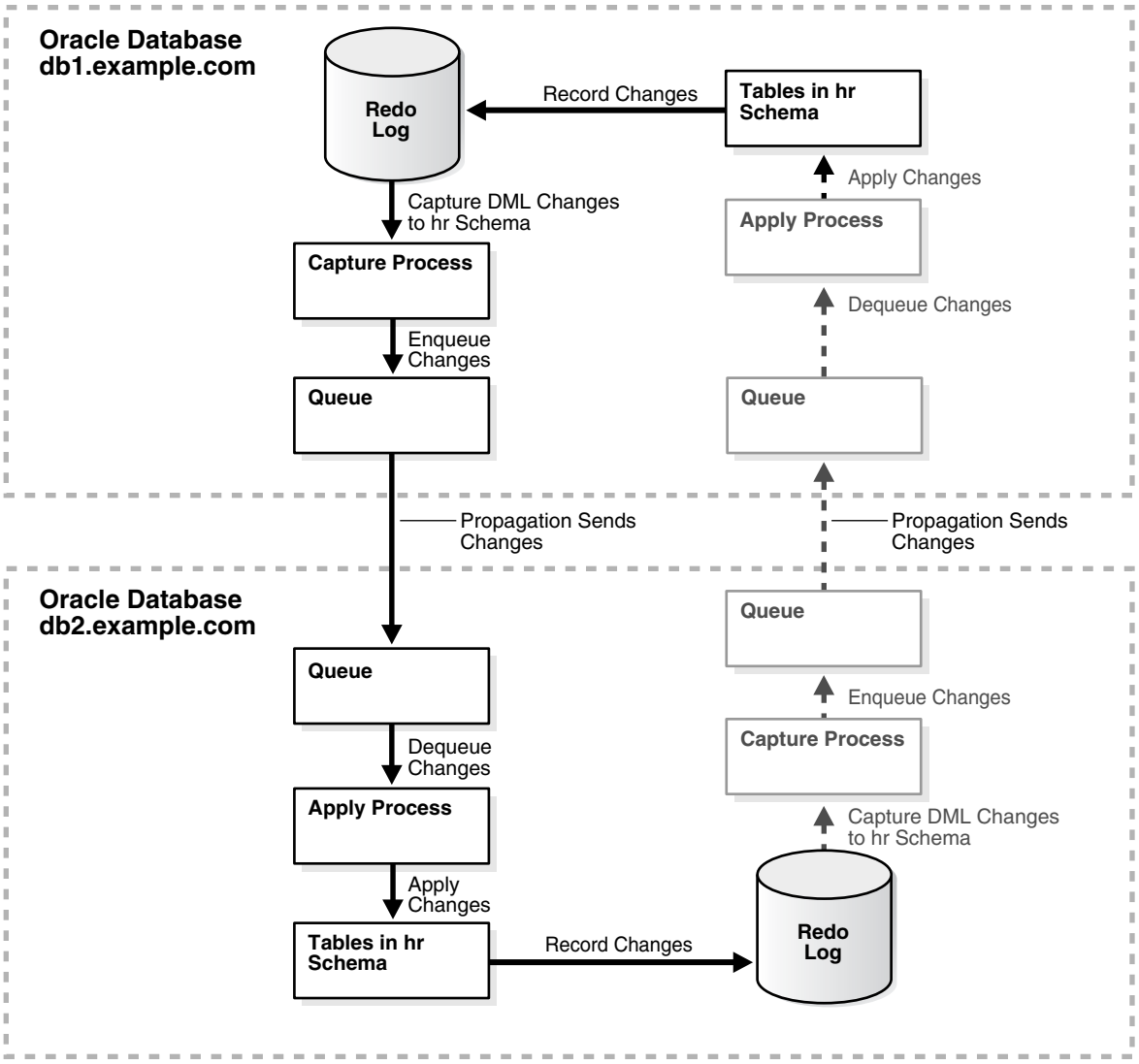
This example uses the `MAINTAIN_SCHEMAS` procedure in the `DBMS_STREAMS_ADM` package to configure the two-database replication environment. This procedure is the fastest and simplest way to configure an Oracle Streams environment that replicates one or more schemas. In addition, the procedure follows established best practices for Oracle Streams replication environments.

The database objects being configured for replication might or might not exist at the destination database when you run the `MAINTAIN_SCHEMAS` procedure. If the database objects do not exist at the destination database, then the `MAINTAIN_SCHEMAS` procedure instantiates them at the destination database using a Data Pump export/import. During instantiation, the instantiation SCN is set for these database objects. If the database objects already exist at the destination database, then the `MAINTAIN_SCHEMAS` procedure retains the existing database objects and sets the instantiation SCN for them. In this example, the `hr` schema exists at both the `db1.example.com` database and the `db2.example.com` database before the `MAINTAIN_SCHEMAS` procedure is run.

This example provides instructions for configuring either one-way or bi-directional replication. To configure bi-directional replication, you must complete additional steps and set the `bi_directional` parameter to `TRUE` when you run the configuration procedure.

Figure 4–11 provides an overview of the environment created in this example. The additional components required for bi-directional replication are shown in gray, and their actions are indicated by dashed lines.

Figure 4–11 Two-Database Replication Environment with Local Capture Processes



To configure this two-database replication environment:

1. Complete the following tasks to prepare for the two-database replication environment:

   a. Configure network connectivity so that the `db1.example.com` database can communicate with the `db2.example.com` database.

      See *Oracle Database 2 Day DBA* for information about configuring network connectivity between databases.

   b. Configure an Oracle Streams administrator at each database that will participate in the replication environment. See "Tutorial: Creating an Oracle Streams Administrator" on page 2-2 for instructions. This example assumes that the Oracle Streams administrator is `strmadmin`.

   c. Create a database link from the `db1.example.com` database to the `db2.example.com` database.

      The database link should be created in the Oracle Streams administrator's schema. Also, the database link should connect to the Oracle Streams administrator at the other database. Both the name and the service name of the

database link must be db2.example.com. See "Tutorial: Creating a Database Link" on page 2-8 for instructions.

d. Configure the db1.example.com database to run in ARCHIVELOG mode. For a capture process to capture changes generated at a source database, the source database must be running in ARCHIVELOG mode. See *Oracle Database Administrator's Guide* for information about configuring a database to run in ARCHIVELOG mode.

2. To configure a bi-directional replication environment, complete the following steps. If you are configuring a one-way replication environment, then these steps are not required, and you can move on to Step 3.

a. Create a database link from the db2.example.com database to the db1.example.com database.

The database link should be created in the Oracle Streams administrator's schema. Also, the database link should connect to the Oracle Streams administrator at the other database. Both the name and the service name of the database link must be db1.example.com. See "Tutorial: Creating a Database Link" on page 2-8 for instructions.

b. Configure the db2.example.com database to run in ARCHIVELOG mode. For a capture process to capture changes generated at a source database, the source database must be running in ARCHIVELOG mode. See *Oracle Database Administrator's Guide* for information about configuring a database to run in ARCHIVELOG mode.

3. Set initialization parameters properly at each database that will participate in the Oracle Streams replication environment. See "Preparing for Oracle Streams Replication" on page 4-21 for instructions.

4. On a command line, open SQL*Plus and connect to the db2.example.com database as the Oracle Streams administrator.

See *Oracle Database 2 Day DBA* for more information about starting SQL*Plus.

5. Create a directory object to hold files that will be generated by the MAINTAIN_SCHEMAS procedure, including the Data Pump export dump file used for instantiation. The directory object can point to any accessible directory on the computer system. For example, the following statement creates a directory object named db2_dir that points to the /usr/db2_log_files directory:

```
CREATE DIRECTORY db2_dir AS '/usr/db2_log_files';
```

6. In SQL*Plus, connect to the db1.example.com database as the Oracle Streams administrator.

See *Oracle Database Administrator's Guide* for information about connecting to a database in SQL*Plus.

7. Create a directory object to hold files that will be generated by the MAINTAIN_SCHEMAS procedure, including the Data Pump export dump file used for instantiation. The directory object can point to any accessible directory on the computer system. For example, the following statement creates a directory object named db1_dir that points to the /usr/db1_log_files directory:

```
CREATE DIRECTORY db1_dir AS '/usr/db1_log_files';
```

8. Run the MAINTAIN_SCHEMAS procedure to configure replication of the hr schema between the db1.example.com database and the db2.example.com database.

   Ensure that the bi_directional parameter is set properly for the replication environment that you are configuring. Either set this parameter to FALSE for one-way replication, or set it to TRUE for bi-directional replication.

```
BEGIN
  DBMS_STREAMS_ADM.MAINTAIN_SCHEMAS(
    schema_names                 => 'hr',
    source_directory_object      => 'db1_dir',
    destination_directory_object => 'db2_dir',
    source_database              => 'db1.example.com',
    destination_database         => 'db2.example.com',
    bi_directional               => FALSE); -- Set to TRUE for bi-directional
END;
/
```

   The MAINTAIN_SCHEMAS procedure can take some time to run because it is performing many configuration tasks. Do not allow data manipulation language (DML) or data definition language (DDL) changes to the replicated database objects at the destination database while the procedure is running. See "About the Oracle Streams Replication Configuration Procedures" on page 4-16.

   When a configuration procedure is run, information about its progress is recorded in the following data dictionary views: DBA_RECOVERABLE_SCRIPT, DBA_RECOVERABLE_SCRIPT_PARAMS, DBA_RECOVERABLE_SCRIPT_BLOCKS, and DBA_RECOVERABLE_SCRIPT_ERRORS. If the procedure stops because it encounters an error, then see *Oracle Streams Replication Administrator's Guide* for instructions about using the RECOVER_OPERATION procedure in the DBMS_STREAMS_ADM package to recover from these errors.

9. If you configured bi-directional replication, then configure latest time conflict resolution for all of the tables in the hr schema at both databases. This schema includes the countries, departments, employees, jobs, job_history, locations, and regions tables. See "Tutorial: Configuring Latest Time Conflict Resolution for a Table" on page 4-56 for instructions.

When you complete the example, a two-database replication environment with the following characteristics is configured:

- At db1.example.com, supplemental logging is configured for the tables in the hr schema.
- The db1.example.com database has the following components:
  - A capture process with a system-generated name. The capture process captures DML changes to the hr schema.
  - A queue with a system-generated name. This queue is for the capture process at the database.
  - A propagation with a system-generated name that sends changes from the queue at the db1.example.com database to the queue at the db2.example.com database.
- The db2.example.com database has the following components:
  - A queue with a system-generated name that receives the changes sent from the db1.example.com database. This queue is for the apply process at the local database.

  - An apply process with a system-generated name. The apply process dequeues changes from its queue and applies them to the `hr` schema.

- If the replication environment is bi-directional, then the following are also configured:
  - At `db2.example.com`, supplemental logging for the tables in the `hr` schema.
  - At `db2.example.com`, a capture process with a system-generated name. The capture process captures DML changes to the `hr` schema.
  - At `db2.example.com`, a queue with a system-generated name. This queue is for the capture process at the database.
  - At `db1.example.com`, a queue with a system-generated name that receives the changes sent from the `db2.example.com` database. This queue is for the apply process at the local database.
  - At `db1.example.com`, an apply process with a system-generated name. The apply process dequeues changes from its queue and applies them to the `hr` schema.
- If the replication environment is bi-directional, then tags are used to avoid change cycling in the following way:
  - Each apply process uses an apply tag, and redo records for changes applied by the apply process include the tag. Each apply process uses an apply tag that is unique in the replication environment.
  - Each capture process captures all of the changes to the replicated database objects, regardless of the tag in the redo record. Therefore, each capture process captures the changes applied by the apply processes on its source database.
  - Each propagation sends all changes made to the replicated database objects to the other database in the replication environment, except for changes that originated at the other database. The propagation rules instruct the propagation to discard these changes.

  See "About Tags for Avoiding Change Cycling" on page 4-10 for more information about how the replication environment avoids change cycling. If you configured one-way replication, then change cycling is not possible because changes are only captured in a single location.

To check the Oracle Streams replication configuration:

1. At the `db1.example.com` database, ensure that the capture process is enabled and that the capture type is local. To do so, follow the instructions in "Viewing Information About a Capture Process" on page 5-12, and check the Status and Capture Type fields on the Capture subpage.
2. At the `db1.example.com` database, ensure that the propagation is enabled. To do so, follow the instructions in "Viewing Information About a Propagation" on page 5-16, and check the Status field on the Propagation subpage.
3. At the `db2.example.com` database, ensure that the apply process is enabled. To do so, follow the instructions in "Viewing Information About an Apply Process" on page 5-20, and check the Status field on the Apply subpage.
4. If you configured bi-directional replication, then complete the following steps:
   a. At the `db2.example.com` database, ensure that the capture process is enabled and that the capture type is local.

b. At the db2.example.com database, ensure that the propagation is enabled.

c. At the db1.example.com database, ensure that the apply process is enabled.

To replicate changes:

1. At a database that captures changes to the hr schema, make DML changes to any table in the hr schema. In this example, the db1.example.com database captures changes to the hr schema, and, if you configured bi-directional replication, then db2.example.com also captures changes to the hr schema.
2. After some time has passed to allow for replication of the changes, use SQL*Plus to query the modified table at the other database to view the DML changes.

Note: The configuration procedures in the DBMS_STREAMS_ADM package do not configure the replicated tables to be read only at the destination databases. If one-way replication is configured and they should be read only, then configure privileges at the destination databases accordingly. However, the apply user for the apply process must be able to make DML changes to the replicated database objects. In this example, the apply user is the Oracle Streams administrator. See *Oracle Database Security Guide* for information about configuring privileges.

See Also:

- "About Hub-And-Spoke Replication Environments" on page 4-13
- "When to Replicate Data with Oracle Streams" on page 1-4
- Chapter 5, "Administering an Oracle Streams Replication Environment"
- *Oracle Streams Replication Administrator's Guide* for more detailed instructions about using the MAINTAIN_SCHEMAS procedure
- *Oracle Database PL/SQL Packages and Types Reference* for reference information about the MAINTAIN_SCHEMAS procedure

Tutorial: Configuring Two-Database Replication with a Downstream Capture Process

The example in this topic configures an Oracle Streams replication environment that replicates data manipulation language (DML) changes to all of the tables in the hr schema. This example configures a two-database replication environment with a downstream capture process at the destination database. This example uses the global database names src.example.com and dest.example.com. However, you can substitute databases in your environment to complete the example. See "About Two-Database Replication Environments" on page 4-11 for more information about two-database replication environments.

In this example, the downstream capture process runs on the destination database dest.example.com. Therefore, the resources required to capture changes are freed at the source database src.example.com. This example configures a real-time downstream capture process, not an archived-log downstream capture process. The advantage of real-time downstream capture is that it reduces the amount of time required to capture the changes made at the source database. The time is reduced because the real-time downstream capture process does not need to wait for the redo log file to be archived before it can capture data from it.

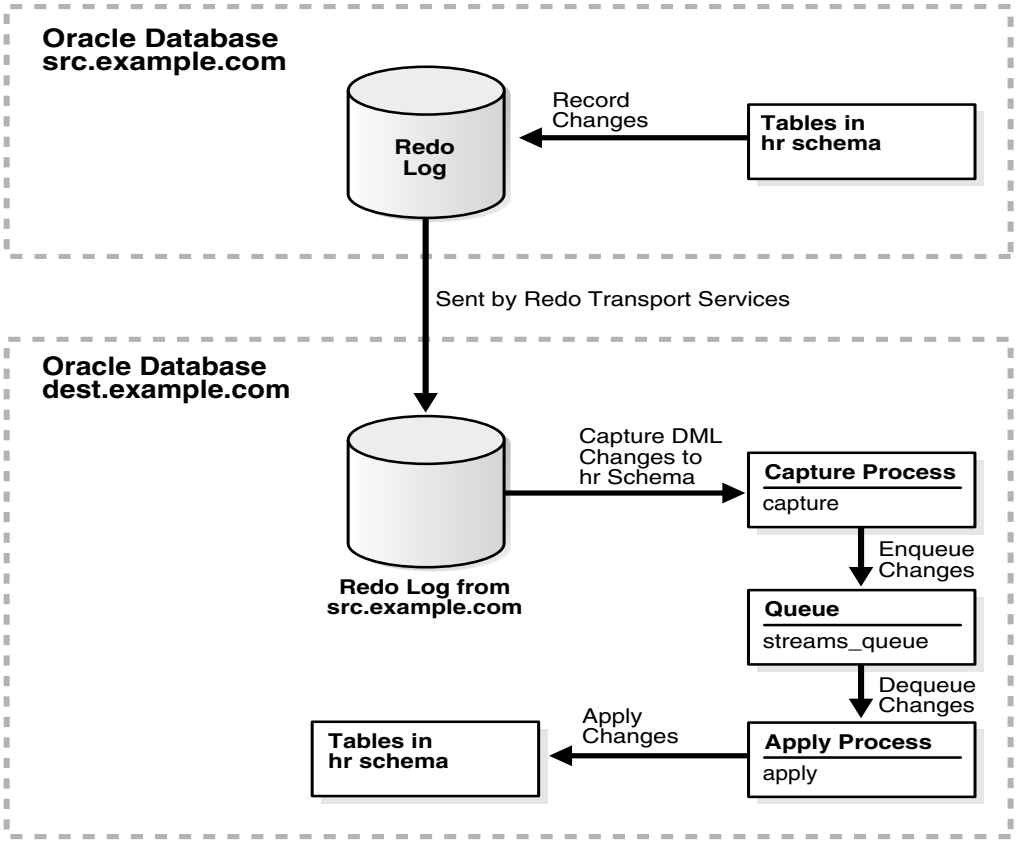
This example assumes that the replicated database objects are used for reporting and analysis at the destination database. Therefore, these database objects are assumed to be read-only at the `dest.example.com` database.

This example uses the `MAINTAIN_SCHEMAS` procedure in the `DBMS_STREAMS_ADM` package to configure the two-database replication environment. This procedure is the fastest and simplest way to configure an Oracle Streams environment that replicates one or more schemas. In addition, the procedure follows established best practices for Oracle Streams replication environments.

The database objects being configured for replication might or might not exist at the destination database when you run the `MAINTAIN_SCHEMAS` procedure. If the database objects do not exist at the destination database, then the `MAINTAIN_SCHEMAS` procedure instantiates them at the destination database using a Data Pump export/import. During instantiation, the instantiation SCN is set for these database objects. If the database objects already exist at the destination database, then the `MAINTAIN_SCHEMAS` procedure retains the existing database objects and sets the instantiation SCN for them. In this example, the `hr` schema exists at both the `src.example.com` database and the `dest.example.com` database before the `MAINTAIN_SCHEMAS` procedure is run.

Figure 4–12 provides an overview of the environment created in this example.

Figure 4–12 Two-Database Replication Environment with a Downstream Capture Process



Note: Local capture processes provide more flexibility than downstream capture processes in replication environments with different platforms or different versions of Oracle Database. See *Oracle Streams Concepts and Administration* for more information.

To configure this two-database replication environment:

1. Complete the following tasks to prepare for the two-database replication environment:

   a. Configure network connectivity so that the `src.example.com` database and the `dest.example.com` database can communicate with each other.

      See *Oracle Database 2 Day DBA* for information about configuring network connectivity between databases.

   b. Configure an Oracle Streams administrator at each database that will participate in the replication environment. See "Tutorial: Creating an Oracle Streams Administrator" on page 2-2 for instructions. This example assumes that the Oracle Streams administrator is `strmadmin`.

   c. Create a database link from the source database to the destination database and from the destination database to the source database. In this example, create the following database links:

      – From the `src.example.com` database to the `dest.example.com` database. Both the name and the service name of the database link must be `dest.example.com`.

      – From the `dest.example.com` database to the `src.example.com` database. Both the name and the service name of the database link must be `src.example.com`.

      The database link from the `dest.example.com` database to the `src.example.com` database is necessary because the `src.example.com` database is the source database for the downstream capture process at the `dest.example.com` database. This database link simplifies the creation and configuration of the capture process.

      Each database link should be created in the Oracle Streams administrator's schema. Also, each database link should connect to the Oracle Streams administrator at the other database. See "Tutorial: Creating a Database Link" on page 2-8 for instructions.

   d. Set initialization parameters properly at each database that will participate in the Oracle Streams replication environment. See "Preparing for Oracle Streams Replication" on page 4-21 for instructions.

   e. Configure both databases to run in `ARCHIVELOG` mode. For a downstream capture process to capture changes generated at a source database, both the source database and the downstream capture database must be running in `ARCHIVELOG` mode. In this example, the `src.example.com` and `dest.example.com` databases must be running in `ARCHIVELOG` mode. See *Oracle Database Administrator's Guide* for information about configuring a database to run in `ARCHIVELOG` mode.

   f. Configure authentication at both databases to support the transfer of redo data.

Redo transport sessions are authenticated using either the Secure Sockets Layer (SSL) protocol or a remote login password file. If the source database has a remote login password file, then copy it to the appropriate directory on the downstream capture database system. The password file must be the same at the source database and the downstream capture database.

In this example, the source database is `src.example.com` and the downstream capture database is `dest.example.com`. See *Oracle Data Guard Concepts and Administration* for detailed information about authentication requirements for redo transport.

2. At the source database `src.example.com`, set the following initialization parameters to configure redo transport services to transmit redo data from the online redo log at the source database to the standby redo log at the downstream database `dest.example.com`:

   - At the source database, configure at least one `LOG_ARCHIVE_DEST_`*n* initialization parameter to transmit redo data to the downstream database. To do this, set the following attributes of this parameter:
     - `SERVICE` - Specify the network service name of the downstream database.
     - `ASYNC` or `SYNC` - Specify a redo transport mode.

       The advantage of specifying `ASYNC` is that it results in little or no effect on the performance of the source database. If the source database is running Oracle Database 10*g* Release 1 or later, then `ASYNC` is recommended to avoid affecting source database performance if the downstream database or network is performing poorly.

       The advantage of specifying `SYNC` is that redo data is sent to the downstream database faster then when `ASYNC` is specified. Also, specifying `SYNC AFFIRM` results in behavior that is similar to `MAXIMUM AVAILABILITY` standby protection mode. Note that specifying an `ALTER DATABASE STANDBY DATABASE TO MAXIMIZE AVAILABILITY` SQL statement has no effect on an Oracle Streams capture process.
     - `NOREGISTER` - Specify this attribute so that the location of the archived redo log files is not recorded in the downstream database control file.
     - `VALID_FOR` - Specify either `(ONLINE_LOGFILE,PRIMARY_ROLE)` or `(ONLINE_LOGFILE,ALL_ROLES)`.
     - `DB_UNIQUE_NAME` - The unique name of the downstream database. Use the name specified for the `DB_UNIQUE_NAME` initialization parameter at the downstream database.

     The following example is a `LOG_ARCHIVE_DEST_`*n* setting that specifies a downstream database:

     ```
     LOG_ARCHIVE_DEST_2='SERVICE=DEST.EXAMPLE.COM ASYNC NOREGISTER
        VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE)
        DB_UNIQUE_NAME=dest'
     ```

   - `LOG_ARCHIVE_DEST_STATE_`*n* - At the source database, set this initialization parameter that corresponds with the `LOG_ARCHIVE_DEST_`*n* parameter for the downstream database to `ENABLE`.

     For example, if the `LOG_ARCHIVE_DEST_2` initialization parameter is set for the downstream database, then set the `LOG_ARCHIVE_DEST_STATE_2` parameter in the following way:

     ```
     LOG_ARCHIVE_DEST_STATE_2=ENABLE
     ```

- LOG_ARCHIVE_CONFIG - Set the DB_CONFIG attribute in this initialization parameter to include the DB_UNIQUE_NAME of the source database and the downstream database.

  For example, if the DB_UNIQUE_NAME of the source database is src, and the DB_UNIQUE_NAME of the downstream database is dest, then specify the following parameter:

  ```
  LOG_ARCHIVE_CONFIG='DG_CONFIG=(src,dest)'
  ```

  By default, the LOG_ARCHIVE_CONFIG parameter enables a database to both send and receive redo.

  See Also: *Oracle Database Reference* and *Oracle Data Guard Concepts and Administration* for more information about these initialization parameters

3. At the downstream database dest.example.com, set the following initialization parameters to configure archiving of the redo data generated locally:

   - Set at least one archive log destination in the LOG_ARCHIVE_DEST_*n* initialization parameter to either a directory or to the flash recovery area on the computer system running the downstream database. To do this, set the following attributes of this parameter:

     – LOCATION - Specify either a valid path name for a disk directory or USE_DB_RECOVERY_FILE_DEST. Each destination that specifies the LOCATION attribute must specify either a unique directory path name or USE_DB_RECOVERY_FILE_DEST. This is the local destination for archived redo log files generated by the local database.

     – VALID_FOR - Specify either (ONLINE_LOGFILE,PRIMARY_ROLE) or (ONLINE_LOGFILE,ALL_ROLES).

     The following example is a LOG_ARCHIVE_DEST_*n* setting for the locally generated redo data at the real-time downstream capture database:

     ```
     LOG_ARCHIVE_DEST_1='LOCATION=/home/arc_dest/local_rl
        VALID_FOR=(ONLINE_LOGFILE,PRIMARY_ROLE)'
     ```

     A real-time downstream capture configuration should keep archived standby redo log files separate from archived online redo log files from the downstream database. Specify ONLINE_LOGFILE instead of ALL_LOGFILES for the redo log type in the VALID_FOR attribute to accomplish this.

     You can specify other attributes in the LOG_ARCHIVE_DEST_*n* initialization parameter if necessary.

   - Set the LOG_ARCHIVE_DEST_STATE_*n* initialization parameter that corresponds with the LOG_ARCHIVE_DEST_*n* parameter previously set in this step to ENABLE.

     For example, if the LOG_ARCHIVE_DEST_1 initialization parameter is set, then set the LOG_ARCHIVE_DEST_STATE_1 parameter in the following way:

     ```
     LOG_ARCHIVE_DEST_STATE_1=ENABLE
     ```

4. At the downstream database dest.example.com, set the following initialization parameters to configure the downstream database to receive redo data from the

source database and write the redo data to the standby redo log at the downstream database:

- Set at least one archive log destination in the LOG_ARCHIVE_DEST_*n* initialization parameter to either a directory or to the flash recovery area on the computer system running the downstream database. To do this, set the following attributes of this parameter:
  - LOCATION - Specify either a valid path name for a disk directory or USE_DB_RECOVERY_FILE_DEST. Each destination that specifies the LOCATION attribute must specify either a unique directory path name or USE_DB_RECOVERY_FILE_DEST. This is the local destination for archived redo log files written from the standby redo logs. Log files from a remote source database should be kept separate from local database log files.
  - VALID_FOR - Specify either (STANDBY_LOGFILE,PRIMARY_ROLE) or (STANDBY_LOGFILE,ALL_ROLES).

  The following example is a LOG_ARCHIVE_DEST_*n* setting at the real-time downstream capture database:

  ```
  LOG_ARCHIVE_DEST_2='LOCATION=/home/arc_dest/srl_src1
     VALID_FOR=(STANDBY_LOGFILE,PRIMARY_ROLE)'
  ```

  You can specify other attributes in the LOG_ARCHIVE_DEST_*n* initialization parameter if necessary.
- Set the LOG_ARCHIVE_DEST_STATE_*n* initialization parameter that corresponds with the LOG_ARCHIVE_DEST_*n* parameter previously set in this step to ENABLE.

  For example, if the LOG_ARCHIVE_DEST_2 initialization parameter is set for the downstream database, then set the LOG_ARCHIVE_DEST_STATE_2 parameter in the following way:

  ```
  LOG_ARCHIVE_DEST_STATE_2=ENABLE
  ```

- LOG_ARCHIVE_CONFIG - Set the DB_CONFIG attribute in this initialization parameter to include the DB_UNIQUE_NAME of the source database and the downstream database.

  For example, if the DB_UNIQUE_NAME of the source database is src, and the DB_UNIQUE_NAME of the downstream database is dest, then specify the following parameter:

  ```
  LOG_ARCHIVE_CONFIG='DG_CONFIG=(src,dest)'
  ```

  By default, the LOG_ARCHIVE_CONFIG parameter enables a database to both send and receive redo.

5. If you reset any initialization parameters while an instance was running at a database in Step 2, 3 or 4, then consider resetting them in the relevant initialization parameter file as well, so that the new values are retained when the database is restarted.

   If you did not reset the initialization parameters while an instance was running, but instead reset them in the initialization parameter file in Step 2, 3 or 4, then restart the database. The source database must be open when it sends redo data to the downstream database, because the global name of the source database is sent to the downstream database only if the source database is open.

6. At the downstream database `dest.example.com`, connect as an administrative user and create standby redo log files.

   Note: The following steps outline the general procedure for adding standby redo log files to the downstream database. The specific steps and SQL statements used to add standby redo log files depend on your environment. For example, in an Oracle Real Application Clusters (Oracle RAC) environment, the steps are different. See *Oracle Data Guard Concepts and Administration* for detailed instructions about adding standby redo log files to a database.

   a. Open SQL*Plus and connect to the source database `src.example.com` as an administrative user.

      For example, to connect to the `src.example.com` database as an administrative user:

      ```
      sqlplus system@src.example.com
      Enter password: password
      ```

      See *Oracle Database 2 Day DBA* for more information about starting SQL*Plus.

   b. Determine the log file size used on the source database `src.example.com`. The standby log file size must exactly match (or be larger than) the source database log file size. For example, if the source database log file size is 500 MB, then the standby log file size must be 500 MB or larger. You can determine the size of the redo log files at the source database (in bytes) by querying the `V$LOG` view at the source database.

      For example, query the `V$LOG` view:

      ```
      SELECT BYTES FROM V$LOG;
      ```

   c. Determine the number of standby log file groups required on the downstream database `dest.example.com`. The number of standby log file groups must be at least one more than the number of online log file groups on the source database. For example, if the source database has two online log file groups, then the downstream database must have at least three standby log file groups. You can determine the number of source database online log file groups by querying the `V$LOG` view at the source database.

      For example, while still connected in SQL*Plus as an administrative user to `src.example.com`, query the `V$LOG` view:

      ```
      SELECT COUNT(GROUP#) FROM V$LOG;
      ```

   d. In SQL*Plus, connect to the downstream database `dest.example.com` as an administrative user.

      ```
      CONNECT system@dest.example.com
      Enter password: password
      ```

   e. Use the SQL statement `ALTER DATABASE ADD STANDBY LOGFILE` to add the standby log file groups to the downstream database `dest.example.com`.

      For example, assume that the source database has two online redo log file groups and is using a log file size of 500 MB. In this case, use the following statements to create the appropriate standby log file groups:

```
ALTER DATABASE ADD STANDBY LOGFILE GROUP 3
   ('/oracle/dbs/slog3a.rdo', '/oracle/dbs/slog3b.rdo') SIZE 500M;

ALTER DATABASE ADD STANDBY LOGFILE GROUP 4
   ('/oracle/dbs/slog4a.rdo', '/oracle/dbs/slog4b.rdo') SIZE 500M;

ALTER DATABASE ADD STANDBY LOGFILE GROUP 5
   ('/oracle/dbs/slog5a.rdo', '/oracle/dbs/slog5b.rdo') SIZE 500M;
```

f. Ensure that the standby log file groups were added successfully by running the following query at `dest.example.com`:

```
SELECT GROUP#, THREAD#, SEQUENCE#, ARCHIVED, STATUS
   FROM V$STANDBY_LOG;
```

Your output should be similar to the following:

```
    GROUP#    THREAD#  SEQUENCE# ARC STATUS
---------- ---------- ---------- --- ----------
         3          0          0 YES UNASSIGNED
         4          0          0 YES UNASSIGNED
         5          0          0 YES UNASSIGNED
```

g. Ensure that log files from the source database are appearing in the directory specified in the `LOCATION` attribute in Step 4. You might need to switch the log file at the source database to see files in the directory.

7. In SQL*Plus, connect to the `src.example.com` database as the Oracle Streams administrator.

See *Oracle Database Administrator's Guide* for information about connecting to a database in SQL*Plus.

8. Create a directory object to hold files that will be generated by the `MAINTAIN_SCHEMAS` procedure, including the Data Pump export dump file used for instantiation. The directory object can point to any accessible directory on the computer system. For example, the following statement creates a directory object named `src_dir` that points to the /usr/src_log_files directory:

```
CREATE DIRECTORY src_dir AS '/usr/src_log_files';
```

9. In SQL*Plus, connect to the `dest.example.com` database as the Oracle Streams administrator.

10. Create a directory object to hold files that will be generated by the `MAINTAIN_SCHEMAS` procedure, including the Data Pump export dump file used for instantiation. The directory object can point to any accessible directory on the computer system. For example, the following statement creates a directory object named `dest_dir` that points to the /usr/dest_log_files directory:

```
CREATE DIRECTORY dest_dir AS '/usr/dest_log_files';
```

11. While still connected to the `dest.example.com` database as the Oracle Streams administrator, run the `MAINTAIN_SCHEMAS` procedure to configure replication between the `src.example.com` database and the `dest.example.com` database:

```
BEGIN
  DBMS_STREAMS_ADM.MAINTAIN_SCHEMAS(
    schema_names                 => 'hr',
    source_directory_object      => 'src_dir',
    destination_directory_object => 'dest_dir',
    source_database              => 'src.example.com',
```

```
    destination_database         => 'dest.example.com',
    capture_name                 => 'capture',
    capture_queue_table          => 'streams_queue_qt',
    capture_queue_name           => 'streams_queue',
    apply_name                   => 'apply',
    apply_queue_table            => 'streams_queue_qt',
    apply_queue_name             => 'streams_queue');
END;
/
```

The MAINTAIN_SCHEMAS procedure can take some time to run because it is performing many configuration tasks. Do not allow data manipulation language (DML) or data definition language (DDL) changes to the replicated database objects at the destination database while the procedure is running.

In the MAINTAIN_SCHEMAS procedure, only the following parameters are required: schema_names, source_directory_object, destination_directory_object, source_database, and destination_database. See "About the Oracle Streams Replication Configuration Procedures" on page 4-16.

This example specifies the other parameters to show that you can choose the name for the capture process, capture process queue table, capture process queue, apply process, apply process queue table, and apply process queue. If you do not specify these parameters, then system-generated names are used.

When you use a configuration procedure to configure downstream capture, the parameters that specify the queue and queue table names are important. In such a configuration, it is more efficient for the capture process and apply process to use the same queue at the downstream capture database to avoid propagating changes between queues. To improve efficiency in this sample configuration, notice that streams_queue is specified for both the capture_queue_name and apply_queue_name parameters. Also, streams_queue_qt is specified for both the capture_queue_table and apply_queue_table parameters.

When a configuration procedure is run, information about its progress is recorded in the following data dictionary views: DBA_RECOVERABLE_SCRIPT, DBA_RECOVERABLE_SCRIPT_PARAMS, DBA_RECOVERABLE_SCRIPT_BLOCKS, and DBA_RECOVERABLE_SCRIPT_ERRORS. If the procedure stops because it encounters an error, then see *Oracle Streams Replication Administrator's Guide* for instructions about using the RECOVER_OPERATION procedure in the DBMS_STREAMS_ADM package to recover from these errors.

Wait until the procedure completes successfully before proceeding to the next step.

12. While still connected to the dest.example.com database as the Oracle Streams administrator, set the downstream_real_time_mine capture process parameter to Y:

```
BEGIN
  DBMS_CAPTURE_ADM.SET_PARAMETER(
    capture_name => 'capture',
    parameter    => 'downstream_real_time_mine',
    value        => 'Y');
END;
/
```

If you would rather set the capture process parameter using Enterprise Manager, then see "Setting a Capture Process Parameter" on page 5-3 for instructions.

13. In SQL*Plus, connect to the source database `src.example.com` as an administrative user.
14. Archive the current log file at the source database:

```
ALTER SYSTEM ARCHIVE LOG CURRENT;
```

    Archiving the current log file at the source database starts real-time mining of the source database redo log.

When you complete the example, a two-database replication environment with the following characteristics is configured:

- At the `src.example.com` database, supplemental logging is configured for the tables in the `hr` schema.
- The `dest.example.com` database has the following components:
  - A downstream capture process named `capture`. The capture process captures changes to the `hr` schema in the redo log information sent from the source database `src.example.com`.
  - A queue named `streams_queue` that uses a queue table named `streams_queue_qt`. This queue is for the capture process and apply process at the database.
  - An apply process named `apply`. The apply process applies changes to the `hr` schema.

To check the Oracle Streams replication configuration:

1. At the `dest.example.com` database, ensure that the capture process is enabled and that the capture type is downstream. To do so, follow the instructions in "Viewing Information About a Capture Process" on page 5-12, and check the Status and Capture Type fields on the Capture subpage.
2. At the `dest.example.com` database, ensure that the apply process is enabled. To do so, follow the instructions in "Viewing Information About an Apply Process" on page 5-20, and check Status field on the Apply subpage.

To replicate changes:

1. At the `src.example.com` database, make DML changes to any table in the `hr` schema, and commit the changes.
2. After some time has passed to allow for replication of the changes, use SQL*Plus to query the modified table at the `dest.example.com` database to view the DML changes.

> **Note:** The configuration procedures in the `DBMS_STREAMS_ADM` package do not configure the replicated tables to be read only at the destination database. If they should be read only, then configure privileges at the destination database accordingly. However, the apply user for the apply process must be able to make DML changes to the replicated database objects. In this example, the apply user is the Oracle Streams administrator. See *Oracle Database Security Guide* for information about configuring privileges.

See Also:

- "About Hub-And-Spoke Replication Environments" on page 4-13
- "When to Replicate Data with Oracle Streams" on page 1-4
- Chapter 5, "Administering an Oracle Streams Replication Environment"
- *Oracle Streams Concepts and Administration* for more information about downstream capture processes
- *Oracle Streams Replication Administrator's Guide* for more detailed instructions about using the `MAINTAIN_SCHEMAS` procedure
- *Oracle Database PL/SQL Packages and Types Reference* for reference information about the `MAINTAIN_SCHEMAS` procedure

Tutorial: Configuring Hub-and-Spoke Replication with Local Capture Processes

The example in this topic configures an Oracle Streams hub-and-spoke replication environment that replicates data manipulation language (DML) changes to all of the tables in the `hr` schema. This example uses a capture process at each database to capture these changes. Hub-and-spoke replication means that a central hub database replicates changes with one or more spoke databases. The spoke databases do not communicate with each other directly. In this sample configuration, the hub database sends changes generated at one spoke database to the other spoke database. See "About Hub-And-Spoke Replication Environments" on page 4-13 for more information about hub-and-spoke replication environments.

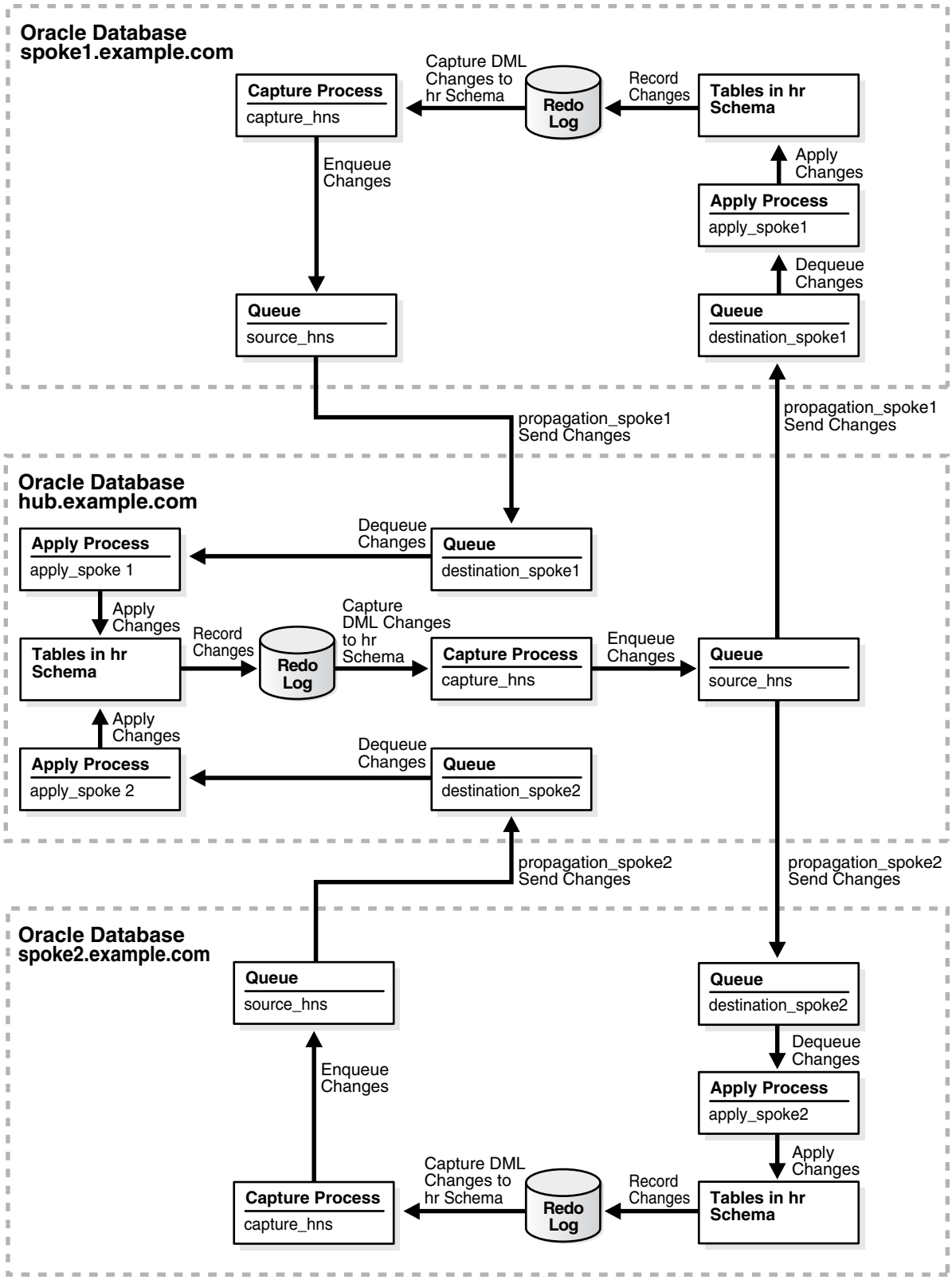
This example uses the `MAINTAIN_SCHEMAS` procedure in the `DBMS_STREAMS_ADM` package to configure the hub-and-spoke replication environment. This procedure is the fastest and simplest way to configure an Oracle Streams environment that replicates one or more schemas. In addition, the procedure follows established best practices for Oracle Streams replication environments.

In this example, the global name of the hub database is `hub.example.com`, and the global names of the spoke databases are `spoke1.example.com` and `spoke2.example.com`. However, you can substitute databases in your environment to complete the example.

The database objects being configured for replication might or might not exist at the destination databases when you run the `MAINTAIN_SCHEMAS` procedure. If the database objects do not exist at a destination database, then the `MAINTAIN_SCHEMAS` procedure instantiates them at the destination database using a Data Pump export/import. During instantiation, the instantiation SCN is set for these database objects. If the database objects already exist at a destination database, then the `MAINTAIN_SCHEMAS` procedure retains the existing database objects and sets the instantiation SCN for them. In this example, the `hr` schema exists at each database before the `MAINTAIN_SCHEMAS` procedure is run.

Figure 4–13 provides an overview of the environment created in this example.

Figure 4–13 Sample Hub-and-Spoke Environment with Capture Processes and Read/Write Spokes



To configure this hub-and-spoke replication environment with read/write spokes:

1. Complete the following tasks to prepare for the hub-and-spoke replication environment:

   a. Configure network connectivity so that the following databases can communicate with each other:

      – The `hub.example.com` database and the `spoke1.example.com` database

      – The `hub.example.com` database and the `spoke2.example.com` database

      See *Oracle Database 2 Day DBA* for information about configuring network connectivity between databases.

   b. Configure an Oracle Streams administrator at each database that will participate in the replication environment. See "Tutorial: Creating an Oracle Streams Administrator" on page 2-2 for instructions. This example assumes that the Oracle Streams administrator is `strmadmin`.

   c. Create a database link from the hub database to each spoke database and from each spoke database to the hub database. In this example, create the following database links:

      – From the `hub.example.com` database to the `spoke1.example.com` database. Both the name and the service name of the database link must be `spoke1.example.com`.

      – From the `hub.example.com` database to the `spoke2.example.com` database. Both the name and the service name of the database link must be `spoke2.example.com`.

      – From the `spoke1.example.com` database to the `hub.example.com` database. Both the name and the service name of the database link must be `hub.example.com`.

      – From the `spoke2.example.com` database to the `hub.example.com` database. Both the name and the service name of the database link must be `hub.example.com`.

      Each database link should be created in the Oracle Streams administrator's schema. Also, each database link should connect to the Oracle Streams administrator at the destination database. See "Tutorial: Creating a Database Link" on page 2-8 for instructions.

   d. Set initialization parameters properly at each database that will participate in the Oracle Streams replication environment. See "Preparing for Oracle Streams Replication" on page 4-21 for instructions.

   e. Configure each source database to run in `ARCHIVELOG` mode. For a capture process to capture changes generated at a source database, the source database must be running in `ARCHIVELOG` mode. In this example, all databases must be running in `ARCHIVELOG` mode. See *Oracle Database Administrator's Guide* for information about configuring a database to run in `ARCHIVELOG` mode.

2. On a command line, open SQL*Plus and connect to the `spoke1.example.com` database as the Oracle Streams administrator.

   See *Oracle Database 2 Day DBA* for more information about starting SQL*Plus.

3. Create a directory object to hold files that will be generated by the MAINTAIN_SCHEMAS procedure, including the Data Pump export dump file used for instantiation. The directory object can point to any accessible directory on the computer system. For example, the following statement creates a directory object named spoke1_dir that points to the /usr/spoke1_log_files directory:

```
CREATE DIRECTORY spoke1_dir AS '/usr/spoke1_log_files';
```

4. In SQL*Plus, connect to the spoke2.example.com database as the Oracle Streams administrator.

   See *Oracle Database Administrator's Guide* for information about connecting to a database in SQL*Plus.

5. Create a directory object to hold files that will be generated by the MAINTAIN_SCHEMAS procedure, including the Data Pump export dump file used for instantiation. The directory object can point to any accessible directory on the computer system. For example, the following statement creates a directory object named spoke2_dir that points to the /usr/spoke2_log_files directory:

```
CREATE DIRECTORY spoke2_dir AS '/usr/spoke2_log_files';
```

6. In SQL*Plus, connect to the hub.example.com database as the Oracle Streams administrator.

7. Create a directory object to hold files that will be generated by the MAINTAIN_SCHEMAS procedure, including the Data Pump export dump file used for instantiation. The directory object can point to any accessible directory on the computer system. For example, the following statement creates a directory object named hub_dir that points to the /usr/hub_log_files directory:

```
CREATE DIRECTORY hub_dir AS '/usr/hub_log_files';
```

8. While still connected in SQL*Plus to the hub.example.com database as the Oracle Streams administrator, run the MAINTAIN_SCHEMAS procedure to configure replication between the hub.example.com database and the spoke1.example.com database:

```
BEGIN
  DBMS_STREAMS_ADM.MAINTAIN_SCHEMAS(
    schema_names                 => 'hr',
    source_directory_object      => 'hub_dir',
    destination_directory_object => 'spoke1_dir',
    source_database              => 'hub.example.com',
    destination_database         => 'spoke1.example.com',
    capture_name                 => 'capture_hns',
    capture_queue_table          => 'source_hns_qt',
    capture_queue_name           => 'source_hns',
    propagation_name             => 'propagation_spoke1',
    apply_name                   => 'apply_spoke1',
    apply_queue_table            => 'destination_spoke1_qt',
    apply_queue_name             => 'destination_spoke1',
    bi_directional               => TRUE);
END;
/
```

   The MAINTAIN_SCHEMAS procedure can take some time to run because it is performing many configuration tasks. Do not allow data manipulation language (DML) or data definition language (DDL) changes to the replicated database objects at the destination database while the procedure is running.

In the MAINTAIN_SCHEMAS procedure, only the following parameters are required: schema_names, source_directory_object, destination_directory_object, source_database, and destination_database. Also, when you use a configuration procedure to configure bi-directional replication, the bi_directional parameter must be set to TRUE. See "About the Oracle Streams Replication Configuration Procedures" on page 4-16.

This example specifies the other parameters to show that you can choose the name for the capture process, capture process queue table, capture process queue, propagation, apply process, apply process queue table, and apply process queue. If you do not specify these parameters, then system-generated names are used.

When a configuration procedure is run, information about its progress is recorded in the following data dictionary views: DBA_RECOVERABLE_SCRIPT, DBA_RECOVERABLE_SCRIPT_PARAMS, DBA_RECOVERABLE_SCRIPT_BLOCKS, and DBA_RECOVERABLE_SCRIPT_ERRORS. If the procedure stops because it encounters an error, then see *Oracle Streams Replication Administrator's Guide* for instructions about using the RECOVER_OPERATION procedure in the DBMS_STREAMS_ADM package to recover from these errors.

9. While still connected in SQL*Plus to the hub.example.com database as the Oracle Streams administrator, run the MAINTAIN_SCHEMAS procedure to configure replication between the hub.example.com database and the spoke2.example.com database:

```
BEGIN
  DBMS_STREAMS_ADM.MAINTAIN_SCHEMAS(
    schema_names                 => 'hr',
    source_directory_object      => 'hub_dir',
    destination_directory_object => 'spoke2_dir',
    source_database              => 'hub.example.com',
    destination_database         => 'spoke2.example.com',
    capture_name                 => 'capture_hns',
    capture_queue_table          => 'source_hns_qt',
    capture_queue_name           => 'source_hns',
    propagation_name             => 'propagation_spoke2',
    apply_name                   => 'apply_spoke2',
    apply_queue_table            => 'destination_spoke2_qt',
    apply_queue_name             => 'destination_spoke2',
    bi_directional               => TRUE);
END;
/
```

10. Configure latest time conflict resolution for all of the tables in the hr schema at the hub.example.com, spoke1.example.com, and spoke2.example.com databases. This schema includes the countries, departments, employees, jobs, job_history, locations, and regions tables. See "Tutorial: Configuring Latest Time Conflict Resolution for a Table" on page 4-56 for instructions.

When you complete the example, a hub-and-spoke replication environment with the following characteristics is configured:

- Supplemental logging is configured for the tables in the hr schema at each database.
- Each database has a capture process named capture_hns. The capture process captures changes to the hr schema at the database.
- Each database has a queue named source_hns that uses a queue table named source_hns_qt. This queue is for the capture process at the database.

- The hub database hub.example.com has the following additional components:
  - An apply process named apply_spoke1. This apply process applies changes to the hr schema that were sent from the spoke1.example.com database.
  - A queue named destination_spoke1 that uses a queue table named destination_spoke1_qt. This queue is for the apply_spoke1 apply process at the database.
  - An apply process named apply_spoke2. This apply process applies changes to the hr schema that were sent from the spoke2.example.com database.
  - A queue named destination_spoke2 that uses a queue table named destination_spoke2_qt. This queue is for the apply_spoke2 apply process at the database.
  - A propagation named propagation_spoke1. This propagation sends changes to the hr schema from the source_hns queue to the destination_spoke1 queue at the spoke1.example.com database.
  - A propagation named propagation_spoke2. This propagation sends changes to the hr schema from the source_hns queue to the destination_spoke2 queue at the spoke2.example.com database.
- The spoke database spoke1.example.com has the following additional components:
  - An apply process named apply_spoke1. The apply process applies changes to the hr schema that were sent from the hub.example.com database.
  - A queue named destination_spoke1 that uses a queue table named destination_spoke1_qt. This queue is for the apply_spoke1 apply process at the database.
  - A propagation named propagation_spoke1. This propagation sends changes to the hr schema from the source_hns queue to the destination_spoke1 queue at the hub.example.com database.
- The spoke database spoke2.example.com has the following additional components:
  - An apply process named apply_spoke2. The apply process applies changes to the hr schema that were sent from the hub.example.com database.
  - A queue named destination_spoke2 that uses a queue table named destination_spoke2_qt. This queue is for the apply_spoke2 apply process at the database.
  - A propagation named propagation_spoke2. This propagation sends changes to the hr schema from the source_hns queue to the destination_spoke2 queue at the hub.example.com database.
- Tags are used to avoid change cycling in the following way:
  - Each apply process uses an apply tag, and redo records for changes applied by the apply process include the tag. Each apply process uses an apply tag that is unique in the replication environment.
  - Each capture process captures all of the changes to the replicated database objects, regardless of the tag in the redo record. Therefore, each capture process captures the changes applied by the apply processes on its source database.

- Each propagation sends all changes made to the replicated database objects to another database in the replication environment, except for changes that originated at the other database. The propagation rules instruct the propagation to discard these changes.

See "About Tags for Avoiding Change Cycling" on page 4-10 and *Oracle Database PL/SQL Packages and Types Reference* for more information about how the replication environment avoids change cycling.

To check the Oracle Streams replication configuration:

1. At each database, ensure that the capture process is enabled and that the capture type is local. To do so, follow the instructions in "Viewing Information About a Capture Process" on page 5-12, and check the Status and Capture Type fields on the Capture subpage.
2. At each database, ensure that each propagation is enabled. To do so, follow the instructions in "Viewing Information About a Propagation" on page 5-16, and check the Status field on the Propagation subpage. The hub database should have two propagations, and they should both be enabled. Each spoke database should have one propagation that is enabled.
3. At each database, ensure that each apply process is enabled. To do so, follow the instructions in "Viewing Information About an Apply Process" on page 5-20, and check the Status field on the Apply subpage. The hub database should have two apply processes, and they should both be enabled. Each spoke database should have one apply process that is enabled.

To replicate changes:

1. At one of the databases, make DML changes to any table in the `hr` schema.
2. After some time has passed to allow for replication of the changes, use SQL*Plus to query the modified table at the other databases to view the DML changes.

See Also:

- "About Hub-And-Spoke Replication Environments" on page 4-13
- "When to Replicate Data with Oracle Streams" on page 1-4
- Chapter 5, "Administering an Oracle Streams Replication Environment"
- *Oracle Streams Replication Administrator's Guide* for more detailed instructions about using the `MAINTAIN_SCHEMAS` procedure
- Oracle Database PL/SQL Packages and Types Reference for reference information about the `MAINTAIN_SCHEMAS` procedure

Tutorial: Configuring Two-Database Replication with Synchronous Captures

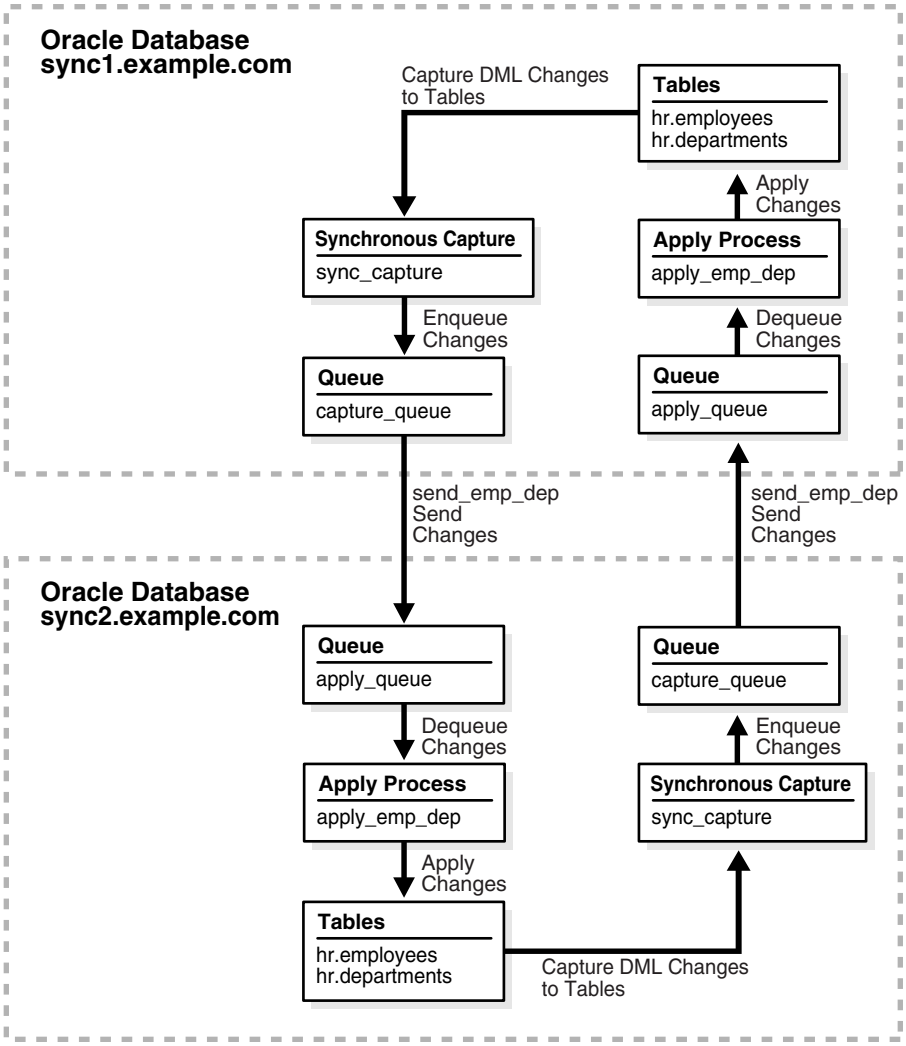
The example in this topic configures an Oracle Streams replication environment that replicates data manipulation language (DML) changes to two tables in the `hr` schema. This example uses a synchronous capture at each database to capture these changes. In this example, the global names of the databases in the Oracle Streams replication environment are `sync1.example.com` and `sync2.example.com`. However, you can substitute any two databases in your environment to complete the example. See "About Two-Database Replication Environments" on page 4-11 for more information about two-database replication environments.

Specifically, this example configures a two-database Oracle Streams replication environment that shares the `hr.employees` and `hr.departments` tables at the `sync1.example.com` and `sync2.example.com` databases. The two databases replicate all of the DML changes to these tables. The `hr` sample schema is installed by default with Oracle Database.

Note: A synchronous capture can only capture changes at the table level. It cannot capture changes at the schema or database level. You can configure a synchronous capture using the `ADD_TABLE_RULES` and `ADD_SUBSET_RULES` procedures in the `DBMS_STREAMS_ADM` package.

Figure 4–14 provides an overview of the environment created in this example.

Figure 4–14 Two-Database Replication Environment with Synchronous Captures



To configure this replication environment with synchronous captures:

1. Complete the following tasks to prepare for the two-database replication environment:

   a. Configure network connectivity so that the two databases can communicate with each other. See *Oracle Database 2 Day DBA* for information about configuring network connectivity between databases.

   b. Configure an Oracle Streams administrator at each database that will participate in the replication environment. See "Tutorial: Creating an Oracle Streams Administrator" on page 2-2 for instructions. This example assumes that the Oracle Streams administrator is `strmadmin`.

   c. Set initialization parameters properly at each database that will participate in the Oracle Streams replication environment. See "Preparing for Oracle Streams Replication" on page 4-21 for instructions.

   d. Ensure that the `hr.employees` and `hr.departments` tables exist at the two databases and are consistent at these databases. If the database objects exist at only one database, then you can use export/import to create and populate them at the other database. See *Oracle Database Utilities* for information about export/import.

2. Create two `ANYDATA` queues at each database. For this example, create the following two queues at each database:

   - A queue named `capture_queue` owned by the Oracle Streams administrator `strmadmin`. This queue will be used by the synchronous capture at the database.
   - A queue named `apply_queue` owned by the Oracle Streams administrator `strmadmin`. This queue will be used by the apply process at the database.

   See "Creating an ANYDATA Queue" on page 2-7 for instructions.

3. Create a database link from each database to the other database:

   a. Create a database link from the `sync1.example.com` database to the `sync2.example.com` database. The database link should be created in the Oracle Streams administrator's schema. Also, the database link should connect to the Oracle Streams administrator at the `sync2.example.com` database. Both the name and the service name of the database link must be `sync2.example.com`.

   b. Create a database link from the `sync2.example.com` database to the `sync1.example.com` database. The database link should be created in the Oracle Streams administrator's schema. Also, the database link should connect to the Oracle Streams administrator at the `sync1.example.com` database. Both the name and the service name of the database link must be `sync1.example.com`.

   See "Tutorial: Creating a Database Link" on page 2-8 for instructions.

4. Configure an apply process at the `sync1.example.com` database. This apply process will apply changes to the shared tables that were captured at the `sync2.example.com` database and propagated to the `sync1.example.com` database.

   a. Open SQL*Plus and connect to the `sync1.example.com` database as the Oracle Streams administrator.

      See *Oracle Database 2 Day DBA* for more information about starting SQL*Plus.

b. Create the apply process:

```
BEGIN
  DBMS_APPLY_ADM.CREATE_APPLY(
    queue_name     => 'strmadmin.apply_queue',
    apply_name     => 'apply_emp_dep',
    apply_captured => FALSE);
END;
/
```

The `apply_captured` parameter is set to `FALSE` because the apply process applies changes in the persistent queue. These are changes that were captured by a synchronous capture. The `apply_captured` parameter should be set to `TRUE` only when the apply process applies changes captured by a capture process.

Do not start the apply process.

c. Add a rule to the apply process rule set:

```
BEGIN
  DBMS_STREAMS_ADM.ADD_TABLE_RULES(
    table_name      => 'hr.employees',
    streams_type    => 'apply',
    streams_name    => 'apply_emp_dep',
    queue_name      => 'strmadmin.apply_queue',
    source_database => 'sync2.example.com');
END;
/
```

This rule instructs the apply process `apply_emp_dep` to apply all DML changes to the `hr.employees` table that appear in the `apply_queue` queue. The rule also specifies that the apply process applies only changes that were captured at the `sync2.example.com` source database.

d. Add an additional rule to the apply process rule set:

```
BEGIN
  DBMS_STREAMS_ADM.ADD_TABLE_RULES(
    table_name      => 'hr.departments',
    streams_type    => 'apply',
    streams_name    => 'apply_emp_dep',
    queue_name      => 'strmadmin.apply_queue',
    source_database => 'sync2.example.com');
END;
/
```

This rule instructs the apply process `apply_emp_dep` to apply all DML changes to the `hr.departments` table that appear in the `apply_queue` queue. The rule also specifies that the apply process applies only changes that were captured at the `sync2.example.com` source database.

5. Configure an apply process at the `sync2.example.com` database. This apply process will apply changes that were captured at the `sync1.example.com` database and propagated to the `sync2.example.com` database.

a. In SQL*Plus, connect to the `sync2.example.com` database as the Oracle Streams administrator.

See *Oracle Database Administrator's Guide* for information about connecting to a database in SQL*Plus.

b. Create the apply process:

```
BEGIN
  DBMS_APPLY_ADM.CREATE_APPLY(
    queue_name     => 'strmadmin.apply_queue',
    apply_name     => 'apply_emp_dep',
    apply_captured => FALSE);
END;
/
```

The `apply_captured` parameter is set to `FALSE` because the apply process applies changes in the persistent queue. These changes were captured by a synchronous capture. The `apply_captured` parameter should be set to `TRUE` only when the apply process applies changes captured by a capture process.

Do not start the apply process.

c. Add a rule to the apply process rule set:

```
BEGIN
  DBMS_STREAMS_ADM.ADD_TABLE_RULES(
    table_name      => 'hr.employees',
    streams_type    => 'apply',
    streams_name    => 'apply_emp_dep',
    queue_name      => 'strmadmin.apply_queue',
    source_database => 'sync1.example.com');
END;
/
```

This rule instructs the apply process `apply_emp_dep` to apply all DML changes that appear in the `apply_queue` queue to the `hr.employees` table. The rule also specifies that the apply process applies only changes that were captured at the `sync1.example.com` source database.

d. Add an additional rule to the apply process rule set:

```
BEGIN
  DBMS_STREAMS_ADM.ADD_TABLE_RULES(
    table_name      => 'hr.departments',
    streams_type    => 'apply',
    streams_name    => 'apply_emp_dep',
    queue_name      => 'strmadmin.apply_queue',
    source_database => 'sync1.example.com');
END;
/
```

This rule instructs the apply process `apply_emp_dep` to apply all DML changes that appear in the `apply_queue` queue to the `hr.departments` table. The rule also specifies that the apply process applies only changes that were captured at the `sync1.example.com` source database.

6. Create a propagation to send changes from a queue at the `sync1.example.com` database to a queue at the `sync2.example.com` database:

a. In SQL*Plus, connect to the `sync1.example.com` database as the Oracle Streams administrator.

b. Create the propagation that sends changes to the `sync2.example.com` database:

```
BEGIN
  DBMS_STREAMS_ADM.ADD_TABLE_PROPAGATION_RULES(
    table_name              => 'hr.employees',
    streams_name            => 'send_emp_dep',
    source_queue_name       => 'strmadmin.capture_queue',
    destination_queue_name  => 'strmadmin.apply_queue@sync2.example.com',
    source_database         => 'sync1.example.com',
    queue_to_queue          => TRUE);
END;
/
```

The `ADD_TABLE_PROPAGATION_RULES` procedure creates the propagation and its positive rule set. This procedure also adds a rule to the propagation rule set that instructs it to send DML changes to the `hr.employees` table to the `apply_queue` queue in the `sync2.example.com` database.

c. Add an additional rule to the propagation rule set:

```
BEGIN
  DBMS_STREAMS_ADM.ADD_TABLE_PROPAGATION_RULES(
    table_name              => 'hr.departments',
    streams_name            => 'send_emp_dep',
    source_queue_name       => 'strmadmin.capture_queue',
    destination_queue_name  => 'strmadmin.apply_queue@sync2.example.com',
    source_database         => 'sync1.example.com',
    queue_to_queue          => TRUE);
END;
/
```

The `ADD_TABLE_PROPAGATION_RULES` procedure adds a rule to the propagation rule set that instructs it to send DML changes to the `hr.departments` table to the `apply_queue` queue in the `sync2.example.com` database.

7. Create a propagation to send changes from a queue at the `sync2.example.com` database to a queue at the `sync1.example.com` database:

a. In SQL*Plus, connect to the `sync2.example.com` database as the Oracle Streams administrator.

b. Create the propagation that sends changes to the `sync1.example.com` database:

```
BEGIN
  DBMS_STREAMS_ADM.ADD_TABLE_PROPAGATION_RULES(
    table_name              => 'hr.employees',
    streams_name            => 'send_emp_dep',
    source_queue_name       => 'strmadmin.capture_queue',
    destination_queue_name  => 'strmadmin.apply_queue@sync1.example.com',
    source_database         => 'sync2.example.com',
    queue_to_queue          => TRUE);
END;
/
```

The `ADD_TABLE_PROPAGATION_RULES` procedure creates the propagation and its positive rule set. This procedure also adds a rule to the propagation rule set that instructs it to send DML changes to the `hr.employees` table to the `apply_queue` queue in the `sync1.example.com` database.

c. Add an additional rule to the propagation rule set:

```
BEGIN
  DBMS_STREAMS_ADM.ADD_TABLE_PROPAGATION_RULES(
    table_name              => 'hr.departments',
    streams_name            => 'send_emp_dep',
    source_queue_name       => 'strmadmin.capture_queue',
    destination_queue_name  => 'strmadmin.apply_queue@sync1.example.com',
    source_database         => 'sync2.example.com',
    queue_to_queue          => TRUE);
END;
/
```

The `ADD_TABLE_PROPAGATION_RULES` procedure adds a rule to the propagation rule set that instructs it to send DML changes to the `hr.departments` table to the `apply_queue` queue in the `sync1.example.com` database.

8. Configure a synchronous capture at the `sync1.example.com` database:

   a. In SQL*Plus, connect to the `sync1.example.com` database as the Oracle Streams administrator.

   b. Run the `ADD_TABLE_RULES` procedure to create the synchronous capture and add a rule to instruct it to capture changes to the `hr.employees` table:

```
BEGIN
  DBMS_STREAMS_ADM.ADD_TABLE_RULES(
    table_name    => 'hr.employees',
    streams_type  => 'sync_capture',
    streams_name  => 'sync_capture',
    queue_name    => 'strmadmin.capture_queue');
END;
/
```

   c. Add an additional rule to the synchronous capture rule set:

```
BEGIN
  DBMS_STREAMS_ADM.ADD_TABLE_RULES(
    table_name    => 'hr.departments',
    streams_type  => 'sync_capture',
    streams_name  => 'sync_capture',
    queue_name    => 'strmadmin.capture_queue');
END;
/
```

Running these procedures performs the following actions:

- Creates a synchronous capture named `sync_capture` at the current database. A synchronous capture with the same name must not exist.
- Enables the synchronous capture. A synchronous capture cannot be disabled.
- Associates the synchronous capture with an existing queue named `capture_queue` owned by `strmadmin`.
- Creates a positive rule set for synchronous capture `sync_capture`. The rule set has a system-generated name.
- Creates a rule that captures DML changes to the `hr.employees` table and adds the rule to the positive rule set for the synchronous capture. The rule has a system-generated name.

- Prepares the hr.employees table for instantiation by running the DBMS_CAPTURE_ADM.PREPARE_SYNC_INSTANTIATION function for the table automatically.
- Creates a rule that captures DML changes to the hr.departments table and adds the rule to the positive rule set for the synchronous capture. The rule has a system-generated name.
- Prepares the hr.departments table for instantiation by running the DBMS_CAPTURE_ADM.PREPARE_SYNC_INSTANTIATION function for the table automatically.

9. Configure a synchronous capture at the sync2.example.com database:
   a. In SQL*Plus, connect to the sync2.example.com database as the Oracle Streams administrator.
   b. Run the ADD_TABLE_RULES procedure to create the synchronous capture and add a rule to instruct it to capture changes to the hr.employees table:

```
BEGIN
  DBMS_STREAMS_ADM.ADD_TABLE_RULES(
    table_name    => 'hr.employees',
    streams_type  => 'sync_capture',
    streams_name  => 'sync_capture',
    queue_name    => 'strmadmin.capture_queue');
END;
/
```

   c. Add an additional rule to the synchronous capture rule set:

```
BEGIN
  DBMS_STREAMS_ADM.ADD_TABLE_RULES(
    table_name    => 'hr.departments',
    streams_type  => 'sync_capture',
    streams_name  => 'sync_capture',
    queue_name    => 'strmadmin.capture_queue');
END;
/
```

   Step 8 describes the actions performed by these procedures at the current database.

10. Set the instantiation SCN for the tables at the sync2.example.com database:
    a. In SQL*Plus, connect to the sync1.example.com database as the Oracle Streams administrator.
    b. Set the instantiation SCN for the hr.employees table at the sync2.example.com database:

```
DECLARE
  iscn  NUMBER;    -- Variable to hold instantiation SCN value
BEGIN
  iscn := DBMS_FLASHBACK.GET_SYSTEM_CHANGE_NUMBER();
  DBMS_APPLY_ADM.SET_TABLE_INSTANTIATION_SCN@sync2.example.com(
    source_object_name    => 'hr.employees',
    source_database_name  => 'sync1.example.com',
    instantiation_scn     => iscn);
END;
/
```

c. Set the instantiation SCN for the `hr.departments` table at the `sync2.example.com` database:

```
DECLARE
  iscn  NUMBER;    -- Variable to hold instantiation SCN value
BEGIN
  iscn := DBMS_FLASHBACK.GET_SYSTEM_CHANGE_NUMBER();
  DBMS_APPLY_ADM.SET_TABLE_INSTANTIATION_SCN@sync2.example.com(
    source_object_name    => 'hr.departments',
    source_database_name  => 'sync1.example.com',
    instantiation_scn     => iscn);
END;
/
```

An instantiation SCN is the lowest SCN for which an apply process can apply changes to a table. Before the apply process can apply changes to the shared tables at the `sync2.example.com` database, an instantiation SCN must be set for each table.

11. Set the instantiation SCN for the tables at the `sync1.example.com` database:

a. In SQL*Plus, connect to the `sync2.example.com` database as the Oracle Streams administrator.

b. Set the instantiation SCN for the `hr.employees` table at the `sync1.example.com` database:

```
DECLARE
  iscn  NUMBER;    -- Variable to hold instantiation SCN value
BEGIN
  iscn := DBMS_FLASHBACK.GET_SYSTEM_CHANGE_NUMBER();
  DBMS_APPLY_ADM.SET_TABLE_INSTANTIATION_SCN@sync1.example.com(
    source_object_name    => 'hr.employees',
    source_database_name  => 'sync2.example.com',
    instantiation_scn     => iscn);
END;
/
```

c. Set the instantiation SCN for the `hr.departments` table at the `sync2.example.com` database:

```
DECLARE
  iscn  NUMBER;    -- Variable to hold instantiation SCN value
BEGIN
  iscn := DBMS_FLASHBACK.GET_SYSTEM_CHANGE_NUMBER();
  DBMS_APPLY_ADM.SET_TABLE_INSTANTIATION_SCN@sync1.example.com(
    source_object_name    => 'hr.departments',
    source_database_name  => 'sync2.example.com',
    instantiation_scn     => iscn);
END;
/
```

12. Start the apply process at each database:

a. In SQL*Plus, connect to the `sync1.example.com` database as the Oracle Streams administrator.

b. Start the apply process:

```
BEGIN
  DBMS_APPLY_ADM.START_APPLY(
    apply_name => 'apply_emp_dep');
END;
```

```
/
```

c. In SQL*Plus, connect to the `sync2.example.com` database as the Oracle Streams administrator.

d. Start the apply process:

```
BEGIN
  DBMS_APPLY_ADM.START_APPLY(
    apply_name => 'apply_emp_dep');
END;
/
```

If you would rather start the apply processes using Enterprise Manager, then see "Starting and Stopping an Apply Process" on page 5-6 for instructions.

13. Configure latest time conflict resolution for the `hr.departments` and `hr.employees` tables at the `sync1.example.com` and `sync2.example.com` databases. See "Tutorial: Configuring Latest Time Conflict Resolution for a Table" on page 4-56 for instructions.

A two-database replication environment with the following characteristics is configured:

- Each database has a synchronous capture named `sync_capture`. The synchronous capture captures all DML changes to the `hr.employees` `hr.departments` tables.
- Each database has a queue named `capture_queue`. This queue is for the synchronous capture at the database.
- Each database has an apply process named `apply_emp_dep`. The apply process applies all DML changes to the `hr.employees` table and `hr.departments` tables.
- Each database has a queue named `apply_queue`. This queue is for the apply process at the database.
- Each database has a propagation named `send_emp_dep`. The propagation sends changes from the `capture_queue` in the local database to the `apply_queue` in the other database. The propagation sends all DML changes to the `hr.employees` and `hr.departments` tables.
- Tags are used to avoid change cycling in the following way:
  - Each apply process uses the default apply tag. The default apply tag is the hexadecimal equivalent of `'00'` (double zero).
  - Each synchronous capture only captures changes in a session with a `NULL` tag. Therefore, neither synchronous capture captures the changes that are being applied by the local apply process. The synchronous capture rules instruct the synchronous capture not to capture these changes.

  See "About Tags for Avoiding Change Cycling" on page 4-10 for more information about how the replication environment avoids change cycling.

To check the Oracle Streams replication configuration:

1. At each database, complete the following steps to ensure that synchronous capture is configured:

   a. Start SQL*Plus and connect to the database as the Oracle Streams administrator.

See *Oracle Database 2 Day DBA* for more information about starting SQL*Plus.

b. Query the `ALL_SYNC_CAPTURE` data dictionary view:

```
SELECT CAPTURE_NAME FROM ALL_SYNC_CAPTURE;
```

Ensure that a synchronous capture named `sync_capture` exists at each database.

2. At each database, ensure that the propagation is enabled. To do so, follow the instructions in "Viewing Information About a Propagation" on page 5-16, and check Status on the Propagation subpage.
3. At each database, ensure that the apply process is enabled. To do so, follow the instructions in "Viewing Information About an Apply Process" on page 5-20, and check Status on the Apply subpage.

To replicate changes:

1. At one of the databases, make DML changes to the `hr.employees` table or `hr.departments` table.
2. After some time has passed to allow for replication of the changes, use SQL*Plus to query the `hr.employees` or `hr.departments` table at the other database to view the changes.

See Also:

- "About Change Capture with a Synchronous Capture" on page 4-4
- "About Hub-And-Spoke Replication Environments" on page 4-13
- "When to Replicate Data with Oracle Streams" on page 1-4
- Chapter 5, "Administering an Oracle Streams Replication Environment"

Tutorial: Configuring Latest Time Conflict Resolution for a Table

Conflict resolution automatically resolves conflicts in a replication environment. See "About Conflicts and Conflict Resolution" on page 4-9 for more information about conflict resolution.

The most common way to resolve update conflicts is to keep the change with the most recent time stamp and discard the older change. With this method, when a conflict is detected during apply, the apply process applies the change if the time-stamp column for the change is more recent than the corresponding row in the table. If the time-stamp column in the table is more recent, then the apply process discards the change.

The example in this topic configures latest time conflict resolution for the `hr.departments` table by completing the following actions:

- Adds a `time` column of the `TIMESTAMP WITH TIME ZONE` data type to the table
- Configures a trigger to update the time column in a row with the current time when the row is changed
- Adds supplemental logging for the columns in the table
- Runs the `SET_UPDATE_CONFLICT_HANDLER` procedure in the `DBMS_APPLY_ADM` package to configure conflict resolution for the table

You can use the steps in this topic to configure conflict resolution for any table. To do so, substitute your schema name for `hr` and your table name for `departments`. Also, substitute the columns in your table for the columns in the `hr.departments` table when you run the `SET_UPDATE_CONFLICT_HANDLER` procedure.

To configure latest time conflict resolution for the hr.departments table:

1. Add a `time` column to the table.

   a. In SQL*Plus, connect to the database as an administrative user, such as the Oracle Streams administrator or `SYSTEM`. Alternatively, you can connect as the user who owns the table to which the time column will be added.

      See *Oracle Database 2 Day DBA* for more information about starting SQL*Plus.

   b. Use the `ALTER TABLE` SQL statement to add the `time` column to the table. In this example, run the following statement to add the `time` column to the `hr.departments` table.

      ```
      ALTER TABLE hr.departments ADD (time TIMESTAMP WITH TIME ZONE);
      ```

2. Create a trigger to update the `time` column in each master table with the current time when a change occurs.

   Tip: Instead of using a trigger to update the `time` column, an application can populate the `time` column each time it modifies or inserts a row into a table.

   a. In Oracle Enterprise Manager, log in to the database as an administrative user, such as the Oracle Streams administrator or `SYSTEM`.

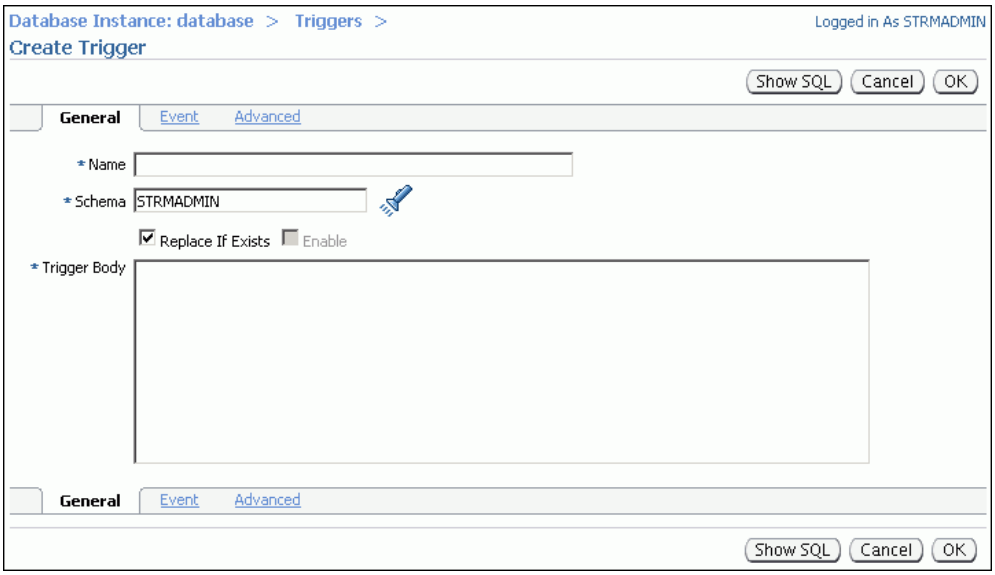
   b. Go to the Database Home page.

   c. Click **Schema** to open the Schema subpage.

   d. Click **Triggers** in the Programs section.

   e. On the Triggers page, click **Create**.

      The Create Trigger page appears, showing the General subpage.



f. Enter the name of the trigger in the **Name** field. In this example, enter `insert_departments_time`.

g. Enter the schema that owns the table in the **Schema** field. In this example, enter `hr` in the **Schema** field.

h. Enter the following in the **Trigger Body** field:

```
BEGIN
   -- Consider time synchronization problems. The previous update to this
   -- row might have originated from a site with a clock time ahead of
   -- the local clock time.
   IF :OLD.TIME IS NULL OR :OLD.TIME < SYSTIMESTAMP THEN
     :NEW.TIME := SYSTIMESTAMP;
   ELSE
     :NEW.TIME := :OLD.TIME + 1 / 86400;
   END IF;
END;
```

i. Click **Event** to open the Event subpage.

j. Ensure that **Table** is selected in the Trigger On list.

k. Enter the table name in the form *`schema.table`* in the **Table (Schema.Table)** field, or use the flashlight icon to find the database object. In this example, enter `hr.departments`.

l. Ensure that **Before** is selected for **Fire Trigger**.

m. Select **Insert** and **Update of Columns** for **Event**.

The columns in the table appear.

n. Select every column in the table except for the new `time` column.

o. Click **Advanced** to open the Advanced subpage.

p. Select **Trigger for each row**.

q. Click **OK** to create the trigger.

Note: You can also use the `CREATE TRIGGER` SQL statement to create a trigger.

---

3. In SQL*Plus, connect to the database as the Oracle Streams administrator.

   See *Oracle Database 2 Day DBA* for more information about starting SQL*Plus.

4. Add supplemental logging for the columns in the table:

```
ALTER TABLE hr.departments ADD SUPPLEMENTAL LOG DATA (ALL) COLUMNS;
```

   Supplemental logging is required for conflict resolution during apply.

5. Run the `SET_UPDATE_CONFLICT_HANDLER` procedure to configure latest time conflict resolution for the table.

   For example, run the following procedure to configure latest time conflict resolution for the `hr.departments` table:

```
DECLARE
  cols  DBMS_UTILITY.NAME_ARRAY;
```

```
BEGIN
  cols(1) := 'department_id';
  cols(2) := 'department_name';
  cols(3) := 'manager_id';
  cols(4) := 'location_id';
  cols(5) := 'time';
  DBMS_APPLY_ADM.SET_UPDATE_CONFLICT_HANDLER(
    object_name        =>  'hr.departments',
    method_name        =>  'MAXIMUM',
    resolution_column  =>  'time',
    column_list        =>  cols);
END;
/
```

Include all of the columns in the table in the `cols` column list.

6. Repeat these steps for any tables that require conflict resolution in your replication environment. You might need to configure conflict resolution for the tables at several databases.

   If you are completing an example that configures or extends a replication environment, then configure latest time conflict resolution for the appropriate tables:

   - For "Tutorial: Configuring Two-Database Replication with Local Capture Processes" on page 4-25, configure conflict resolution for all of the tables in the `hr` schema at the `db1.example.com` and `db2.example.com` databases. This schema includes the `countries`, `departments`, `employees`, `jobs`, `job_history`, `locations`, and `regions` tables.
   - For "Tutorial: Configuring Hub-and-Spoke Replication with Local Capture Processes" on page 4-40, configure conflict resolution for all of the tables in the `hr` schema at the `hub1.example.com`, `spoke1.example.com`, and `spoke2.example.com` databases. This schema includes the `countries`, `departments`, `employees`, `jobs`, `job_history`, `locations`, and `regions` tables.
   - For "Tutorial: Configuring Two-Database Replication with Synchronous Captures" on page 4-46, configure conflict resolution for the `hr.departments` and `hr.employees` tables at the `sync1.example.com` and `sync2.example.com` databases.
   - For "Tutorial: Adding Database Objects to a Replication Environment" on page 6-3, configure conflict resolution for the `oe.orders` and `oe.order_items` tables at the `hub.example.com`, `spoke1.example.com`, and `spoke2.example.com` databases.
   - For "Tutorial: Adding Databases to a Replication Environment" on page 6-7, configure conflict resolution for all of the tables in the `hr` schema at the `spoke3.example.com` database. This schema includes the `countries`, `departments`, `employees`, `jobs`, `job_history`, `locations`, and `regions` tables.

If you were directed to this section from an example, then go back to the example now.

See Also:

- "About Conflicts and Conflict Resolution" on page 4-9
- "Displaying the Configured Update Conflict Handlers" on page 5-23

5

Administering an Oracle Streams Replication Environment

This chapter describes how to manage, monitor, and troubleshoot an Oracle Streams replication environment.

This chapter contains the following sections:

- Managing an Oracle Streams Replication Environment
- Monitoring an Oracle Streams Replication Environment
- Troubleshooting an Oracle Streams Replication Environment

See Also:

- Chapter 4, "Replicating Data Using Oracle Streams"
- Chapter 6, "Extending an Oracle Streams Replication Environment"
- *Oracle Streams Concepts and Administration*
- *Oracle Streams Replication Administrator's Guide*

Managing an Oracle Streams Replication Environment

An Oracle Streams replication environment should not require much management. If the environment is configured properly, then it should replicate changes to database objects automatically with minimal administration required. This section contains instructions for performing administrative tasks that might be required from time to time in an Oracle Streams replication environment.

The following topics describe managing an Oracle Streams replication environment:

- Managing Capture Processes
- Enabling and Disabling a Propagation
- Managing Apply Processes

See Also:

- Chapter 4, "Replicating Data Using Oracle Streams"
- "Monitoring an Oracle Streams Replication Environment" on page 5-9
- "Troubleshooting an Oracle Streams Replication Environment" on page 5-26

Managing Capture Processes

You can use Enterprise Manager to manage capture processes. This topic includes instructions for completing the most common management tasks for capture processes.

The following topics describe managing capture processes:

- Starting and Stopping a Capture Process
- Setting a Capture Process Parameter

See Also:

- "About Change Capture with a Capture Process" on page 4-3
- "Monitoring Capture Processes" on page 5-12

Starting and Stopping a Capture Process

This topic contains instructions for starting and stopping a capture process in Enterprise Manager.

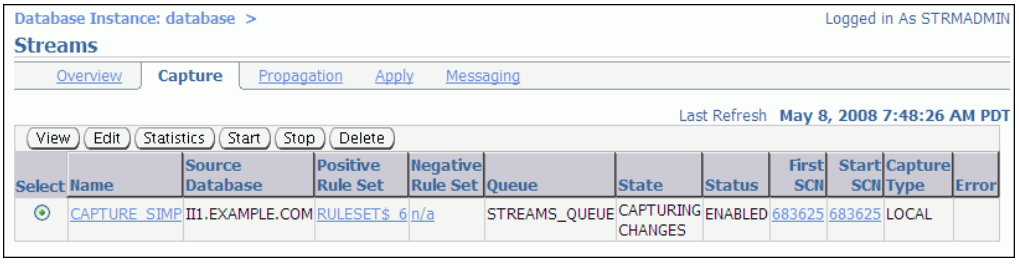
A capture process might stop automatically if it encounters an error, such as an unsupported data type. When this happens, the error is displayed on the Capture subpage of the Streams page in Enterprise Manager. In this case, you should correct the error and restart the capture process.

Also, it is important to remember that a capture process can capture changes that were made to database objects while the capture process was stopped. These changes are recorded in the redo log, and a capture process finds the changes that it is configured to capture in the redo log after it restarts. If you want to avoid capturing specific changes to database objects, then you should use tags to accomplish this. See "About Tags for Avoiding Change Cycling" on page 4-10.

To start or stop a capture process:

1. In Oracle Enterprise Manager, log in to the database as the Oracle Streams administrator.
2. Go to the Database Home page.
3. Click **Data Movement** to open the Data Movement subpage.
4. Click **Manage** in the Streams section.

   The Streams page appears, showing the Overview subpage.
5. Click **Capture** to open the Capture subpage.



6. Select the capture process that you want to start or stop.
7. Click **Start** to start a disabled or aborted capture process, or click **Stop** to stop an enabled capture process.

8. Click **Yes** on the confirmation page to finish starting or stopping the capture process.

Note: You can also use the `START_CAPTURE` procedure and `STOP_CAPTURE` procedure in the `DBMS_CAPTURE_ADM` package to start and stop a capture process.

See Also:

- "Monitoring Capture Processes" on page 5-12
- "About Change Capture with a Capture Process" on page 4-3

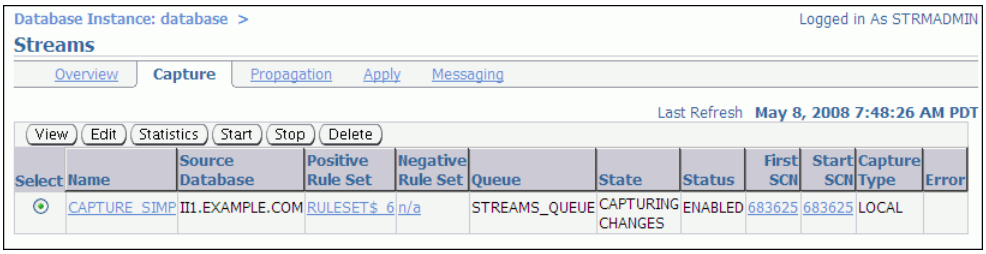
Setting a Capture Process Parameter

This topic contains instructions for setting capture process parameters in Enterprise Manager. Capture process parameters control the way a capture process operates. You can set a parameter to change a specific way in which a capture process operates. For example, you can change the `parallelism` capture process parameter to control the number of processes that capture changes. Typically, you adjust capture process parallelism to achieve the best capture process performance.

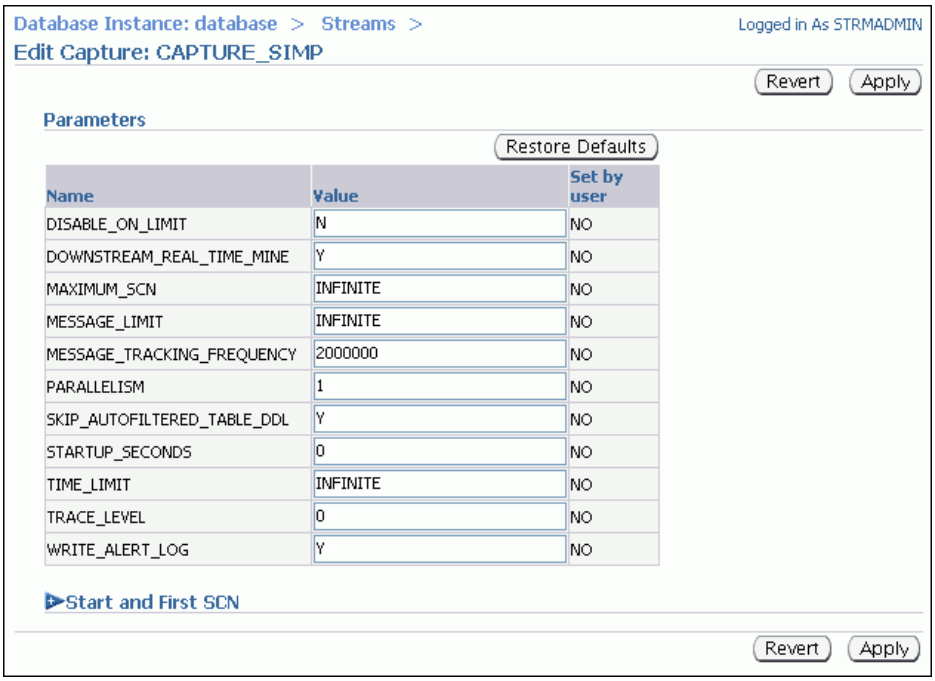
To set a capture process parameter:

1. In Oracle Enterprise Manager, log in to the database as the Oracle Streams administrator.
2. Go to the Database Home page.
3. Click **Data Movement** to open the Data Movement subpage.
4. Click **Manage** in the Streams section.

   The Streams page appears, showing the Overview subpage.
5. Click **Capture** to open the Capture subpage.



6. Select the capture process that you want to modify.
7. Click **Edit** to open the Edit Capture page.



8. Modify one or more capture process parameters in the Parameters section.

   See *Oracle Database PL/SQL Packages and Types Reference* for information about the parameters. If you change the `parallelism` parameter, then the capture process automatically stops and restarts when you click **Apply**.

9. Click **Apply** to save your changes.

Note: You can also use the `DBMS_CAPTURE_ADM.SET_PARAMETER` procedure to set a capture process parameter.

See Also:

- "Monitoring Capture Processes" on page 5-12
- "About Change Capture with a Capture Process" on page 4-3

Enabling and Disabling a Propagation

This topic contains instructions for enabling or disabling a propagation in Enterprise Manager.

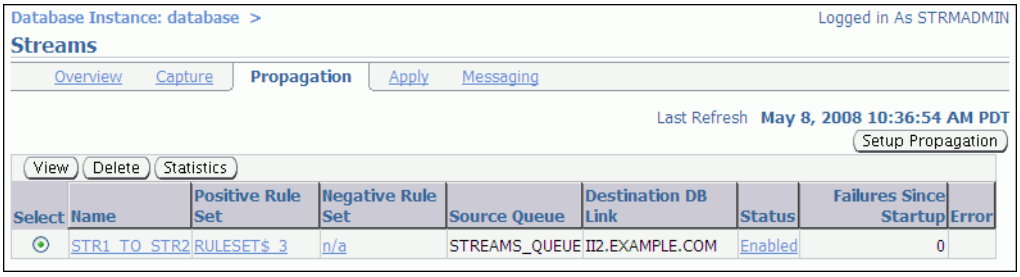
You might need to disable a propagation if the database to which the propagation sends messages goes down or if a network problem prevents the propagation from sending messages. In these situations, you can disable the propagation and enable it when the problem is corrected.

Also, a propagation becomes disabled automatically after it fails to send messages in 16 consecutive attempts. When this happens, the error is displayed on the Propagation subpage of the Streams page in Enterprise Manager. In this case, you should correct the error and enable the propagation.

To enable or disable a propagation:

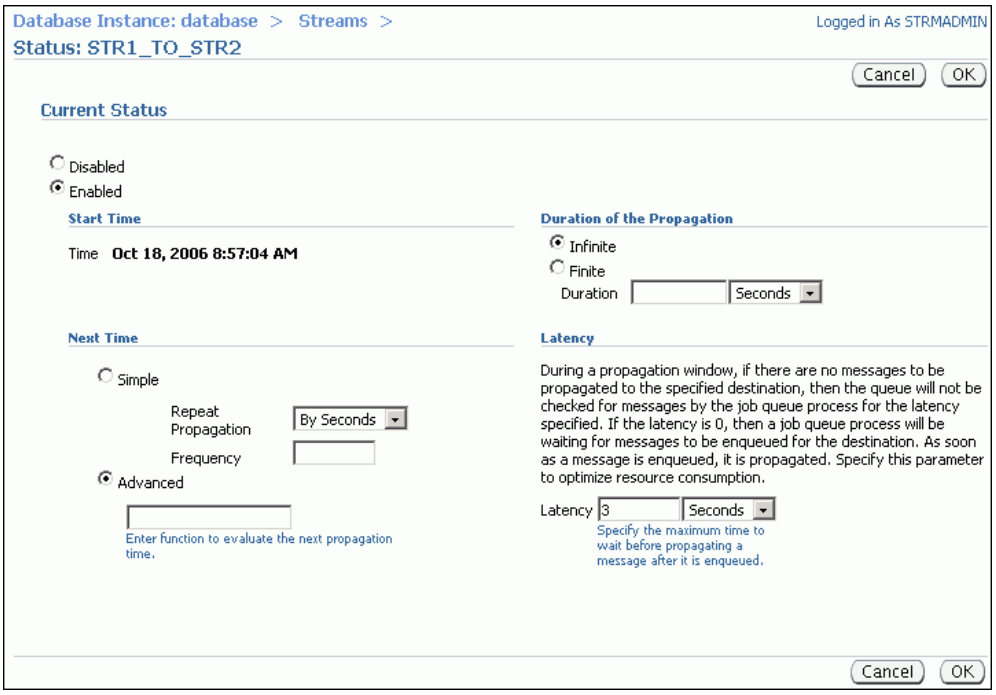
1. In Oracle Enterprise Manager, log in to the database as the Oracle Streams administrator.
2. Go to the Database Home page.
3. Click **Data Movement** to open the Data Movement subpage.
4. Click **Manage** in the Streams section.

   The Streams page appears, showing the Overview subpage.
5. Click **Propagation** to open the Propagation subpage.



6. Click the link in the **Status** field of the propagation that you want to enable or disable. The link text is either **Enabled** or **Disabled**.

   The Status page for the propagation appears.



7. Change the status to either **Enabled** or **Disabled** in the Current Status section.

   Do not modify the settings under any of the following sections: Next Time, Duration of Propagation, or Latency. Generally, the default values in these sections provide the most efficient propagation. Modifying the propagation schedule is outside the scope of this guide.
8. Click **OK** to save your changes.

Note: You can also use the `START_PROPAGATION` procedure and `STOP_PROPAGATION` procedure in the `DBMS_PROPAGATION_ADM` package to start and stop a propagation.

See Also:

- "About Change Propagation Between Databases" on page 4-5
- "Monitoring Propagations" on page 5-16
- *Oracle Streams Advanced Queuing User's Guide* for more information about managing propagations and propagation schedules

Managing Apply Processes

You can use Enterprise Manager to manage apply processes. This section includes instructions for completing the most common management tasks for apply processes.

The following topics describe managing apply processes:

- Starting and Stopping an Apply Process
- Setting an Apply Process Parameter

See Also:

- "About Change Apply" on page 4-6
- "Monitoring Apply Processes" on page 5-19
- "Managing Apply Errors" on page 5-30

Starting and Stopping an Apply Process

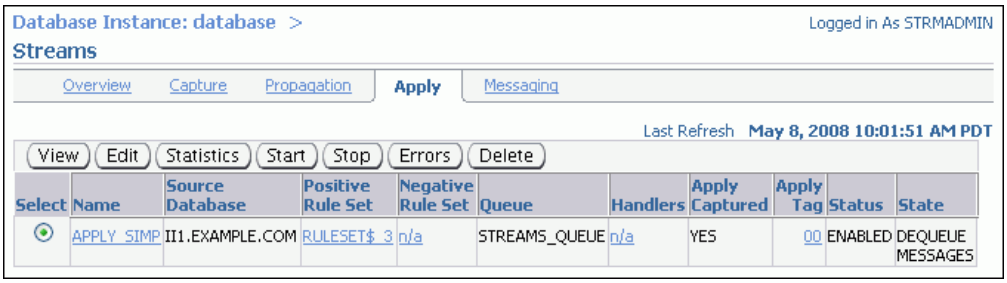
This topic contains instructions for starting and stopping an apply process in Enterprise Manager.

An apply process might stop automatically if it encounters an error, such as attempting to update a row that does not exist in a table. When this happens, the status of the apply process is `ABORTED` on the Apply subpage of the Streams page in Enterprise Manager. In this case, you should correct the error and restart the apply process.

To start or stop an apply process:

1. In Oracle Enterprise Manager, log in to the database as the Oracle Streams administrator.
2. Go to the Database Home page.
3. Click **Data Movement** to open the Data Movement subpage.
4. Click **Manage** in the Streams section.

   The Streams page appears, showing the Overview subpage.
5. Click **Apply** to open the Apply subpage.



6. Select the apply process that you want to start or stop.
7. Click **Start** to start a disabled or aborted apply process, or click **Stop** to stop an enabled apply process.
8. Click **Yes** on the confirmation page to finish starting or stopping the apply process.

Note: You can also use the `START_APPLY` procedure and `STOP_APPLY` procedure in the `DBMS_APPLY_ADM` package to start and stop an apply process.

See Also:

- "Monitoring Apply Processes" on page 5-19
- "Managing Apply Errors" on page 5-30
- "About Change Apply" on page 4-6

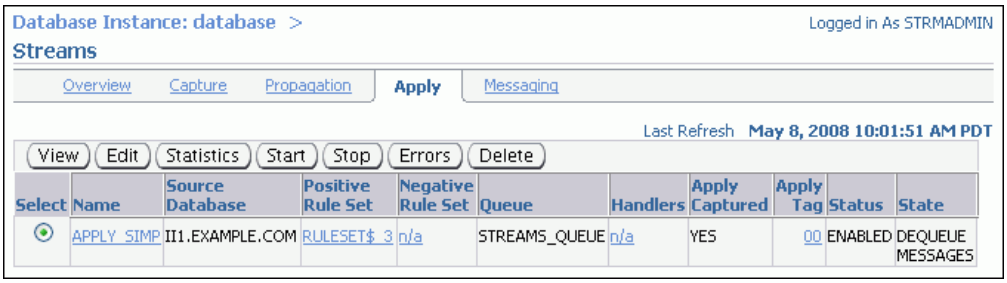
Setting an Apply Process Parameter

This topic contains instructions for setting apply process parameters in Enterprise Manager. Apply process parameters control the way an apply process operates. You can set a parameter to change a specific way in which an apply process operates. For example, you can change the `parallelism` apply process parameter to control the number of processes that apply changes. Typically, you adjust apply process parallelism to achieve the best apply process performance.

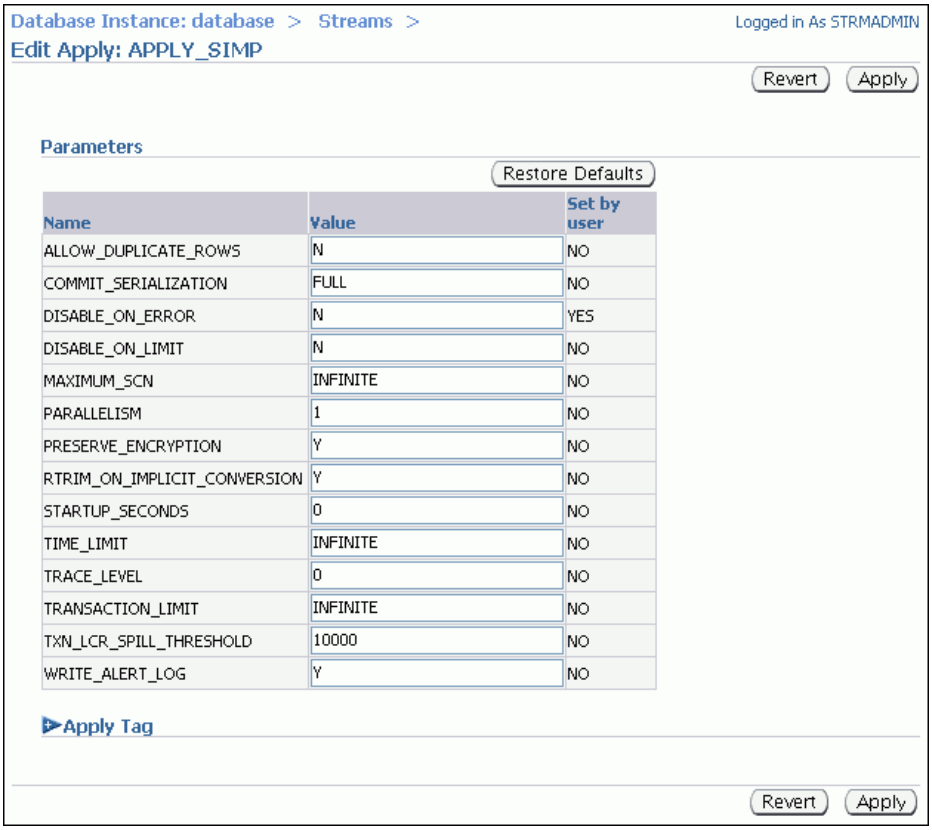
To set an apply process parameter:

1. In Oracle Enterprise Manager, log in to the database as the Oracle Streams administrator.
2. Go to the Database Home page.
3. Click **Data Movement** to open the Data Movement subpage.
4. Click **Manage** in the Streams section.

   The Streams page appears, showing the Overview subpage.
5. Click **Apply** to open the Apply subpage.



6. Select the apply process that you want to modify.
7. Click **Edit** to open the Edit Apply page.



8. Modify one or more apply process parameters in the Parameters section.

   See *Oracle Database PL/SQL Packages and Types Reference* for information about the parameters. If you change the `parallelism` parameter, then the apply process automatically stops and restarts when you click **Apply**.

9. Click **Apply** to save your changes.

Note: You can also use the `DBMS_APPLY_ADM.SET_PARAMETER` procedure to set an apply process parameter.

See Also:

- "Monitoring Apply Processes" on page 5-19
- "Managing Apply Errors" on page 5-30
- "About Change Apply" on page 4-6

Monitoring an Oracle Streams Replication Environment

This section describes using Enterprise Manager and SQL*Plus to display general information about replication components and the replication topology. It also contains instructions for monitoring capture processes, propagations, and apply processes.

The following topics describe monitoring an Oracle Streams replication environment:

- Displaying an Overview of the Replication Components at a Database
- Displaying the Topology of the Oracle Streams Replication Environment at a Database
- Monitoring Capture Processes
- Monitoring Propagations
- Monitoring Apply Processes
- Viewing Buffered Queue Statistics
- Displaying the Amount of Time Between Capture and Apply

See Also:

- "Managing an Oracle Streams Replication Environment" on page 5-1
- Chapter 4, "Replicating Data Using Oracle Streams"

Displaying an Overview of the Replication Components at a Database

The Overview subpage of the Streams page in Enterprise Manager contains information about the Oracle Streams components in the current database. This information includes:

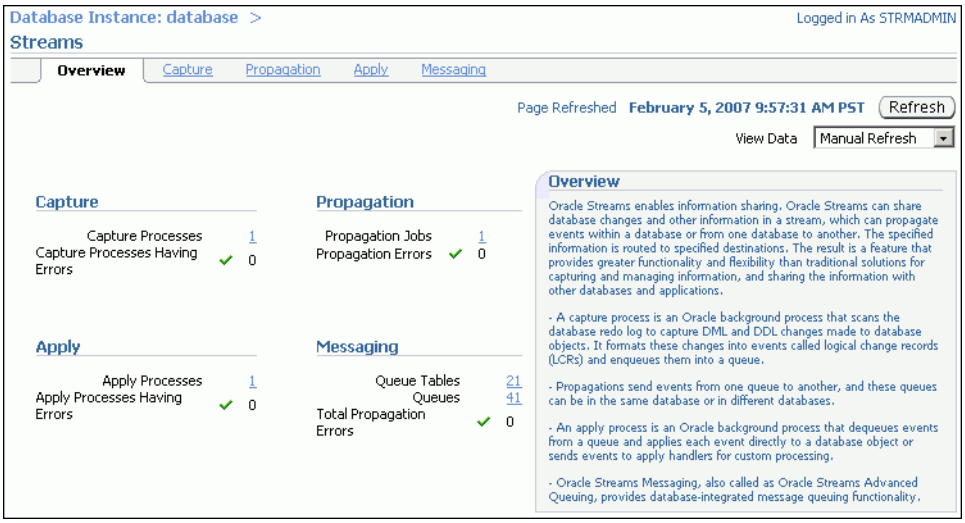
- The number of capture processes, propagations, apply processes, queues, and queue tables in the local database.
- The number of capture processes, propagations, and apply processes that currently have errors.

You can click a number to drill down to more information about a component or to manage a component.

To display an overview of the replication components at a database:

1. In Oracle Enterprise Manager, log in to the database as the Oracle Streams administrator.
2. Go to the Database Home page.
3. Click **Data Movement** to open the Data Movement subpage.
4. Click **Manage** in the Streams section.

The Streams page appears, showing the Overview subpage. Click **Help** for more information about the Oracle Streams Overview page.



See Also:

- "Displaying the Topology of the Oracle Streams Replication Environment at a Database" on page 5-10

Displaying the Topology of the Oracle Streams Replication Environment at a Database

An Oracle Streams topology displays a graphical representation of the local database and other databases that interact with the local database in the Oracle Streams environment. In a replication environment, the topology shows the following information about the databases displayed:

- The database links used by propagations from the current database to other databases in the Oracle Streams environment. Each arrow that originates at the current database shows a database link used by a propagation from the current database to another database. Replication environments use database links to send changes made to replicated objects to other databases.
- The database links used by propagations from other databases in the Oracle Streams environment to the current database for which an apply process at the current database applies the propagated messages. Each arrow that terminates at the current database shows a database link used by a propagation from another database to the current database whose messages are applied at the current database.

To view the Oracle Streams topology:

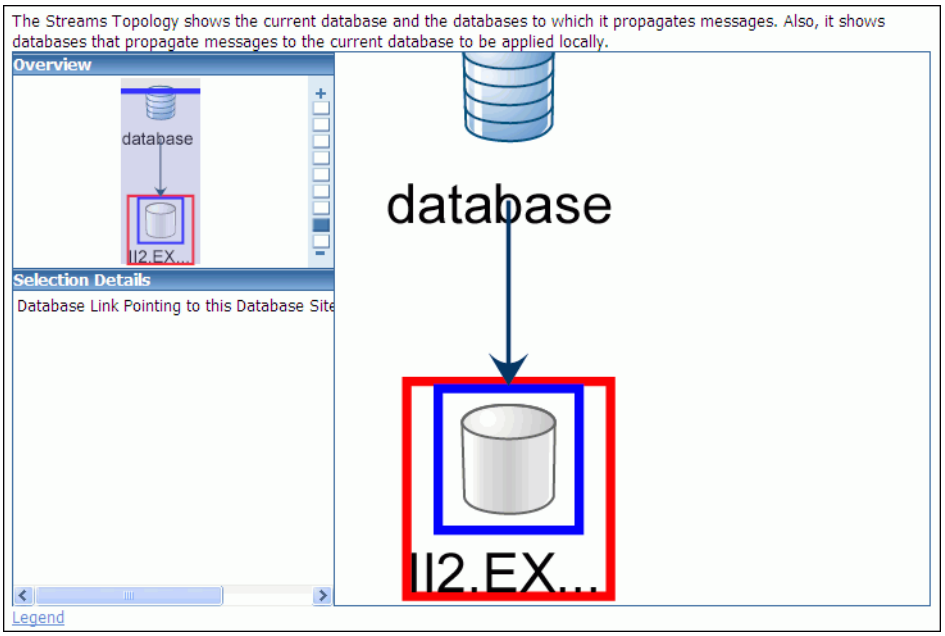
1. In Oracle Enterprise Manager, log in to the database as the Oracle Streams administrator.
2. Go to the Database Home page.
3. Click **Data Movement** to open the Data Movement subpage.
4. Click **Manage** in the Streams section.

   The Streams page appears, showing the Overview subpage.
5. Scroll down until you see the Oracle Streams topology.

You can use the **Overview** window to select a particular portion of the topology to view in detail and to zoom in and zoom out. You can also select a database in the topology and view the **Selection Details** window for information about the database.

Click **Help** or the **Legend** link for more information about the Oracle Streams topology.

The following graphic shows an example of the Oracle Streams topology for an Oracle Streams replication environment:



The Oracle Streams topology in the previous figure includes the following elements:

- The current database is named `database`.
- The `ii2.example.com` database is part of the replication environment that interacts with the current database.
- The current database sends changes to the `ii2.example.com` database, and these changes are applied by an apply process at the `ii2.example.com` database.

In addition to the graphical display of the Oracle Streams topology in Enterprise Manager, you can use the `DBMS_STREAMS_ADVISOR_ADM` package to gather information about the Oracle Streams topology. After this information is gathered, you can view the Oracle Streams topology by querying the following data dictionary views:

- `DBA_STREAMS_TP_COMPONENT` contains information about each Oracle Streams component at each database.
- `DBA_STREAMS_TP_COMPONENT_LINK` contains information about how messages flow between Oracle Streams components.
- `DBA_STREAMS_TP_COMPONENT_STAT` contains statistics about each Oracle Streams component.
- `DBA_STREAMS_TP_DATABASE` contains information about each database that contains Oracle Streams components.

- DBA_STREAMS_TP_PATH_BOTTLENECK contains information about Oracle Streams components that might be slowing down the flow of a stream.
- DBA_STREAMS_TP_PATH_STAT contains statistics about each stream that exists in the Oracle Streams topology.

When you gather information using the DBMS_STREAMS_ADVISOR_ADM package, the Oracle Streams Performance Advisor places information about Oracle Streams performance in these views. You can query these views to determine how Oracle Streams components are performing currently and for information about ways to make them perform better.

See Also:

- "Displaying an Overview of the Replication Components at a Database" on page 5-9
- *Oracle Streams Concepts and Administration* for more information about using the DBMS_STREAMS_ADVISOR_ADM package, the topology data dictionary views, and Oracle Streams Performance Advisor
- *Oracle Database Reference*

Monitoring Capture Processes

You can use Enterprise Manager to view detailed information about capture processes. You can also view statistics for capture processes.

The following topics describe monitoring capture processes:

- Viewing Information About a Capture Process
- Viewing Statistics for a Capture Process

See Also:

- "About Change Capture with a Capture Process" on page 4-3
- "Managing Capture Processes" on page 5-2

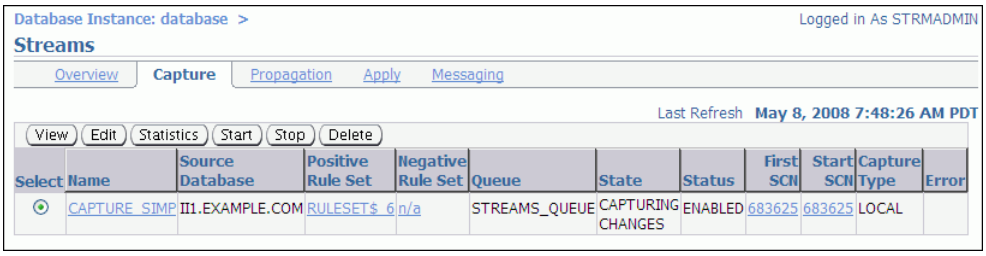
Viewing Information About a Capture Process

You can use Enterprise Manager to view information about a capture process. This information includes the capture process status and state, the rules used by the capture process, and other information about the capture process.

To view detailed information about a capture process in Enterprise Manager:

1. In Oracle Enterprise Manager, log in to the database as the Oracle Streams administrator.
2. Go to the Database Home page.
3. Click **Data Movement** to open the Data Movement subpage.
4. Click **Manage** in the Streams section.

   The Streams page appears, showing the Overview subpage.
5. Click **Capture** to open the Capture subpage.



The Capture subpage shows general information about each capture process in the database. This information includes capture process rule sets, queue, state, and status. Click **Help** for more information about the Capture subpage.

6. Select the capture process that you want to monitor.
7. Click **View** to open the View Capture Details page.

The View Capture Details page includes detailed information about the capture process. It also enables you to display the database objects for which the capture process captures changes. Rules control which database changes are captured by a capture process. Use the search tool in the Objects section to display the capture process rules:

- To display all of the rules in the positive rule set, choose **Positive Rule Type** and click **Go**. Positive rules instruct a capture process to capture changes to database objects.
- To display all of the rules in the negative rule set, choose **Negative Rule Type** and click **Go**. Negative rules instruct a capture process not to capture changes to database objects.

Click **Help** for more information about this page.

Note: You can also query the `ALL_CAPTURE` data dictionary view for information about a capture process.

See Also:

- "About Change Capture with a Capture Process" on page 4-3
- "About Rules for Controlling the Behavior of Capture, Propagation, and Apply" on page 4-7
- "Managing Capture Processes" on page 5-2

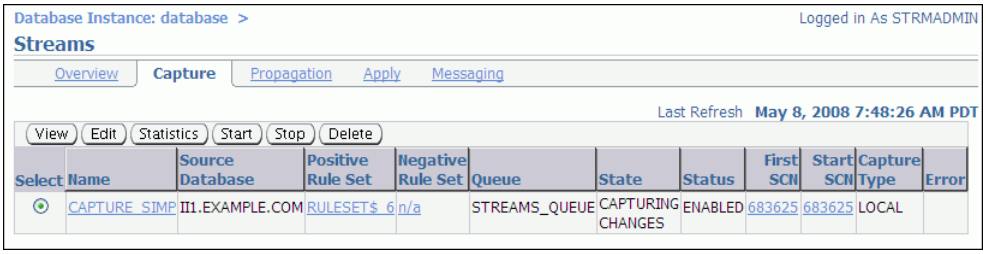
Viewing Statistics for a Capture Process

You can use Enterprise Manager to view statistics for a capture process. The statistics include the number of messages in the capture process queue, the number of messages captured and enqueued by the capture process, and other statistics relating to the capture process.

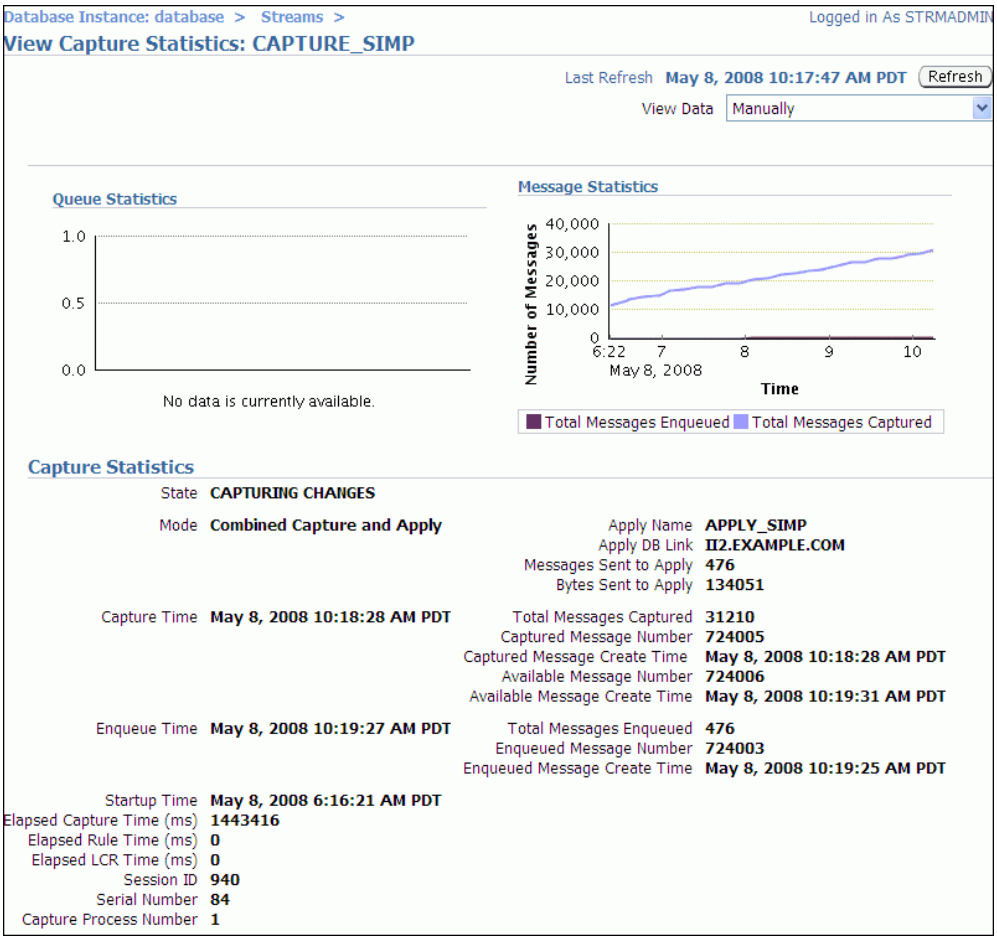
To view statistics for a capture process in Enterprise Manager:

1. In Oracle Enterprise Manager, log in to the database as the Oracle Streams administrator.
2. Go to the Database Home page.
3. Click **Data Movement** to open the Data Movement subpage.
4. Click **Manage** in the Streams section.

   The Streams page appears, showing the Overview subpage.
5. Click **Capture** to open the Capture subpage.



   The Capture subpage shows general information about each capture process in the database. This information includes capture process rule sets, queue, state, and status. Click **Help** for more information about the Capture subpage.
6. Select the capture process that you want to monitor.
7. Click **Statistics** to open the View Capture Statistics page.



The View Capture Statistics page includes the following information:

- The **Queue Statistics** graph shows the number of messages currently in the capture process queue. The **No of Enqueued Messages** line in the graph shows the total number of messages currently in the buffered queue. The **No of Spilled Messages** line in the graph shows the total number of messages that have spilled from memory into the persistent queue table.
- The **Message Statistics** graph shows the total number of changes enqueued and captured by the capture process since it last started. **Total Messages Enqueued** shows the number of changes enqueued by the capture process. **Total Messages Captured** shows the number of changes that were evaluated in detail against the capture process rules. If a change does not satisfy the capture process rules, then the change is not enqueued.
- The **Capture Statistics** section includes the current state of the capture process, as well as statistics related to time and changes (messages) captured. See the documentation for the `V$STREAMS_CAPTURE` dynamic performance view in *Oracle Database Reference* for more information about these statistics.

Click **Help** for more information about this page.

See Also:

- "About Change Capture with a Capture Process" on page 4-3
- "Managing Capture Processes" on page 5-2

Monitoring Propagations

You can use Enterprise Manager to view detailed information about propagations. You can also view statistics for propagations.

The following topics describe monitoring propagations:

- Viewing Information About a Propagation
- Viewing Statistics for a Propagation

See Also:

- "About Change Propagation Between Databases" on page 4-5
- "Enabling and Disabling a Propagation" on page 5-4

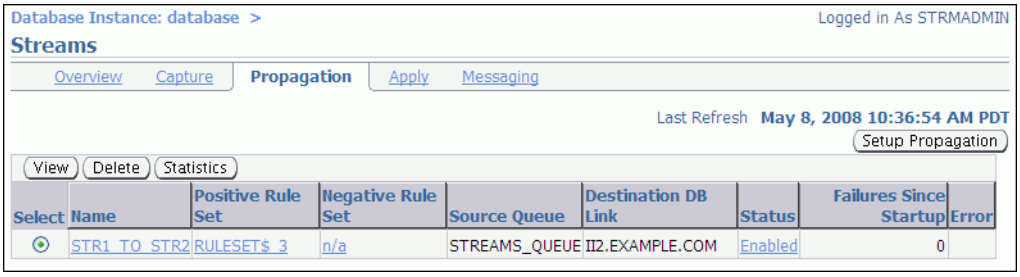
Viewing Information About a Propagation

You can use Enterprise Manager to view information about a propagation. This information includes the propagation status, whether the propagation has failed, the rules used by the propagation, and other information about the propagation.

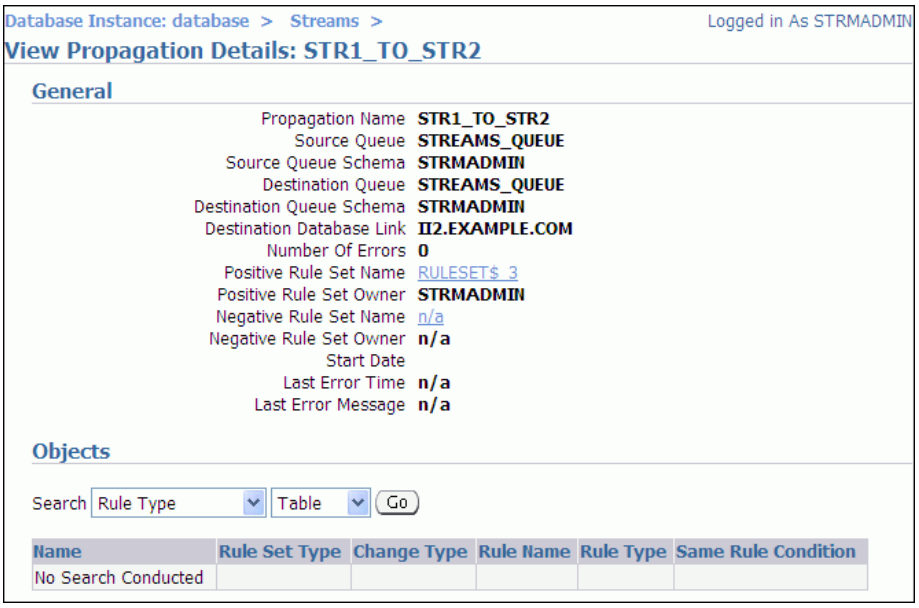
To view information about a propagation in Enterprise Manager:

1. In Oracle Enterprise Manager, log in to the database as the Oracle Streams administrator.
2. Go to the Database Home page.
3. Click **Data Movement** to open the Data Movement subpage.
4. Click **Manage** in the Streams section.

   The Streams page appears, showing the Overview subpage.
5. Click **Propagation** to open the Propagation subpage.



6. Select the propagation that you want to monitor.
7. Click **View** to open the View Propagation Details page for the propagation.



The View Propagation Details page includes detailed information about the propagation. It also enables you to display the database objects for which the propagation sends changes. Rules control which database changes are sent by a propagation. Use the search tool in the Objects section to display the propagation rules:

- To display all of the rules in the positive rule set, choose **Positive Rule Type** and click **Go**. Positive rules instruct a propagation to send changes to database objects to the destination queue.
- To display all of the rules in the negative rule set, choose **Negative Rule Type** and click **Go**. Negative rules instruct a propagation not to send changes to database objects to the destination queue.

Click **Help** for more information about this page.

Note: You can also query the `ALL_PROPAGATION` data dictionary view for information about a propagation.

See Also:

- "About Change Propagation Between Databases" on page 4-5
- "About Rules for Controlling the Behavior of Capture, Propagation, and Apply" on page 4-7
- "Enabling and Disabling a Propagation" on page 5-4

Viewing Statistics for a Propagation

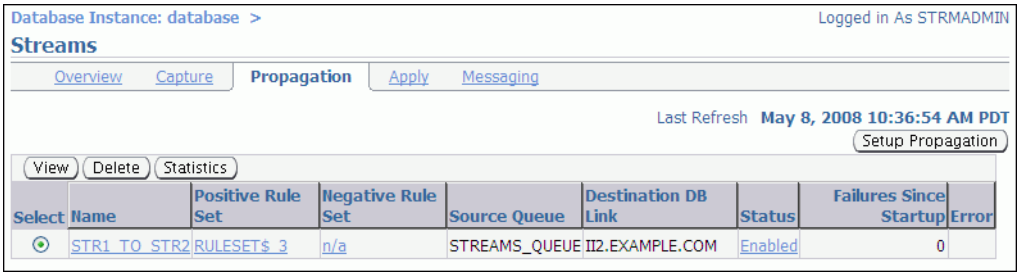
You can use Enterprise Manager to view statistics for a propagation. The statistics include the number of messages in the propagation source queue, the number of messages sent by the propagation, and other statistics relating to the propagation.

Note: Propagation statistics are not calculated when combined capture and apply is used.

To view statistics for a propagation in Enterprise Manager:

1. In Oracle Enterprise Manager, log in to the database as the Oracle Streams administrator.
2. Go to the Database Home page.
3. Click **Data Movement** to open the Data Movement subpage.
4. Click **Manage** in the Streams section.

   The Streams page appears, showing the Overview subpage.
5. Click **Propagation** to open the Propagation subpage.

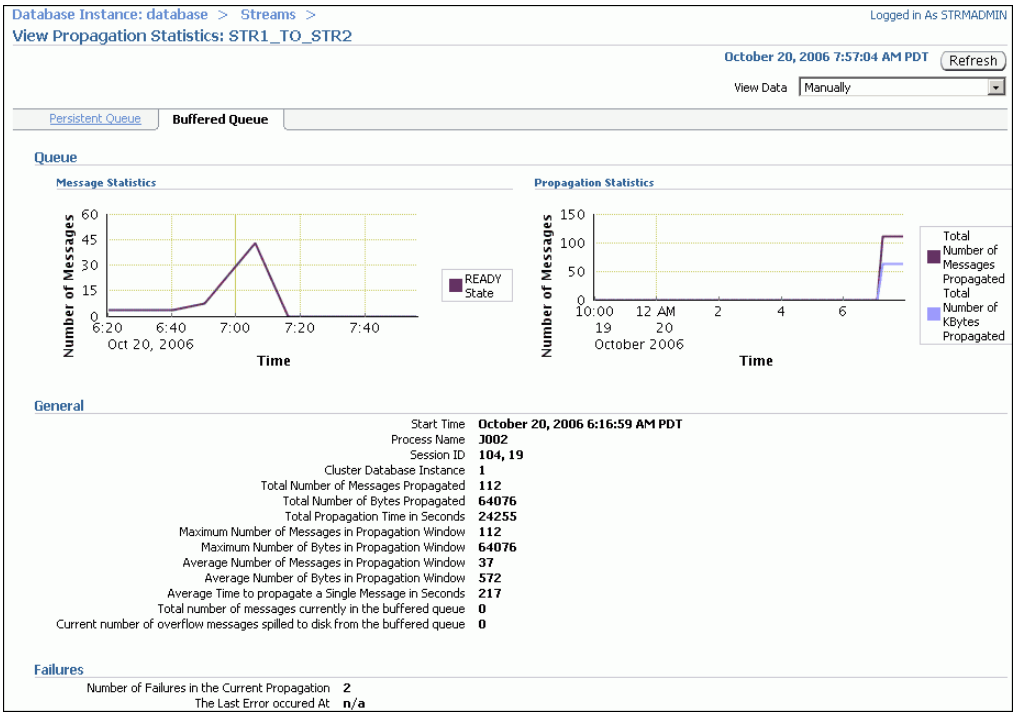


6. Select the propagation that you want to monitor.
7. Click **Statistics** to open the View Propagation Statistics page for the propagation.

The View Propagation Statistics page includes the following subpages:

- The Persistent Queue subpage shows propagation statistics for messages that were enqueued into the persistent queue portion of the queue used by the propagation. Use this subpage to view propagation statistics if the propagation sends changes captured by a synchronous capture.
- The Buffered Queue subpage shows propagation statistics for messages that were enqueued into the buffered queue portion of the queue used by the propagation. Use this subpage to view propagation statistics if the propagation sends changes captured by a capture process.

Both subpages contain graphs that show the number of messages in the queue and the number of messages sent by the propagation over several hours. Both subpages also contain other propagation statistics, such as the total number of messages and bytes propagated since the propagation was last started. Click **Help** for more information about the statistics on the current subpage.



Note: You can also query the following dynamic performance views for buffered queue statistics and propagation statistics:

- `V$BUFFERED_QUEUES`
- `V$PROPAGATION_SENDER`
- `V$PROPAGATION_RECEIVER`

See Also:

- "About Change Propagation Between Databases" on page 4-5
- "Enabling and Disabling a Propagation" on page 5-4
- *Oracle Streams Concepts and Administration* for information about combined capture and apply

Monitoring Apply Processes

You can use Enterprise Manager to view detailed information about apply processes. You can also view statistics for apply processes.

The following topics describe monitoring apply processes:

- Viewing Information About an Apply Process
- Viewing Statistics for an Apply Process
- Displaying the Configured Update Conflict Handlers

See Also:

- "About Change Apply" on page 4-6
- "Managing Apply Processes" on page 5-6
- "Managing Apply Errors" on page 5-30

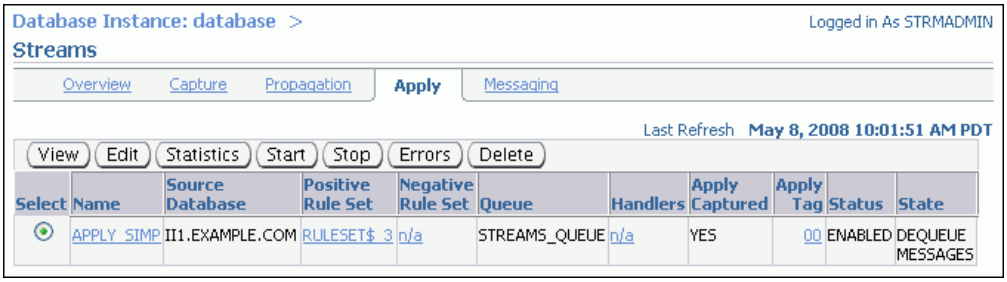
Viewing Information About an Apply Process

You can use Enterprise Manager to view information about an apply process. This information includes the apply process status and state, the rules used by the apply process, and other information about the apply process.

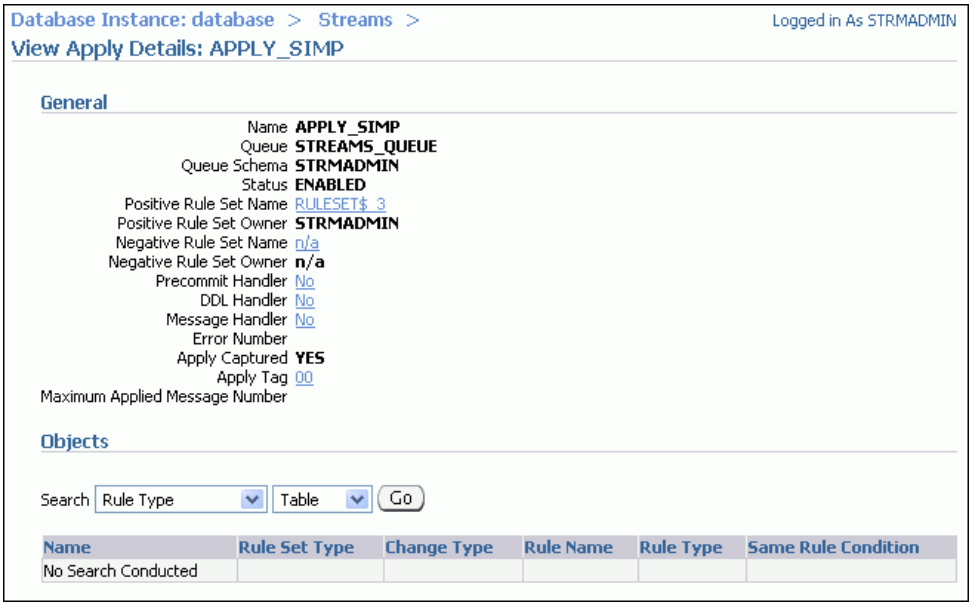
To view information about an apply process in Enterprise Manager:

1. In Oracle Enterprise Manager, log in to the database as the Oracle Streams administrator.
2. Go to the Database Home page.
3. Click **Data Movement** to open the Data Movement subpage.
4. Click **Manage** in the Streams section.

   The Streams page appears, showing the Overview subpage.
5. Click **Apply** to open the Apply subpage.



6. Select the apply process that you want to monitor.
7. Click **View** to open the View Apply Details page.



The View Apply Details page includes detailed information about the apply process. It also enables you to display the database objects for which the apply process applies changes. Rules control which database changes are dequeued and applied by an apply process. Use the search tool in the Objects section to display the apply process rules:

- To display all of the rules in the positive rule set, choose **Positive Rule Type** and click **Go**. Positive rules instruct an apply process to dequeue and apply changes to database objects.
- To display all of the rules in the negative rule set, choose **Negative Rule Type** and click **Go**. Negative rules instruct an apply process not to dequeue changes to database objects.

Click **Help** for more information about this page.

Note: You can also query the `ALL_APPLY` data dictionary view for information about an apply process.

See Also:

- "About Change Apply" on page 4-6
- "About Rules for Controlling the Behavior of Capture, Propagation, and Apply" on page 4-7
- "Managing Apply Processes" on page 5-6

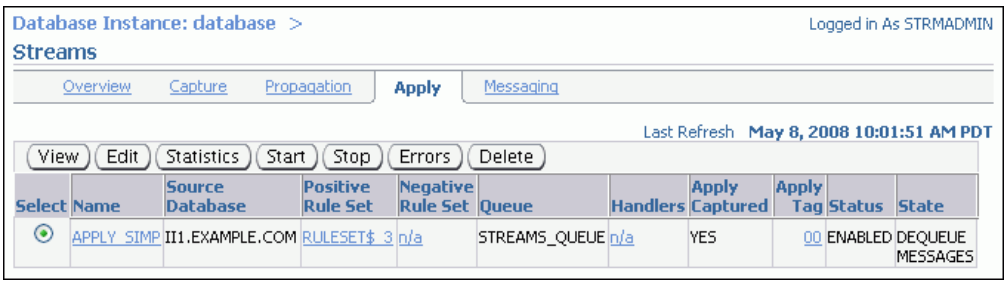
Viewing Statistics for an Apply Process

You can use Enterprise Manager to view statistics for an apply process. The statistics include the number of messages in the apply process queue, the number of messages applied by the apply process, and other statistics relating to the apply process.

To view statistics for an apply process in Enterprise Manager:

1. In Oracle Enterprise Manager, log in to the database as the Oracle Streams administrator.
2. Go to the Database Home page.
3. Click **Data Movement** to open the Data Movement subpage.
4. Click **Manage** in the Streams section.

   The Streams page appears, showing the Overview subpage.
5. Click **Apply** to open the Apply subpage.

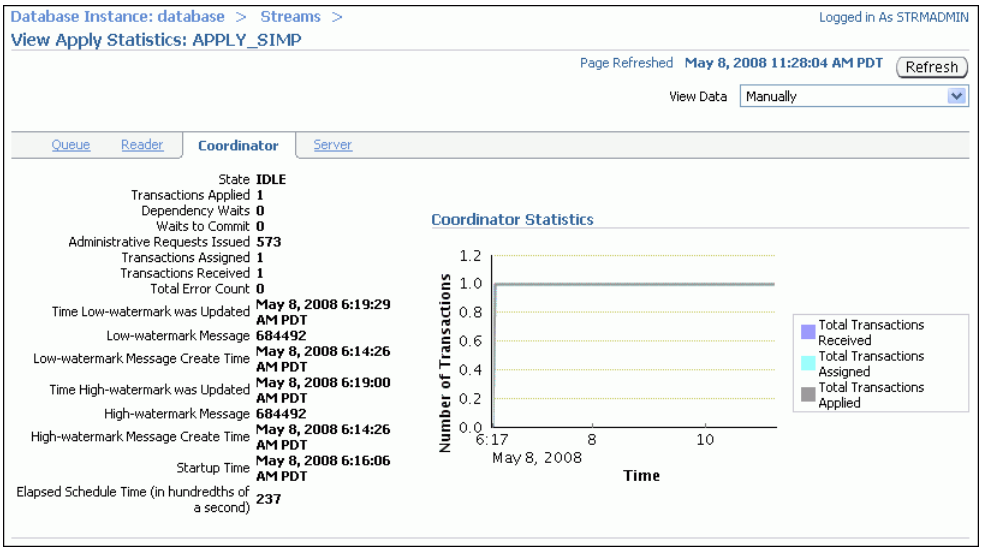


6. Select the apply process that you want to monitor.
7. Click **Statistics** to open the View Apply Statistics page.

The View Apply Statistics page includes the following subpages:

- The Queue subpage shows the number of messages over the past several hours in both the persistent queue portion and the buffered queue portion of the apply process queue. If the apply process applies changes captured by a synchronous capture, then analyze the persistent queue statistics. If the apply process applies changes captured by a capture process, then analyze the buffered queue statistics.
- The Reader subpage shows statistics for the reader server. The reader server dequeues messages from the apply process queue and assembles them into separate transactions.
- The Coordinator subpage shows statistics for the coordinator process. The coordinator process gets transactions from the reader server and passes them to apply servers.
- The Server subpage shows statistics for the apply servers. The apply servers apply changes to database objects or pass the changes to apply handlers. To view details about a specific apply server, select it and click **View Details**.

Click **Help** for more information about the statistics on the current subpage.



Note: You can also query the following dynamic performance views for buffered queue statistics and apply process statistics:

- `V$BUFFERED_QUEUES`
- `V$STREAMS_APPLY_COORDINATOR`
- `V$STREAMS_APPLY_READER`
- `V$STREAMS_APPLY_SERVER`

See Also:

- "About Change Apply" on page 4-6
- "Managing Apply Processes" on page 5-6

Displaying the Configured Update Conflict Handlers

In a replication environment, update conflict handlers automatically resolve conflicts that occur when the same row is updated at two different databases at nearly the same time. You can use the ALL_APPLY_CONFLICT_COLUMNS data dictionary view to list the update conflict handlers.

To display the configured update conflict handlers:

1. Open SQL*Plus and connect to the database as the Oracle Streams administrator.

   See *Oracle Database 2 Day DBA* for more information about starting SQL*Plus.

2. Run the following query:

```
COLUMN OBJECT_OWNER HEADING 'Table|Owner' FORMAT A5
COLUMN OBJECT_NAME HEADING 'Table Name' FORMAT A12
COLUMN METHOD_NAME HEADING 'Method' FORMAT A12
COLUMN RESOLUTION_COLUMN HEADING 'Resolution|Column' FORMAT A13
COLUMN COLUMN_NAME HEADING 'Column Name' FORMAT A30

SELECT OBJECT_OWNER,
       OBJECT_NAME,
       METHOD_NAME,
       RESOLUTION_COLUMN,
       COLUMN_NAME
  FROM ALL_APPLY_CONFLICT_COLUMNS
  ORDER BY OBJECT_OWNER, OBJECT_NAME, RESOLUTION_COLUMN;
```

   The output will be similar to the following:

```
Table                          Resolution
Owner Table Name   Method      Column       Column Name
----- ------------ ----------- ------------ ------------------------------
HR    COUNTRIES    MAXIMUM     TIME         TIME
HR    COUNTRIES    MAXIMUM     TIME         REGION_ID
HR    COUNTRIES    MAXIMUM     TIME         COUNTRY_NAME
HR    DEPARTMENTS  MAXIMUM     TIME         TIME
HR    DEPARTMENTS  MAXIMUM     TIME         MANAGER_ID
HR    DEPARTMENTS  MAXIMUM     TIME         LOCATION_ID
HR    DEPARTMENTS  MAXIMUM     TIME         DEPARTMENT_NAME
```

   The output in this example shows that latest time conflict resolution is configured for the hr.countries and hr.departments tables. When a conflict occurs for any column listed under Column Name, the change with the maximum, or latest, time in the TIME resolution column is used, and the other change is discarded.

See Also:

- "About Conflicts and Conflict Resolution" on page 4-9
- "Tutorial: Configuring Latest Time Conflict Resolution for a Table" on page 4-56

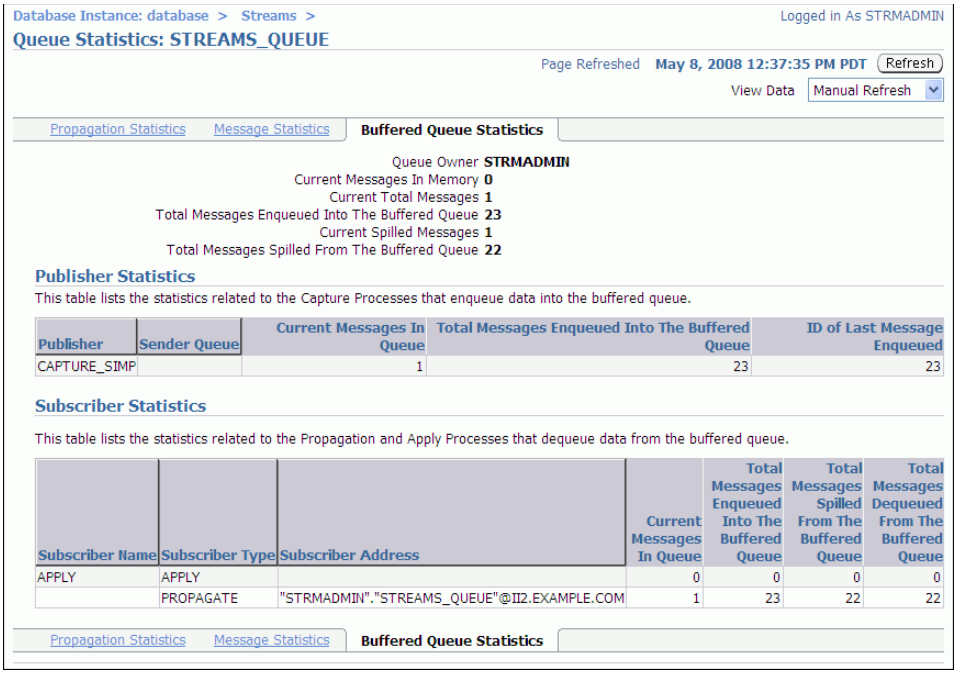
Viewing Buffered Queue Statistics

You can use Enterprise Manager to view statistics for a buffered queue that is used by a capture process, propagation, or apply process in your replication environment. In an Oracle Streams replication environment that uses capture processes to capture changes, each capture process enqueues changes into the buffered queue portion of its queue. The changes remain in buffered queues as they are propagated from one queue to another, and apply processes dequeue the changes from a buffered queue.

To view buffered queue statistics:

1. In Oracle Enterprise Manager, log in to the database as the Oracle Streams administrator.
2. Go to the Database Home page.
3. Click **Data Movement** to open the Data Movement subpage.
4. Click **Manage** in the Streams section.

   The Streams page appears, showing the Overview subpage.
5. Click **Messaging** to open the Messaging subpage.
6. Select the queue that you want to monitor.
7. Select **Queue Statistics** in the Actions list.
8. Click **Go** to open the Queue Statistics page.
9. Click **Buffered Queue Statistics** to open the Buffered Queue Statistics subpage.



This Buffered Queue Statistics subpage includes the following information about the buffered queue portion of the selected queue:

- Information about the number of changes in the queue and the number of changes that have spilled from memory onto disk.
- The number of changes enqueued by local capture processes. When the capture process is local, the **Sender Queue** field is empty.
- The number of changes captured by remote capture processes at a different database and sent to the buffered queue. When a capture process is remote, the **Sender Queue** field contains the queue and database from which the changes were sent.
- The number of changes dequeued by local apply processes from the buffered queue.

- The number of changes sent by propagations from the buffered queue to a different queue.

Click **Help** for more information about the statistics on this subpage.

Note: You can also query the following dynamic performance views for buffered queue statistics:

- V$BUFFERED_QUEUES
- V$BUFFERED_PUBLISHERS
- V$BUFFERED_SUBSCRIBERS

See Also:

- "About Oracle Streams Replication" on page 4-1
- "Modifying Queues" on page 9-24

Displaying the Amount of Time Between Capture and Apply

You can query the V$STREAMS_APPLY_COORDINATOR dynamic performance view to monitor the performance of an Oracle Streams replication environment. Specifically, you can query this view to determine the amount of time between when a change was captured at the source database and when it was applied by an apply process at the current database. The query shows the amount of time for a recent change that was applied by each apply process in the current database. In an Oracle Streams replication environment, the amount of time between capture and apply should be relatively short.

Some changes might take longer than others to be captured, propagated, and applied. The query shows the message number of the change you are monitoring, and each change has a different message number. Therefore, you can run this query multiple times to view the amount of time between capture and apply for different changes.

To determine the amount of time between capture and apply for a recent change:

1. Open SQL*Plus and connect to the database running the apply process as the Oracle Streams administrator.

   See *Oracle Database 2 Day DBA* for more information about starting SQL*Plus.

2. Run the following query:

```
COLUMN APPLY_NAME HEADING 'Apply Process Name' FORMAT A30
COLUMN 'Time Between Capture and Apply' FORMAT 999999
COLUMN HWM_MESSAGE_NUMBER HEADING 'Applied Message Number' FORMAT 999999999

SELECT APPLY_NAME,
      (HWM_TIME-HWM_MESSAGE_CREATE_TIME)*86400 "Time in Seconds",
       HWM_MESSAGE_NUMBER
  FROM V$STREAMS_APPLY_COORDINATOR;
```

   The output will be similar to the following:

```
Apply Process Name             Time in Seconds Applied Message Number
------------------------------ --------------- ----------------------
APPLY_SIMP                                   3                1486062
```

See Also:

- "About Oracle Streams Replication" on page 4-1
- "Managing an Oracle Streams Replication Environment" on page 5-1

Troubleshooting an Oracle Streams Replication Environment

This section describes the most common problems in an Oracle Streams replication environment. It also describes how to correct these problems.

The following topics describe troubleshooting an Oracle Streams replication environment:

- Responding to Automated Alerts in Enterprise Manager
- Managing Apply Errors
- Managing a Replication Environment When a Destination Is Unavailable

See Also:

- Chapter 4, "Replicating Data Using Oracle Streams"
- Chapter 5, "Administering an Oracle Streams Replication Environment"
- *Oracle Streams Concepts and Administration* for more information about troubleshooting Oracle Streams
- *Oracle Streams Replication Administrator's Guide* for more information about troubleshooting Oracle Streams replication environments

Responding to Automated Alerts in Enterprise Manager

An **alert** is a warning about a potential problem or an indication that a critical threshold has been crossed. There are two types of alerts:

- **Stateless:** Alerts that indicate single events that are not necessarily tied to the system state. For example, an alert that indicates that a capture aborted with a specific error is a stateless alert.
- **Stateful:** Alerts that are associated with a specific system state. Stateful alerts are usually based on a numeric value, with thresholds defined at warning and critical levels. For example, an alert on the current Oracle Streams pool memory usage percentage, with the warning level at 85% and the critical level at 95%, is a stateful alert.

To view the alerts for a database:

1. Log in to Enterprise Manager as an administrative user, such as the Oracle Streams administrator.
2. Go to the Database Home page of the database you want to manage.
3. View the alerts for the database in the Alerts section. You might need to scroll down to see the Alerts section.

Oracle Streams has its own set of alerts. Table 5–1 describes the alerts related to Oracle Streams.

Table 5–1 Oracle Streams Alerts

Alert	Message	Alert Type
Capture Aborts Alert	STREAMS capture process *capture_name* aborted with ORA-*error_number*	Stateless
Propagation Aborts Alert	STREAMS propagation process *source_queue, destination_queue, database_link* aborted after 16 failures	Stateless
Apply Aborts Alert	STREAMS apply process *apply_name* aborted with ORA-*error_number*	Stateless
Apply Error Alert	STREAMS error queue for apply process *apply_name* contains new transaction with ORA-*error_number*	Stateless
Oracle Streams Pool Alert	*Automatically set by alert infrastructure*	Stateful

Note: Oracle Streams alerts are informational only. They do not need to be managed. If you monitor your Oracle Streams environment regularly and address problems as they arise, then you might not need to monitor Oracle Streams alerts.

See Also:

- "Troubleshooting an Oracle Streams Replication Environment" on page 5-26
- *Oracle Streams Concepts and Administration*
- *Oracle Streams Replication Administrator's Guide*

Capture Aborts Alert

This alert indicates a critical error. The capture process stops and any replication that depends on the capture process also stops. Also, the capture process makes no further progress in scanning the redo log until it is restarted.

Response

Obtain the exact error message in one of the following ways:

- In Enterprise Manager, see "Viewing Information About a Capture Process" on page 5-12 for instructions.
- In SQL*Plus, query the `ALL_CAPTURE` view.

Take the appropriate action for the error.

After taking the appropriate action, restart the capture process in one of the following ways:

- In Enterprise Manager, see "Starting and Stopping a Capture Process" on page 5-2 for instructions.
- In SQL*Plus, run the `DBMS_CAPTURE_ADM.START_CAPTURE` procedure.

See Also:

- "Responding to Automated Alerts in Enterprise Manager" on page 5-26
- "About Change Capture with a Capture Process" on page 4-3
- "Managing Capture Processes" on page 5-2
- "Monitoring Capture Processes" on page 5-12
- *Oracle Streams Concepts and Administration*
- *Oracle Streams Replication Administrator's Guide*

Propagation Aborts Alert

This alert indicates a critical error. The propagation stops, and the messages that are normally sent from one queue to another by the propagation remain in the source queue. Replication that depends on the propagation also stops. Eventually, the source queue can grow too large, and messages can spill to disk. When messages spill to disk, it degrades Oracle Streams performance.

Response

Obtain the exact error message in one of the following ways:

- In Enterprise Manager, see "Viewing Information About a Propagation" on page 5-16 for instructions.
- In SQL*Plus, query the `DBA_QUEUE_SCHEDULES` view.

Take the appropriate action for the error.

After taking the appropriate action, restart the propagation in one of the following ways:

- In Enterprise Manager, see "Enabling and Disabling a Propagation" on page 5-4 for instructions.
- In SQL*Plus, run the `DBMS_PROPAGATION_ADM.START_PROPAGATION` procedure.

See Also:

- "Responding to Automated Alerts in Enterprise Manager" on page 5-26
- "About Change Propagation Between Databases" on page 4-5
- "Monitoring Propagations" on page 5-16
- *Oracle Streams Concepts and Administration*
- *Oracle Streams Replication Administrator's Guide*

Apply Aborts Alert

This alert indicates a critical error. The apply process stops, and the messages that are normally dequeued by the apply process remain in the apply process queue. Replication that depends on the apply process also stops. Eventually, the apply process queue can grow too large, and messages can spill to disk. Other queues that send messages to the apply process queue might also grow and spill messages to disk. When messages spill to disk, it degrades Oracle Streams performance.

Response

Obtain the exact error message in one of the following ways:

- In Enterprise Manager, see "Viewing Information About an Apply Process" on page 5-20 for instructions.
- In SQL*Plus, query the `ALL_APPLY` view.

Take the appropriate action for the error. If the error is an `ORA-26714` error, then consider setting the `DISABLE_ON_ERROR` apply process parameter to `N` to avoid aborting on future user errors. See "Setting an Apply Process Parameter" on page 5-7 for instructions.

After taking the appropriate action, restart the apply process in one of the following ways:

- In Enterprise Manager, see "Starting and Stopping an Apply Process" on page 5-6 for instructions.
- In SQL*Plus, run the `DBMS_APPLY_ADM.START_APPLY` procedure.

See Also:

- "Responding to Automated Alerts in Enterprise Manager" on page 5-26
- *Oracle Streams Concepts and Administration*
- *Oracle Streams Replication Administrator's Guide*

Apply Error Alert

This alert indicates that the apply process encountered an error when it was applying a transaction. The apply process moves all of the messages in the transaction to the error queue. Other transactions that depend on the error transaction might also result in apply errors, and the error queue might grow quickly. Therefore, an administrator should resolve the apply errors as soon as possible.

Response

Manage the apply errors in the error queue in one of the following ways:

- In Enterprise Manager, see "Managing Apply Errors" on page 5-30 for instructions.
- In SQL*Plus, query the `ALL_APPLY_ERROR` view to view the errors, resolve the errors, and either execute or delete the errors using procedures in the `DBMS_APPLY_ADM` package.

See Also:

- "Responding to Automated Alerts in Enterprise Manager" on page 5-26
- "About Change Apply" on page 4-6
- "Managing Apply Processes" on page 5-6
- "Monitoring Apply Processes" on page 5-19
- *Oracle Streams Concepts and Administration*
- *Oracle Streams Replication Administrator's Guide*

Oracle Streams Pool Alert

A metric is a unit of measurement used to report the health of the system. This alert is generated when the memory usage of the Oracle Streams pool has exceeded the percentage specified by the `STREAMS_POOL_USED_PCT` metric.

This alert can be raised only if the database is not using Automatic Memory Management or Automatic Shared Memory Management. Specifically, this alert can only be raised if the following initialization parameters are set to `0` (zero) or are unset:

- `MEMORY_TARGET`
- `SGA_TARGET`

Note: Oracle recommends using either Automatic Memory Management or Automatic Shared Memory Management.

Response

This metric threshold can be set automatically by the alert infrastructure, or you can set it using Enterprise Manager. If the currently running workload is typical, then consider increasing the size of the Oracle Streams pool. Some of the links under the Related Topics heading on the Database Home page enable you to manage metrics. See *Oracle Database 2 Day DBA* for information about viewing and setting metric thresholds.

See Also:

- "Responding to Automated Alerts in Enterprise Manager" on page 5-26
- "Preparing for Oracle Streams Replication" on page 4-21
- *Oracle Streams Concepts and Administration*

Managing Apply Errors

Apply errors can occur for a variety of reasons. When a change in a transaction causes an apply error, the apply process moves the change and all of the other changes in the same transaction to the error queue. When apply errors occur, you must identify the cause of the error and correct it. You can retry a specific error transaction or you can retry all error transactions for an apply process.

The following topics describe managing apply errors:

- Correcting Apply Errors in Database Objects
- Retrying or Deleting Apply Error Transactions

See Also:

- "About Change Apply" on page 4-6
- "Managing Apply Processes" on page 5-6
- "Monitoring Apply Processes" on page 5-19
- *Oracle Streams Concepts and Administration*

Correcting Apply Errors in Database Objects

You might need to make data manipulation language (DML) or data definition language (DDL) changes to database objects to correct the conditions that caused one or more apply errors before you retry error transactions. If you must make changes to a database object, but you do not want to replicate these changes, then set a session tag in the session that makes the changes.

For example, to update the hr.employees table to correct an apply error:

1. Open SQL*Plus and connect as a user who can update the `hr.employees` table.

   See *Oracle Database 2 Day DBA* for more information about starting SQL*Plus.

2. Set a session tag:

   ```
   BEGIN
      DBMS_STREAMS.SET_TAG(
         tag  =>  HEXTORAW('1D'));
   END;
   /
   ```

   Ensure that you set the tag to a value that prevents changes from being captured by capture processes and synchronous captures.

3. Update the `hr.employees` table to correct the error.

4. Exit the SQL*Plus session.

After you correct the problem that caused one or more error transactions, you can retry the error transactions or delete them. See "Retrying or Deleting Apply Error Transactions" on page 5-31 for instructions.

For information about specific apply errors and how to correct them, see *Oracle Streams Replication Administrator's Guide*.

Retrying or Deleting Apply Error Transactions

"Correcting Apply Errors in Database Objects" on page 5-31 describes correcting the problem that caused one or more error transactions. After you correct the problem, you can retry the error transactions or delete them:

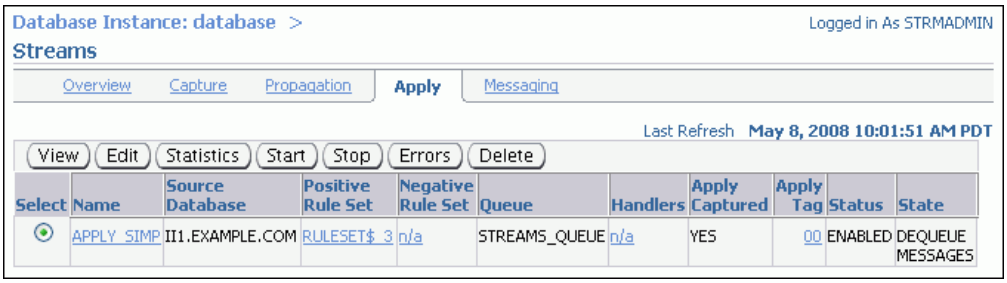
- Retry the transactions if the changes in the transaction should be made to the destination table. Retry them only if you have not already modified the data in the table to make these changes.
- Delete the transactions if you made all of the changes in the transactions to the destination table, or if the changes should not be made to the table.

To retry or delete apply error transactions:

1. In Oracle Enterprise Manager, log in to the database as the Oracle Streams administrator.
2. Go to the Database Home page.
3. Click **Data Movement** to open the Data Movement subpage.
4. Click **Manage** in the Streams section.

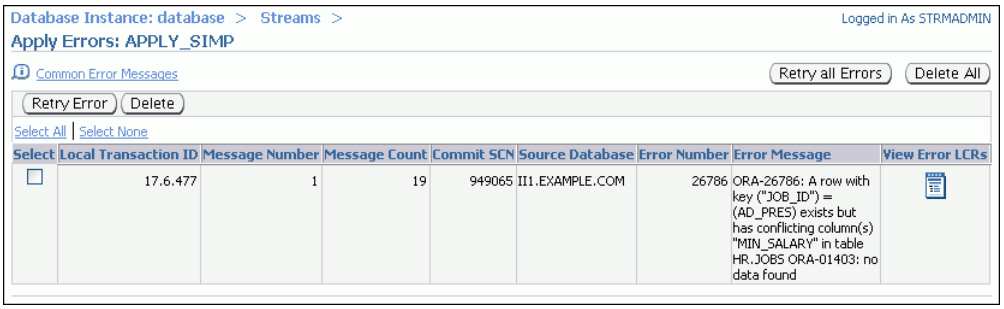
   The Streams page appears, showing the Overview subpage.

5. Click **Apply** to open the Apply subpage.

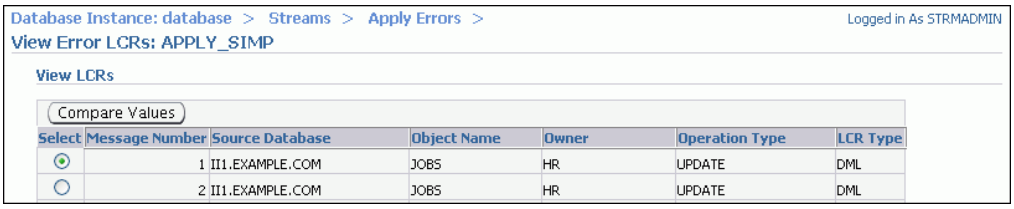


An apply error might cause an apply process to abort. When an apply process aborts, the status of the apply process is `ABORTED`.

6. Select the apply process with error transactions.
7. Click **Errors** to open the Apply Errors page.



You can view detailed information about an error by clicking the icon in the **View Error LCRs** field for the error transaction. The detailed information appears on the View Error LCRs page and includes each change in the transaction.



To drill down further, select a row change and click **Compare Values**. The Compare Values page appears and compares the row change with the data in the table to which the change should be applied. Click **Help** on the current page for information about the page.

8. On the Apply Errors page, complete one of the following actions:
   - To retry all error transactions, click **Retry all Errors**.
   - To delete all error transactions, click **Delete All**.
   - To retry a specific error transaction, select the error transaction and click **Retry Error**.
   - To delete a specific error transaction, select the error transaction and click **Delete**.

Note: You can also use the following procedures in the DBMS_APPLY_ADM package to delete or reexecute error transactions:

- DELETE_ALL_ERRORS
- DELETE_ERROR
- EXECUTE_ALL_ERRORS
- EXECUTE_ERROR

Managing a Replication Environment When a Destination Is Unavailable

Sometimes, a destination queue in an Oracle Streams replication environment stops accepting propagated changes. The destination queue might stop accepting changes if, for example, the database that contains the queue goes down, there is a problem with the destination queue, the computer system running the database that contains the queue goes down, or for some other reason.

When a destination is unavailable in a replication environment that uses capture processes, captured changes that cannot be sent to a destination queue remain in the source queue. The source queue size increases and, eventually, the changes spill out of the buffered queue memory onto disk. When this happens, the performance of the Oracle Streams replication environment suffers.

To determine whether a large number of captured changes are spilling to disk, follow the instructions in "Viewing Statistics for a Capture Process" on page 5-14. The **Queue Statistics** graph shows the current number of changes spilling to disk in the capture process queue.

If your replication environment uses capture processes to capture changes, then you can use the SPLIT_STREAMS procedure and MERGE_STREAMS_JOB procedure in the DBMS_STREAMS_ADM package to manage an unavailable destination.

To manage the unavailable destination:

1. Use the SPLIT_STREAMS procedure to split off the stream for the problem destination from all of the other streams flowing from a capture process to other destinations.

   The SPLIT_STREAMS procedure clones the capture process, queue, and propagation, and the cloned versions of these components are used by the stream that is split off. While the stream that cannot propagate changes is split off, the streams to other destinations proceed as usual.

2. Correct the problem that is causing the destination to be unavailable.

3. Use the MERGE_STREAMS_JOB procedure to merge the split stream back to the original capture process. MERGE_STREAMS_JOB can be run automatically by a merge job created by the SPLIT_STREAMS procedure. After the merge operation completes, the original replication environment is restored.

See Also:

- *Oracle Streams Replication Administrator's Guide* for detailed instructions about using split and merge
- *Oracle Database PL/SQL Packages and Types Reference*
- Chapter 4, "Replicating Data Using Oracle Streams"

6

Extending an Oracle Streams Replication Environment

This chapter describes extending an Oracle Streams replication environment by adding database objects or databases.

This chapter contains the following sections:

See Also:

- Chapter 4, "Replicating Data Using Oracle Streams"
- Chapter 5, "Administering an Oracle Streams Replication Environment"
- "When to Replicate Data with Oracle Streams" on page 1-4
- *Oracle Streams Concepts and Administration*
- *Oracle Streams Replication Administrator's Guide*

About Extending an Oracle Streams Replication Environment

Sometimes it is necessary to extend an Oracle Streams replication environment when the needs of your organization change. You can extend an Oracle Streams replication environment by adding database objects or databases.

There are two ways to extend an Oracle Streams replication environment:

- Use a Single Configuration Procedure in the DBMS_STREAMS_ADM Package
- Add the Oracle Streams Components Individually in Multiple Steps

Use a Single Configuration Procedure in the DBMS_STREAMS_ADM Package

The easiest way to extend an Oracle Streams replication environment is to run one of the following procedures in the `DBMS_STREAMS_ADM` package:

- The `MAINTAIN_GLOBAL` procedure can add a new database to an environment that replicates changes to all of the database objects in the databases.
- The `MAINTAIN_SCHEMAS` procedure can add one or more new schemas to the existing databases in the replication environment, or it can add a new database that replicates schemas that are currently being replicated.

- The `MAINTAIN_SIMPLE_TTS` procedure can add a new simple tablespace to an existing replication environment, or it can add a new database that replicates a simple tablespace that is currently being replicated.
- The `MAINTAIN_TABLES` procedure can add one or more new tables to the existing databases in the replication environment, or it can add a new database that replicates tables that are currently being replicated.
- The `MAINTAIN_TTS` procedure can add a new set of tablespaces to an existing replication environment, or it can add a new database that replicates a set of tablespaces that are currently being replicated.

To use one of these procedures to extend an Oracle Streams replication environment, the environment must meet the following conditions:

- It must be a two-database or hub-and-spoke replication environment that was configured by one of the configuration procedures in the `DBMS_STREAMS_ADM` package. See "About the Common Types of Oracle Streams Replication Environments" on page 4-11 for information about these types of replication environments.
- It cannot use a synchronous capture at any database in the Oracle Streams replication environment. See "About Change Capture with a Synchronous Capture" on page 4-4 for more information about synchronous capture.
- If you are adding a database to the environment, then each database that captures changes must use a local capture process. No database can use a downstream capture process. If you are adding one or more database objects to the environment, then the databases can use either local or downstream capture processes. See "About Change Capture with a Capture Process" on page 4-3 for more information about downstream capture.
- If you are adding database objects to the replication environment, then the database objects must exist at the database specified in the `source_database` parameter of the configuration procedure but not at any of the other databases.
- If you are adding a database to the replication environment, then the new database must not contain any of the database objects that are replicated in the current environment.

If your environment meets these conditions, then complete the steps in one of the following sections to extend the environment:

- Tutorial: Adding Database Objects to a Replication Environment
- Tutorial: Adding Databases to a Replication Environment

Add the Oracle Streams Components Individually in Multiple Steps

If you cannot extend the Oracle Streams replication environment by using a configuration procedure in the `DBMS_STREAMS_ADM` package, then you must complete the configuration steps manually. These steps include adding the necessary rules and Oracle Streams components to the environment, as well as other configuration steps.

If you must extend the Oracle Streams replication environment manually, then see the instructions in *Oracle Streams Replication Administrator's Guide*.

See Also:

- Chapter 4, "Replicating Data Using Oracle Streams"
- "About Hub-And-Spoke Replication Environments" on page 4-13
- *Oracle Database PL/SQL Packages and Types Reference* for information about the procedures in the `DBMS_STREAMS_ADM` chapter

Tutorial: Adding Database Objects to a Replication Environment

This topic includes an example that uses the `MAINTAIN_TABLES` procedure in the `DBMS_STREAMS_ADM` package to add tables to an existing hub-and-spoke replication environment. When the example is complete, the Oracle Streams replication environment replicates the changes made to the added tables at the databases in the environment. See "About the Oracle Streams Replication Configuration Procedures" on page 4-16 for more information about the configuration procedures.

Specifically, the example in this topic extends the replication environment configured in "Tutorial: Configuring Hub-and-Spoke Replication with Local Capture Processes" on page 4-40. That configuration has the following characteristics:

- The `hr` schema is replicated at the `hub.example.com`, `spoke1.example.com`, and `spoke2.example.com` databases.
- The `hub.example.com` database is the hub database in the hub-and-spoke environment, while the other databases are the spoke databases.
- The spoke databases allow changes to the replicated schema, and each database has a local capture process to capture these changes.
- Update conflict handlers are configured for each replicated table at each database to resolve conflicts

This example adds the following tables to the environment:

- `oe.orders`
- `oe.order_items`

This example uses the tables in the `oe` sample schema. The `oe` sample schema is installed by default with Oracle Database.

Note: Before you use a configuration procedure in the `DBMS_STREAMS_ADM` package to extend an Oracle Streams replication environment, ensure that the environment meets the conditions described in "About Extending an Oracle Streams Replication Environment" on page 6-1.

To add database objects to an Oracle Streams replication environment:

1. Ensure that the following directory objects exist, and remove any files related to the previous configuration from them, including Data Pump export dump files and export log files:
   - The `hub_dir` directory object at the `hub.example.com` database.
   - The `spoke1_dir` directory object at the `spoke1.example.com` database.
   - The `spoke2_dir` directory object at the `spoke2.example.com` database.

2. Stop the capture process at the hub database in the hub-and-spoke environment.

   In this example, stop the capture process at the `hub.example.com` database. The replicated database objects can remain open to changes while the capture process is stopped. These changes will be captured when the capture process is restarted.

   a. In Oracle Enterprise Manager, log in to the hub database as the Oracle Streams administrator.

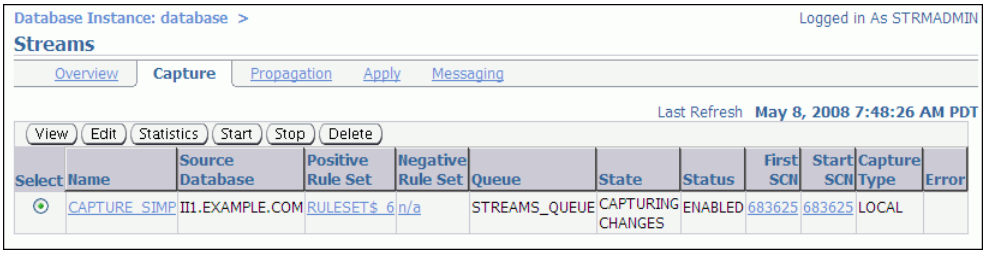
   b. Go to the Database Home page.

   c. Click **Data Movement** to open the Data Movement subpage.

   d. Click **Manage** in the Streams section.

      The Streams page appears, showing the Overview subpage.

   e. Click **Capture** to open the Capture subpage.



   f. Select the capture process that you want to stop.

   g. Click **Stop**.

   h. Click **Yes** on the confirmation page to stop the capture process.

   Note: You can also use the `DBMS_CAPTURE_ADM.STOP_CAPTURE` procedure to stop a capture process.

   ---

3. In SQL*Plus, run the appropriate configuration procedure in the `DBMS_STREAMS_ADM` package at the hub database to add each new database object for each spoke database.

   You might need to run the procedure several times if the environment has more than one spoke database. In this example, complete the following steps:

   a. Open SQL*Plus and connect to the `hub.example.com` database as the Oracle Streams administrator.

      See *Oracle Database 2 Day DBA* for more information about starting SQL*Plus.

   b. Run the `MAINTAIN_TABLES` procedure to add the `oe.orders` and `oe.order_items` tables for replication between `hub.example.com` and `spoke1.example.com`:

```
DECLARE
  tables DBMS_UTILITY.UNCL_ARRAY;
  BEGIN
    tables(1) := 'oe.orders';
    tables(2) := 'oe.order_items';
    DBMS_STREAMS_ADM.MAINTAIN_TABLES(
      table_names                 => tables,
      source_directory_object     => 'hub_dir',
      destination_directory_object => 'spoke1_dir',
```

```
      source_database             => 'hub.example.com',
      destination_database        => 'spoke1.example.com',
      capture_name                => 'capture_hns',
      capture_queue_table         => 'source_hns_qt',
      capture_queue_name          => 'source_hns',
      propagation_name            => 'propagation_spoke1',
      apply_name                  => 'apply_spoke1',
      apply_queue_table           => 'destination_spoke1_qt',
      apply_queue_name            => 'destination_spoke1',
      bi_directional              => TRUE);
END;
/
```

The MAINTAIN_TABLES procedure can take some time to run because it is performing many configuration tasks. Do not allow data manipulation language (DML) or data definition language (DDL) changes to the specified tables at the destination database while the procedure is running. When the procedure completes, the new database objects are added to the environment, and the capture process that was stopped in Step 2 is restarted. See "About the Oracle Streams Replication Configuration Procedures" on page 4-16 for more information about the configuration procedures.

When a configuration procedure is run, information about its progress is recorded in the following data dictionary views: DBA_RECOVERABLE_SCRIPT, DBA_RECOVERABLE_SCRIPT_PARAMS, DBA_RECOVERABLE_SCRIPT_BLOCKS, and DBA_RECOVERABLE_SCRIPT_ERRORS. If the procedure stops because it encounters an error, then see *Oracle Streams Replication Administrator's Guide* for instructions about using the RECOVER_OPERATION procedure in the DBMS_STREAMS_ADM package to recover from these errors.

The parameter values that specify Oracle Streams component names must be the same as the values specified in the configuration procedure in the DBMS_STREAMS_ADM package that configured the replication environment. The Oracle Streams component names specified include the capture process name, queue names, queue table names, the propagation name, and the apply process name. In this example, the Oracle Streams component names match the ones specified in "Tutorial: Configuring Hub-and-Spoke Replication with Local Capture Processes" on page 4-40.

c. Run the MAINTAIN_TABLES procedure to add the oe.orders and oe.order_items tables for replication between hub.example.com and spoke2.example.com:

```
DECLARE
  tables DBMS_UTILITY.UNCL_ARRAY;
  BEGIN
    tables(1) := 'oe.orders';
    tables(2) := 'oe.order_items';
    DBMS_STREAMS_ADM.MAINTAIN_TABLES(
      table_names                 => tables,
      source_directory_object     => 'hub_dir',
      destination_directory_object => 'spoke2_dir',
      source_database             => 'hub.example.com',
      destination_database        => 'spoke2.example.com',
      capture_name                => 'capture_hns',
      capture_queue_table         => 'source_hns_qt',
      capture_queue_name          => 'source_hns',
      propagation_name            => 'propagation_spoke2',
      apply_name                  => 'apply_spoke2',
```

```
      apply_queue_table           => 'destination_spoke2_qt',
      apply_queue_name            => 'destination_spoke2',
      bi_directional              => TRUE);
END;
/
```

4. Set the instantiation SCN for the replicated tables at the spoke databases:

   Note: This step is required in this example because the replicated tables existed at the spoke databases before the `MAINTAIN_TABLES` procedure was run. If the replicated tables did not exist at the spoke databases before the `MAINTAIN_TABLES` procedure was run, then the procedure sets the instantiation SCN for the replicated tables and this step is not required.

   a. In SQL*Plus, connect to the `hub.example.com` database as the Oracle Streams administrator.

      See *Oracle Database Administrator's Guide* for information about connecting to a database in SQL*Plus.

   b. Set the instantiation SCN for the `oe.orders` table at the `spoke1.example.com` database:

```
DECLARE
  iscn  NUMBER;    -- Variable to hold instantiation SCN value
BEGIN
  iscn := DBMS_FLASHBACK.GET_SYSTEM_CHANGE_NUMBER();
  DBMS_APPLY_ADM.SET_TABLE_INSTANTIATION_SCN@spoke1.example.com(
    source_object_name    => 'oe.orders',
    source_database_name  => 'hub.example.com',
    instantiation_scn     => iscn);
END;
/
```

   c. Set the instantiation SCN for the `oe.order_items` table at the `spoke1.example.com` database:

```
DECLARE
  iscn  NUMBER;    -- Variable to hold instantiation SCN value
BEGIN
  iscn := DBMS_FLASHBACK.GET_SYSTEM_CHANGE_NUMBER();
  DBMS_APPLY_ADM.SET_TABLE_INSTANTIATION_SCN@spoke1.example.com(
    source_object_name    => 'oe.order_items',
    source_database_name  => 'hub.example.com',
    instantiation_scn     => iscn);
END;
/
```

   d. Set the instantiation SCN for the `oe.orders` table at the `spoke2.example.com` database:

```
DECLARE
  iscn  NUMBER;    -- Variable to hold instantiation SCN value
BEGIN
  iscn := DBMS_FLASHBACK.GET_SYSTEM_CHANGE_NUMBER();
  DBMS_APPLY_ADM.SET_TABLE_INSTANTIATION_SCN@spoke2.example.com(
    source_object_name    => 'oe.orders',
    source_database_name  => 'hub.example.com',
```

```
    instantiation_scn     => iscn);
END;
/
```

e. Set the instantiation SCN for the `oe.order_items` table at the `spoke2.example.com` database:

```
DECLARE
  iscn  NUMBER;    -- Variable to hold instantiation SCN value
BEGIN
  iscn := DBMS_FLASHBACK.GET_SYSTEM_CHANGE_NUMBER();
  DBMS_APPLY_ADM.SET_TABLE_INSTANTIATION_SCN@spoke2.example.com(
    source_object_name    => 'oe.order_items',
    source_database_name  => 'hub.example.com',
    instantiation_scn     => iscn);
END;
/
```

5. Configure latest time conflict resolution for the `orders` and `order_items` tables in the `oe` schema at the `hub.example.com`, `spoke1.example.com`, and `spoke2.example.com` databases. See "Tutorial: Configuring Latest Time Conflict Resolution for a Table" on page 4-56 for instructions.

See Also:

- Chapter 4, "Replicating Data Using Oracle Streams"

Tutorial: Adding Databases to a Replication Environment

This topic includes an example that uses the `MAINTAIN_SCHEMAS` procedure in the `DBMS_STREAMS_ADM` package to add a new spoke database to an existing hub-and-spoke replication environment. When the example is complete, the Oracle Streams replication environment replicates the changes made to the schema with the new database. See "About the Oracle Streams Replication Configuration Procedures" on page 4-16 for more information about the configuration procedures.

Specifically, the example in this topic extends the replication environment configured in "Tutorial: Configuring Hub-and-Spoke Replication with Local Capture Processes" on page 4-40. That configuration has the following characteristics:

- The `hr` schema is replicated at the `hub.example.com`, `spoke1.example.com`, and `spoke2.example.com` databases.
- The `hub.example.com` database is the hub database in the hub-and-spoke environment, while the other databases are the spoke databases.
- The spoke databases allow changes to the replicated schema, and each database has a local capture process to capture these changes.

This example adds the `spoke3.example.com` database to the environment.

Note: Before you use a configuration procedure in the `DBMS_STREAMS_ADM` package to extend an Oracle Streams replication environment, ensure that the environment meets the conditions described in "About Extending an Oracle Streams Replication Environment" on page 6-1.

To add a database to an existing Oracle Streams replication environment:

1. Complete the following tasks to prepare the environment for the new database:

   a. Configure network connectivity so that the hub database can communicate with the new spoke database. In this example, configure network connectivity so that the `hub.example.com` database and the `spoke3.example.com` databases can communicate with each other.

      See *Oracle Database 2 Day DBA* for information about configuring network connectivity between databases.

   b. Configure an Oracle Streams administrator at the new spoke database. In this example, configure an Oracle Streams administrator at the `spoke3.example.com` database. See "Tutorial: Creating an Oracle Streams Administrator" on page 2-2 for instructions. This example assumes that the Oracle Streams administrator is `strmadmin`.

   c. Create a database link from the hub database to new spoke database and from new spoke database to the hub database. In this example, create the following database links:

      – From the `hub.example.com` database to the `spoke3.example.com` database. Both the name and the service name of the database link must be `spoke3.example.com`.

      – From the `spoke3.example.com` database to the `hub.example.com` database. Both the name and the service name of the database link must be `hub.example.com`.

      Each database link should be created in the Oracle Streams administrator's schema. Also, each database link should connect to the Oracle Streams administrator at the destination database. See "Tutorial: Creating a Database Link" on page 2-8 for instructions.

   d. Set initialization parameters properly at the new spoke database. In this example, set initialization parameters properly at the `spoke3.example.com` database. See "Preparing for Oracle Streams Replication" on page 4-21 for instructions.

   e. Configure the new spoke database to run in `ARCHIVELOG` mode. For a capture process to capture changes generated at a source database, the source database must be running in `ARCHIVELOG` mode. In this example, configure the `spoke3.example.com` database to run in `ARCHIVELOG` mode. See *Oracle Database Administrator's Guide* for information about configuring a database to run in `ARCHIVELOG` mode.

   f. Ensure that the `hub_dir` directory objects exist at the `hub.example.com` database, and remove any files related to the previous configuration from it, including Data Pump export dump files and export log files.

2. Open SQL*Plus and connect to the `spoke3.example.com` database as the Oracle Streams administrator.

   See *Oracle Database 2 Day DBA* for more information about starting SQL*Plus.

3. Create a directory object to hold files that will be generated by the `MAINTAIN_SCHEMAS` procedure, including the Data Pump export dump file used for instantiation. The directory object can point to any accessible directory on the computer system. For example, the following statement creates a directory object named `spoke3_dir` that points to the `/usr/spoke3_log_files` directory:

   ```
   CREATE DIRECTORY spoke3_dir AS '/usr/spoke3_log_files';
   ```

4. Stop the capture process at the hub database in the hub-and-spoke environment.

   In this example, stop the capture process at the `hub.example.com` database. The replicated database objects can remain open to changes while the capture process is stopped. These changes will be captured when the capture process is restarted.

   a. In Oracle Enterprise Manager, log in to the hub database as the Oracle Streams administrator.

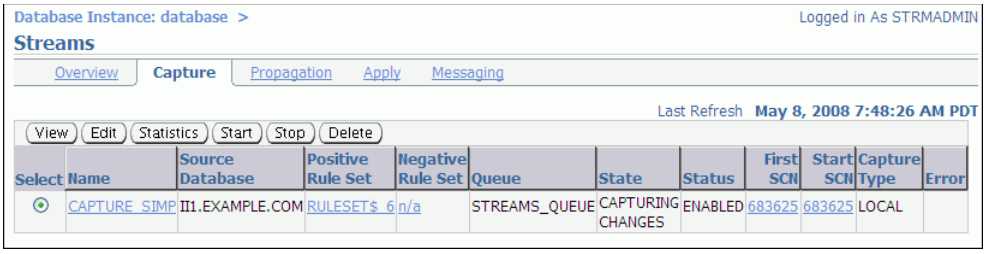
   b. Go to the Database Home page.

   c. Click **Data Movement** to open the Data Movement subpage.

   d. Click **Manage** in the Streams section.

      The Streams page appears, showing the Overview subpage.

   e. Click **Capture** to open the Capture subpage.



   f. Select the capture process that you want to stop.

   g. Click **Stop**.

   h. Click **Yes** on the confirmation page to stop the capture process.

   Note: You can also use the `DBMS_CAPTURE_ADM.STOP_CAPTURE` procedure to stop a capture process.

5. In SQL*Plus, run the appropriate configuration procedure in the `DBMS_STREAMS_ADM` package at the hub database to add the new spoke database.

   In this example, complete the following steps:

   a. Open SQL*Plus and connect to the `hub.example.com` database as the Oracle Streams administrator.

   b. Run the `MAINTAIN_SCHEMAS` procedure to add the `spoke3.example.com` database to the Oracle Streams replication environment:

```
BEGIN
  DBMS_STREAMS_ADM.MAINTAIN_SCHEMAS(
    schema_names                 => 'hr',
    source_directory_object      => 'hub_dir',
    destination_directory_object => 'spoke3_dir',
    source_database              => 'hub.example.com',
    destination_database         => 'spoke3.example.com',
    capture_name                 => 'capture_hns',
    capture_queue_table          => 'source_hns_qt',
    capture_queue_name           => 'source_hns',
    propagation_name             => 'propagation_spoke3',
    apply_name                   => 'apply_spoke3',
    apply_queue_table            => 'destination_spoke3_qt',
```

```
    apply_queue_name             => 'destination_spoke3',
    bi_directional               => TRUE);
END;
/
```

The `MAINTAIN_SCHEMAS` procedure can take some time to run because it is performing many configuration tasks. Do not allow data manipulation language (DML) or data definition language (DDL) changes to the database objects in the specified schema at the destination database while the procedure is running. When the procedure completes, the new database objects are added to the environment, and the capture process that was stopped in Step 4 is restarted. See "About the Oracle Streams Replication Configuration Procedures" on page 4-16 for more information about the configuration procedures.

The parameter values specified in `capture_name`, `capture_queue_table`, and `capture_queue_name` must be the same as the values specified in the configuration procedure in the `DBMS_STREAMS_ADM` package that configured the replication environment. In this example, these parameter values match the ones specified in "Tutorial: Configuring Hub-and-Spoke Replication with Local Capture Processes" on page 4-40.

When a configuration procedure is run, information about its progress is recorded in the following data dictionary views: `DBA_RECOVERABLE_SCRIPT`, `DBA_RECOVERABLE_SCRIPT_PARAMS`, `DBA_RECOVERABLE_SCRIPT_BLOCKS`, and `DBA_RECOVERABLE_SCRIPT_ERRORS`. If the procedure stops because it encounters an error, then see *Oracle Streams Replication Administrator's Guide* for instructions about using the `RECOVER_OPERATION` procedure in the `DBMS_STREAMS_ADM` package to recover from these errors.

6. Configure latest time conflict resolution for all of the tables in the `hr` schema at the `spoke3.example.com` database. This schema includes the `countries`, `departments`, `employees`, `jobs`, `job_history`, `locations`, and `regions` tables. "Tutorial: Configuring Latest Time Conflict Resolution for a Table" on page 4-56 for instructions.

See Also:

- Chapter 4, "Replicating Data Using Oracle Streams"

7

Replicating Data Using Materialized Views

This chapter contains conceptual information about replicating data using materialized views. It also describes how to synchronize the replicated data periodically to a transactionally consistent point in time.

This chapter contains the following sections:

See Also:

- Chapter 8, "Administering a Materialized View Replication Environment"
- Chapter 4, "Replicating Data Using Oracle Streams"
- Chapter 1, "Introduction to Data Replication and Integration"

About Materialized View Replication

Replication is the process of sharing database objects and data at multiple databases. To maintain replicated database objects and data at multiple databases, a change to one of these database objects at a database is shared with the other databases. In this way, the database objects and data are kept synchronized at all of the databases in the replication environment.

In many environments, such as field sales, databases are not connected to the network for periods of time. In these environments, replicas must synchronize data on demand, or at regularly scheduled intervals, such as nightly. These databases can synchronize with other databases that share the same data when they are reconnected to the network.

To address these needs, Oracle Database offers a replication type called **materialized view** replication. A materialized view contains a complete or partial copy of a table from a single point in time. Materialized views can be either read-only or updatable:

- **Read-only materialized views** provide read-only access to the table copy. Using read-only materialized views, applications and users can access local copies of tables that reside at remote locations. Read-only materialized views provide local access to data and reduce the resources required at any single location by allowing queries on the same data at multiple locations. For example, read-only materialized views are typically used for reporting purposes.
- **Updatable materialized views** provide read/write access to the table copy. Using updatable materialized views, applications and users can change both the table and the copy of the table, and these changes can be synchronized at a point in time. For example, updatable materialized views are typically used to periodically disseminate a product catalog to regional offices and to enable the sales force to place orders from customer sites.

Because of their support for easy mass deployment and disconnected computing, both read-only and updatable materialized views are especially suited to mobile computing applications.

The following topics provide more information about materialized view replication:

- About Master Sites, Master Tables, and Materialized View Sites
- About Materialized View Refresh
- About Refresh Groups

See Also:

- "When to Replicate Data with Materialized Views" on page 1-5
- "Replicating Read-Only Data Using Materialized Views" on page 7-8
- "Replicating Read/Write Data Using Materialized Views" on page 7-12

About Master Sites, Master Tables, and Materialized View Sites

In a replication environment, a materialized view shares data with a table in a different database called a **master site**. The table associated with the materialized view at the master site is called the **master table**. A materialized view contains a complete or partial copy of a master table from a single point in time. The database in which the materialized view resides is called the **materialized view site**.

See Also:

- "Configuring Materialized View Sites" on page 7-4

About Materialized View Refresh

Refreshing a materialized view synchronizes data in the materialized view with data in its master table. The materialized view controls when the refresh is performed, either on a fixed schedule or on demand. During a refresh, row data from the master table is pulled down and applied at the materialized view site. Oracle Database provides the following refresh methods:

- **Fast refresh** pulls down only the rows that have changed since the last refresh.
- **Complete refresh** updates the entire materialized view.
- **Force refresh** performs a fast refresh when possible. When a fast refresh is not possible, force refresh performs a complete refresh.

A fast refresh is more efficient than a complete refresh. A fast refresh of a materialized view is possible only if the master table has a **materialized view log**. A materialized view log is a table at the master site that records all of the insert, update, and delete operations performed on the master table.

A materialized view log is associated with a single master table, and each master table has only one materialized view log, regardless of how many materialized views refresh from the master table. When a fast refresh is performed on a materialized view, entries in the materialized view log that have appeared since the materialized view was last refreshed are applied to the materialized view.

See Also:

- "Refreshing Materialized Views" on page 8-2
- "About Refresh Groups" on page 7-3
- "Determining Which Materialized Views Are Currently Refreshing" on page 8-13

About Refresh Groups

When it is important for two or more materialized views to be transactionally consistent with each other, you can organize them into **refresh groups**. By refreshing a refresh group, you can ensure that the data in all of the materialized views in the refresh group correspond to the same transactionally consistent point in time. A materialized view in a refresh group still can be refreshed individually. However, doing so nullifies the benefits of the refresh group because refreshing the materialized view individually does not refresh the other materialized views in the refresh group.

See Also:

- "About Materialized View Refresh" on page 7-2
- "Refreshing a Refresh Group" on page 8-2

Preparing for Materialized View Replication

Before configuring materialized view replication, prepare the databases that will participate in the replication environment.

To prepare for materialized view replication:

1. If possible, then ensure that each replicated table has a primary key. Where a primary key is not possible, each replicated table must have a column or set of columns that can be used as a unique identifier for each row of the table. If the tables that you plan to use in your replication environment do not have a primary key or a set of unique columns, then alter these tables accordingly.
2. Set initialization parameters properly at each database in your replication environment before you configure materialized view replication:
   - **Global names:** Set the `GLOBAL_NAMES` initialization parameter to `TRUE` at each database that will participate in the replication environment. See "Setting the GLOBAL_NAMES Initialization Parameter to TRUE" on page 2-1.
   - **System Global Area (SGA) and the shared pool:** Ensure that the shared pool is large enough to accommodate the components created for the replication environment. The shared pool is part of the SGA. You can manage the shared pool by setting the `MEMORY_TARGET` initialization parameter (Automatic Memory Management), the `SGA_TARGET` initialization parameter (Automatic

Shared Memory Management), or the SHARED_POOL_SIZE initialization parameter. Typically, the shared pool should be larger for an Oracle database in a replication environment than in a nonreplication environment.

See *Oracle Database 2 Day DBA* for information about modifying initialization parameters.

3. Configure network connectivity so that the databases in the replication environment can communicate with each other. See *Oracle Database 2 Day DBA* for information about configuring network connectivity between databases.

See Also:

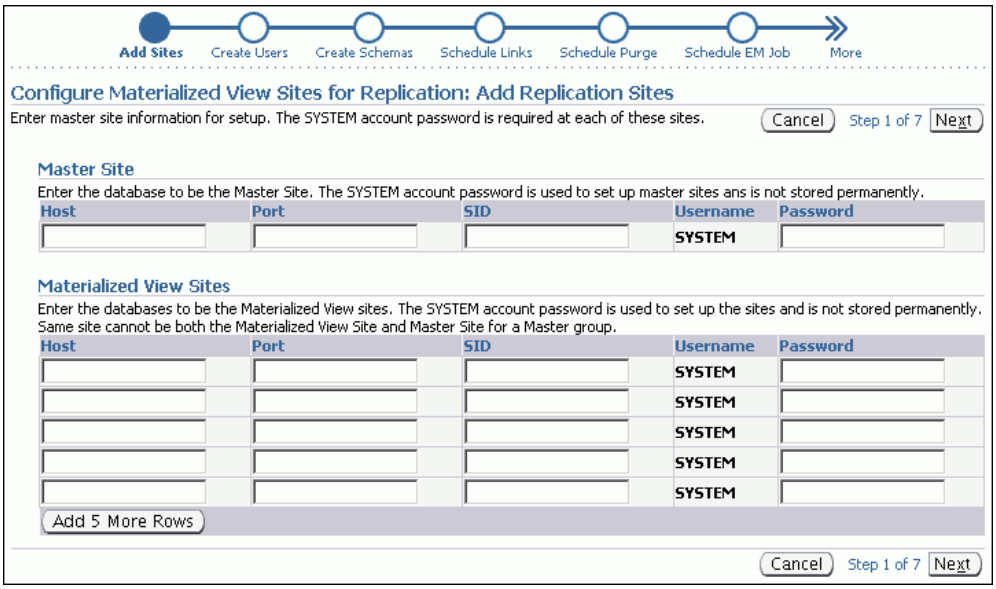
- "Replicating Read-Only Data Using Materialized Views" on page 7-8
- "Replicating Read/Write Data Using Materialized Views" on page 7-12
- *Oracle Database Advanced Replication* for more information about initialization parameters in a replication environment

Configuring Materialized View Sites

Before you configure materialized views, you must configure the databases that will function as materialized view sites in your replication environment. Completing the steps in this topic will configure the necessary users and database links required by the materialized view environment. In addition, if you plan to configure updatable materialized views, then these steps also configure scheduled links and scheduled purges for changes made to updatable materialized views.

To configure materialized view sites with Enterprise Manager:

1. Complete the actions described in "Preparing for Materialized View Replication" on page 7-3.
2. In Oracle Enterprise Manager, log in to the materialized view site as an administrative user, such as SYSTEM.
3. Go to the Database Home page.
4. Click **Data Movement** to open the Data Movement subpage.
5. Click **Setup** in the Advanced Replication section.
6. On the Advanced Replication: Setup page, expand **Updateable Materialized View Replication**.
7. Select **Configure Materialized View Sites for Replication**.
8. Click **Continue** to open the Configure Materialized View Sites for Replication Wizard.



Complete the pages in the wizard to configure the materialized view sites. On the Create Users page, enter an appropriate password for the materialized view administrator. See *Oracle Database 2 Day + Security Guide* for information about choosing passwords. For more information about using the wizard, click **Help** for each page of the wizard.

> **Tip:** On the Create Schemas page of the wizard, make sure you select the schemas that contain the database objects that you want to replicate.

When you finish the wizard, an Enterprise Manager job is scheduled to configure the materialized view sites. After the job runs successfully, the materialized view sites are configured, and each one has a materialized view administrator. By default, the user name of the materialized view administrator is `mvadmin`. You specified the password for this user when you completed the wizard. Also, each materialized view site contains each schema that you specified on the Create Schemas page of the wizard.

9. At each materialized view site, create the required database links for the replicated schemas.

   To create materialized views in a schema at the materialized view site, the schema must be able to connect to the corresponding schema at the master site through a database link.

   To create the database links:

   a. On a command line, open SQL*Plus and connect to the materialized view database as the owner of the replicated schema.

      For example, if you plan to create materialized views in the `hr` schema at the `ii2.example.com` database, then enter the following:

      ```
      sqlplus hr@ii2.example.com
      Enter password: password
      ```

      See *Oracle Database 2 Day DBA* for more information about starting SQL*Plus.

b. Create a database link that connects to the corresponding replicated schema at the master database.

   For example, to create a database link that connects to the `hr` schema at the `ii1.example.com` database, enter the following:

```
CREATE DATABASE LINK ii1.example.com CONNECT TO hr
   IDENTIFIED BY password USING 'ii1.example.com';
```

c. Repeat Steps a through b for each replicated schema at the materialized view database.

d. Repeat Steps a through c for each materialized view database.

See Also:

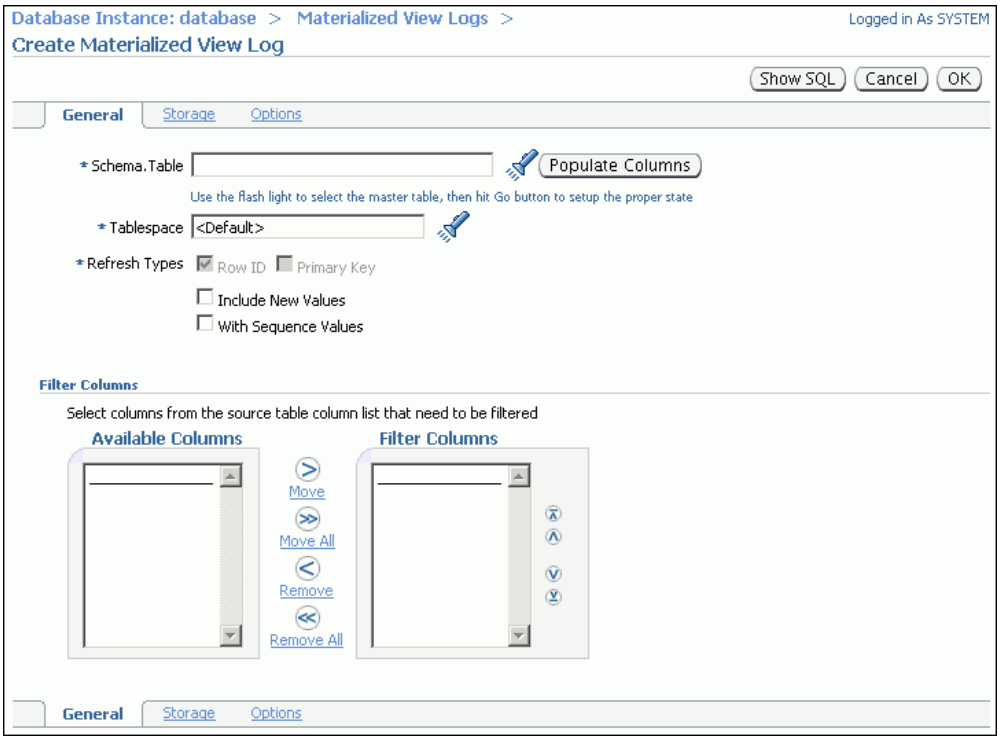
- "When to Replicate Data with Materialized Views" on page 1-5
- "About Master Sites, Master Tables, and Materialized View Sites" on page 7-2

Configuring Materialized View Logs at the Master Site

To enable fast refresh of a materialized view, configure a materialized view log for its master table at the master site. A fast refresh is more efficient than a complete refresh, because it updates only rows that have changed since the last refresh. See "About Materialized View Refresh" on page 7-2 for more information.

To configure materialized view logs:

1. In Oracle Enterprise Manager, log in to the master site as an administrative user, such as `SYSTEM`.
2. Go to the Database Home page.
3. Click **Data Movement** to open the Data Movement subpage.
4. Click **Setup** in the Advanced Replication section.
5. On the Advanced Replication: Setup page, expand **Updateable Materialized View Replication**.
6. Select **Create Materialized View log on source database tables for fast refresh**.
7. Click **Continue** to open the Create Materialized View Log page.



8. Enter the schema name and table name for the master table in the form *schema*.*table* in the **Schema.Table** field, or click the flashlight icon to find the table.
9. Click **Populate Columns**.
10. If you have a specific tablespace for the materialized view log you are creating, then enter the tablespace name in the **Tablespace** field. Otherwise, leave `<Default>` in the field. Click **Help** for more information about the page.
11. Ensure that **Primary Key** is selected in Refresh Types.
12. Click **OK** to create the materialized view log.
13. Repeat Steps 5 through 12 for each master table.

Note: You can also use the `CREATE MATERIALIZED VIEW LOG` SQL statement to create a materialized view log.

See Also:

- "About Materialized View Refresh" on page 7-2
- "Viewing Materialized View Logs at a Master Site" on page 8-18
- "Preventing Materialized View Logs From Becoming Too Large" on page 8-22
- "Tutorial: Cleaning Up Materialized View Support at a Master Site" on page 8-8

Replicating Read-Only Data Using Materialized Views

A read-only replica of a table is appropriate for databases that provide access to the replicated data without allowing applications to modify the replicated data.

The following topics describe replicating read-only data using materialized views:

- About Replicating Read-Only Data Using Materialized Views
- Tutorial: Configuring Read-Only Data Replication Using Materialized Views

See Also:

- "When to Replicate Data with Materialized Views" on page 1-5
- "Replicating Read/Write Data Using Materialized Views" on page 7-12
- "Administering a Materialized View Replication Environment" on page 8-1

About Replicating Read-Only Data Using Materialized Views

Read-only materialized views provide read-only access to the table data that originates from a master site. Applications can query data from read-only materialized views to avoid network access to the master site, regardless of network availability. However, applications throughout the system must access data at the master site to perform insert, update, and delete operations.

Figure 7–1 shows how a read-only materialized view works.

Figure 7–1 Read-Only Materialized View

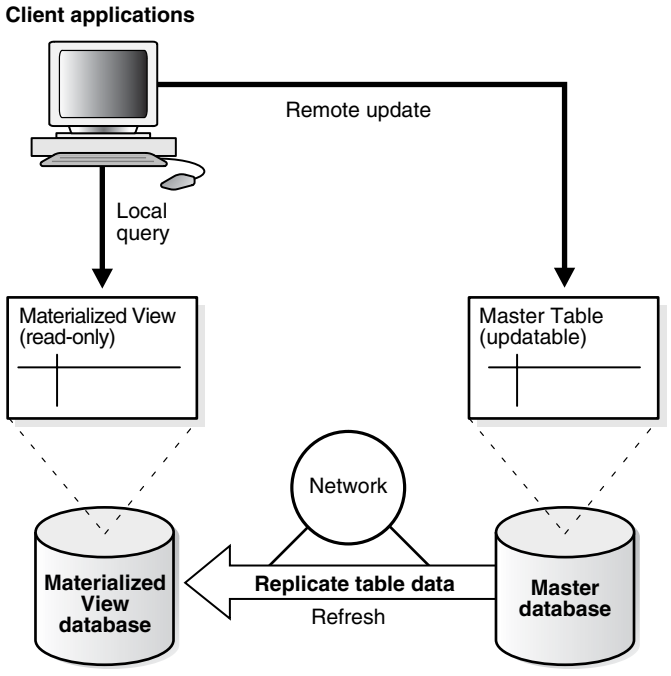


Figure 7–1 shows that client applications can perform local queries on a read-only materialized view at the materialized view site. These applications can update data at the remote master. The materialized view is updated with the changes at the master when the materialized view refreshes over the network.

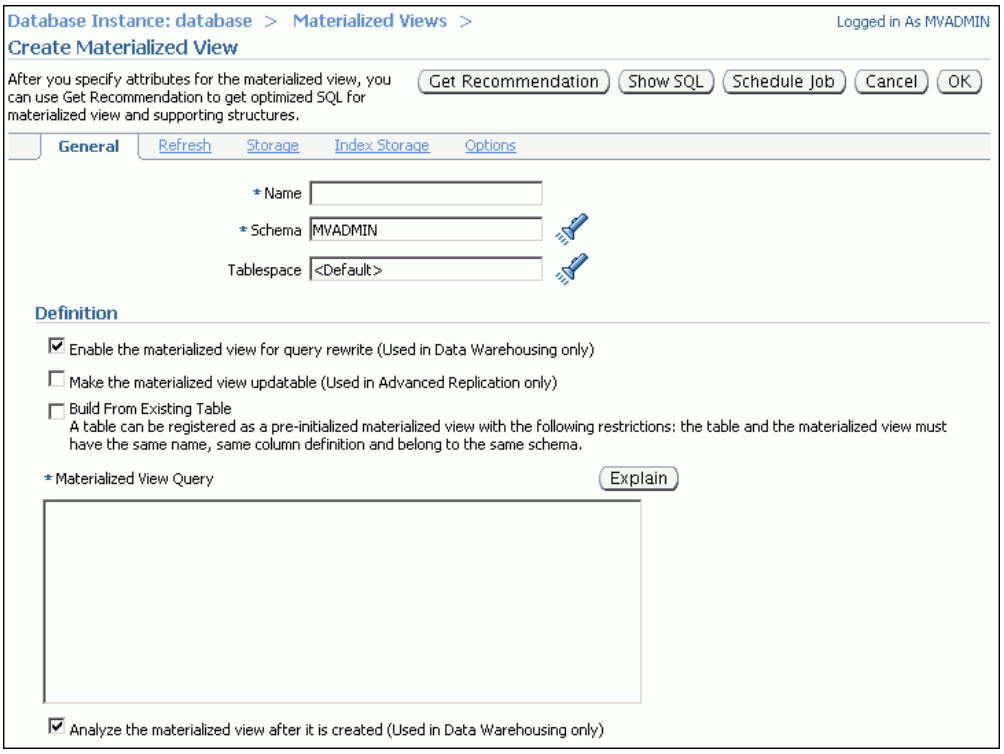
Tutorial: Configuring Read-Only Data Replication Using Materialized Views

The example in this topic configures read-only data replication using materialized views. Specifically, this example creates the following read-only materialized views at a materialized view site:

- The `mvadmin.employees_mvr` materialized view based on the `hr.employees` table at the master site.
- The `mvadmin.departments_mvr` materialized view based on the `hr.departments` table at the master site.

To configure these read-only materialized views:

1. Complete the actions described in the following topics:
   - "Preparing for Materialized View Replication" on page 7-3
   - "Configuring Materialized View Sites" on page 7-4
   - "Configuring Materialized View Logs at the Master Site" on page 7-6 if you want to perform a fast refresh of the materialized views. For this example, create a materialized view log for each of the following tables: `hr.departments` and `hr.employees`.
2. In Enterprise Manager, log in to the materialized view site as the materialized view administrator. The materialized view site is the database in which you want to create materialized views. The default user name for the materialized view administrator is `mvadmin`.
3. Go to the Database Home page.
4. Click **Data Movement** to open the Data Movement subpage.
5. Click **Setup** in the Advanced Replication section.
6. On the Advanced Replication: Setup page, expand **Read-only Materialized Views**.
7. Select **Create Materialized View**.
8. Click **Continue** to open the Create Materialized View page.



9. Complete the following actions to create the `hr.employees_mvr` materialized view:

   a. Enter `employees_mvr` in the **Name** field.

   b. Enter `hr` in the **Schema** field.

   c. If you have a specific tablespace for the materialized view you are creating, then enter the tablespace name in the **Tablespace** field. Otherwise, leave `<Default>` in the field.

   d. Deselect **Enable the materialized view for query rewrite**.

   e. Ensure that the following are not selected:

      – **Make the materialized view updatable**

      – **Build From Existing Table**

   f. Enter the following in the **Materialized View Query** field:

```
SELECT employee_id,
        first_name,
        last_name,
        email,
        phone_number,
        hire_date,
        job_id,
        salary,
        commission_pct,
        manager_id,
        department_id
FROM hr.employees@master_site
```

      Replace *`master_site`* with the global name of the master site that contains the master table.

g. Deselect **Analyze the materialized view after it is created**.

h. If necessary, then set options for the materialized view on the other subpages. Click **Help** on a selected subpage for information about the subpage.

i. Click **OK** to create the materialized view.

The `hr.employees_mvr` materialized view is a complete copy of the master `hr.employees` table. To create a materialized view that contains a subset of the data, you can include a `WHERE` clause in your `SELECT` statement.

10. On the Advanced Replication: Setup page, expand **Read-only Materialized Views**.
11. Select **Create Materialized View**.
12. Click **Continue** to open the Create Materialized View page.
13. Complete the following actions to create the `hr.departments_mvr` materialized view:

a. Enter `departments_mvr` in the **Name** field.

b. Enter `hr` in the **Schema** field.

c. If you have a specific tablespace for the materialized view you are creating, then enter the tablespace name in the **Tablespace** field. Otherwise, leave `<Default>` in the field.

d. Deselect **Enable the materialized view for query rewrite**.

e. Ensure that the following are not selected:

- **Make the materialized view updatable**
- **Build From Existing Table**

f. Enter the following in the **Materialized View Query** field:

```
SELECT department_id,
       department_name,
       manager_id,
       location_id
FROM hr.departments@master_site
```

Replace *`master_site`* with the master site that contains the master table.

g. Deselect **Analyze the materialized view after it is created**.

h. If necessary, then set options for the materialized view on the other subpages. Click **Help** on a selected subpage for information about the subpage.

i. Click **OK** to create the materialized view.

14. Create a refresh group and add the materialized views to it so that they are consistent with a single point in time when they are refreshed. In this example, add the `hr.employees_mvr` and `hr.departments_mvr` materialized views to the refresh group. See "Configuring a Refresh Group" on page 7-23.

Note: You can also use the `CREATE MATERIALIZED VIEW` SQL statement to create a materialized view.

See Also:

- "When to Replicate Data with Materialized Views" on page 1-5
- "About Replicating Read-Only Data Using Materialized Views" on page 7-8
- "Administering a Materialized View Replication Environment" on page 8-1

Replicating Read/Write Data Using Materialized Views

A read/write replica of a table is appropriate for databases that allow applications to modify the replicated data.

The following topics describe replicating read/write data using materialized views:

- About Replicating Read/Write Data Using Materialized Views
- Configuring Replication of Read/Write Data Using Materialized Views

See Also:

- "When to Replicate Data with Materialized Views" on page 1-5
- "Replicating Read-Only Data Using Materialized Views" on page 7-8
- Chapter 8, "Administering a Materialized View Replication Environment"

About Replicating Read/Write Data Using Materialized Views

Updatable materialized views enable users to insert, update, and delete rows in the materialized views. When an updatable materialized view is refreshed, these changes are pushed to the master site and applied to the master table.

Figure 7–2 shows how an updatable materialized view works.

Figure 7–2 Updatable Materialized View

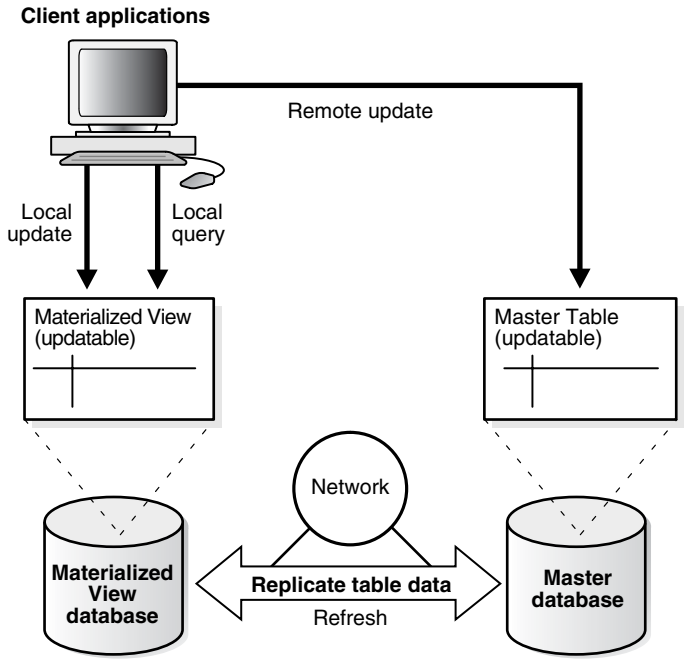


Figure 7–2 shows that client applications can query and update an updatable materialized view at the materialized view site. These applications can also update data at the remote master site. When the materialized view initializes a refresh over the network, the refresh includes two key phases. First, transactions that have been performed on the materialized view are pushed to the master site and applied as appropriate. Next, after resolving any conflicting updates, the materialized view pulls the changed row data from the master site and applies it.

The following topics provide more information about replicating read/write data using materialized views:

- About Replication Groups and Updatable Materialized Views
- About Scheduled Links and Deferred Transactions
- About Conflicts and Updatable Materialized Views

See Also:

- "Configuring Replication of Read/Write Data Using Materialized Views" on page 7-16
- Chapter 8, "Administering a Materialized View Replication Environment"

About Replication Groups and Updatable Materialized Views

A **replication group** is a collection of replication objects that are logically related. Organizing related database objects within a replication group makes it easier to administer many objects together. At a master site, a replication group is called a **master group**. At a materialized view site, a replication group is called a **materialized view group**.

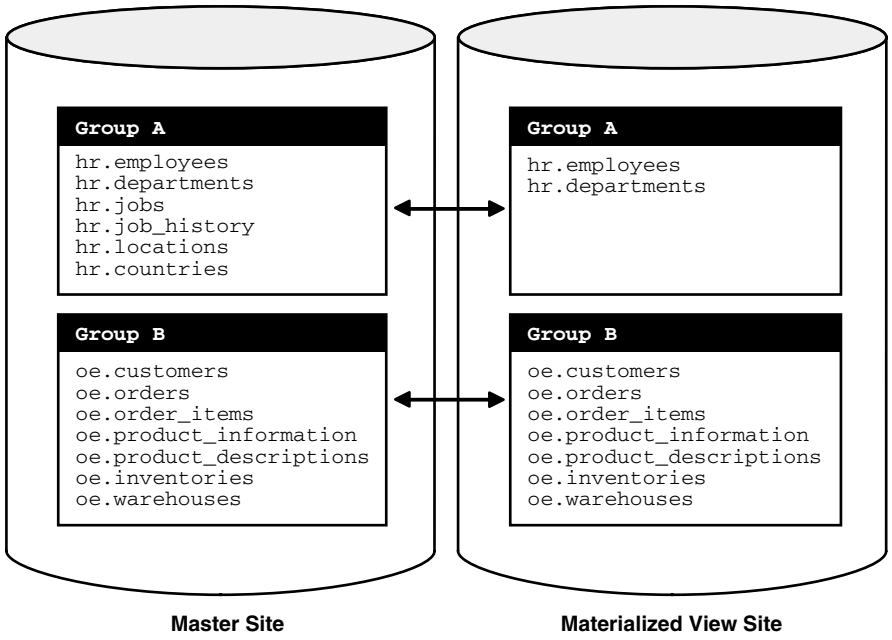
For changes made to an updatable materialized view to be pushed to the master during refresh, the following conditions must be met:

- The updatable materialized view must belong to a materialized view group.
- The materialized view group must be based on a master group at the master site.

The materialized view group can contain materialized views for all of the tables in the master group or for a subset of the tables.

Figure 7–3 shows two materialized view groups. Materialized view group A contains materialized views that correspond to a subset of the tables in the master group, while materialized view group B contains a materialized view for each table in the master group.

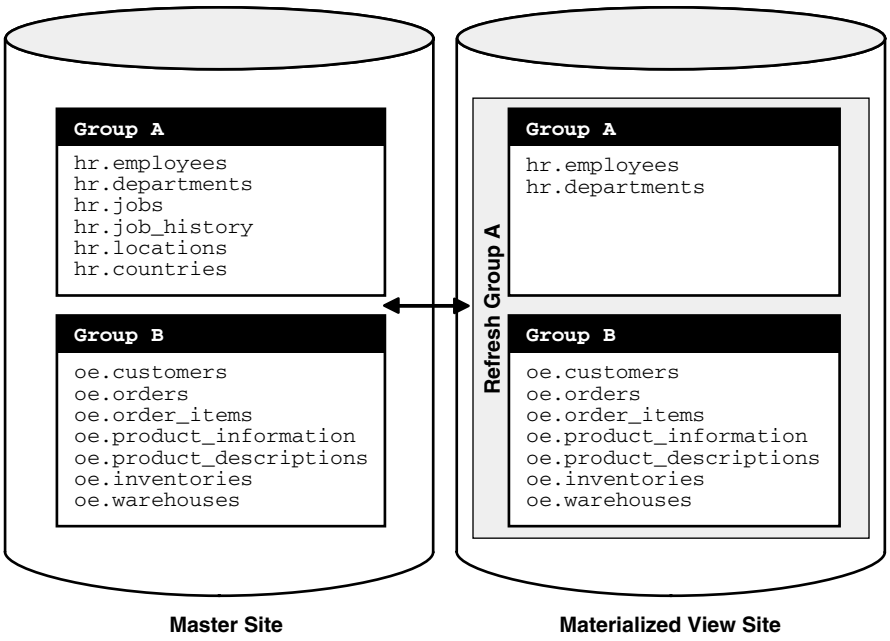
Figure 7–3 Materialized View Groups Correspond with Master Groups



If you are using materialized view groups and refresh groups at the same site, then a single refresh group can contain multiple materialized view groups. A materialized view group is not the same as a refresh group, although it can contain the same materialized views.

Figure 7–4 shows a refresh group that contains two materialized view groups.

Figure 7–4 Refresh Groups Can Contain Objects from Multiple Materialized View Groups



See Also:

- "About Refresh Groups" on page 7-3
- "Configuring Replication of Read/Write Data Using Materialized Views" on page 7-16

About Scheduled Links and Deferred Transactions

At a materialized view site, **deferred transactions** include changes to updatable materialized views. The deferred transactions are stored at the materialized view site so that they can be sent to the master site and applied to master tables.

A refresh of an updatable materialized view first pushes the deferred transactions at the materialized view site to its master site. Then, the data at the master site is pulled down and applied to the materialized view.

Optionally, you can choose to push the deferred transactions at a regular interval independent of refresh. A **scheduled link** is a database link with a user-defined schedule to push deferred transactions. A scheduled link determines how a materialized view site sends its deferred transaction queue to its master site. When you create a scheduled link, Oracle Database creates a job in the local job queue to push the deferred transaction queue to the master site.

If the materialized view site has a constant connection to its master site, then you optionally can use a scheduled link to push the deferred transactions to the master site at regular intervals. If the materialized view site is disconnected from its master site for extended periods of time, then it is typically better not to push deferred transactions on a schedule. In this case, it is best to refresh on demand, which also pushes changes to the master site.

After changes made to updatable materialized views are pushed to the master site, they no longer need to be stored at the materialized view site. To keep the size of the deferred transaction queue in check, you should purge successfully completed deferred transactions. These deferred transactions can be purged at regular intervals according to a **purge schedule**.

See Also:

- "About Replicating Read/Write Data Using Materialized Views" on page 7-12
- "About Master Sites, Master Tables, and Materialized View Sites" on page 7-2
- "About Materialized View Refresh" on page 7-2
- "Configuring Materialized View Sites" on page 7-4
- "Viewing Information About Deferred Transactions for Updatable Materialized Views" on page 8-15

About Conflicts and Updatable Materialized Views

When you use updatable materialized views, **conflicts** are possible. A conflict occurs when a change is made to the same row in the master table and materialized view at nearly the same time. During refresh, the master site detects a conflict when the same row was changed at the master table and its corresponding materialized view since the last refresh.

By default, Oracle Database always detects and logs conflicting updates. You can use **conflict resolution** to ensure that each conflict is resolved in accordance with your business rules and to ensure that the data converges correctly at all sites. Oracle Database provides built-in conflict resolution methods that you can configure at the master site.

See Also:

- "Configuring Replication of Read/Write Data Using Materialized Views" on page 7-16
- *Oracle Database Advanced Replication*
- "About Replicating Read/Write Data Using Materialized Views" on page 7-12
- "About Replication Groups and Updatable Materialized Views" on page 7-13

Configuring Replication of Read/Write Data Using Materialized Views

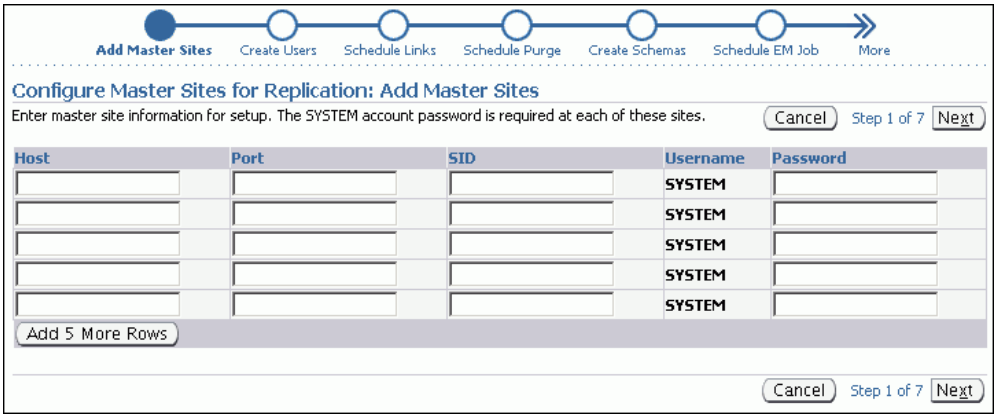
This topic provides instructions for configuring read/write data replication using materialized views. Specifically, this example configures:

- A master group at a master site that contains master tables. See "About Replication Groups and Updatable Materialized Views" on page 7-13.
- A materialized view group at a materialized view site that contains updatable materialized views that are based on the master tables at the master site. For materialized views to be updatable, they must be in a materialized view group at the materialized view site, and the materialized view group must correspond with a master group at the master site. See "About Replication Groups and Updatable Materialized Views" on page 7-13.
- Latest timestamp conflict resolution at the master site for the master tables. Conflict resolution ensures that the master tables and materialized views remain consistent if changes are made to the same rows in the master table and its corresponding updatable materialized view at nearly the same time. See "About Conflicts and Updatable Materialized Views" on page 7-16.

When a conflict occurs, latest timestamp conflict resolution means that the most recent change is retained and the older change is discarded. So, if a row is updated in a materialized view at one point in time, and then the row is updated in the master table at a later time, then the row in the master table replaces the row in the materialized view automatically when the materialized view is refreshed. If the row in the materialized view was updated more recently than the row in the master table, then the row in the materialized view replaces the row in the master table during refresh.

To configure updatable materialized views:

1. Complete the actions described in the following topics:
   - "Preparing for Materialized View Replication" on page 7-3
   - "Configuring Materialized View Sites" on page 7-4
   - "Configuring Materialized View Logs at the Master Site" on page 7-6 if you want to perform a fast refresh of the materialized views. Configure a materialized view log for each table that will be a master table.
2. Configure the master site. The master site contains the master tables on which the updatable materialized views are based.
   a. In Oracle Enterprise Manager, log in to the database that will be the master site as an administrative user, such as `SYSTEM`.
   b. Go to the Database Home page.
   c. Click **Data Movement** to open the Data Movement subpage.
   d. Click **Setup** in the Advanced Replication section.
   e. On the Advanced Replication: Setup page, expand **Updateable Materialized View Replication**.
   f. Select **Configure Master Sites for Replication**.
   g. Click **Continue** to open the Configure Master Sites for Replication Wizard.



Complete the pages in the wizard to configure the master site. For more information about using the wizard, click **Help** for each page of the wizard.

When you finish the wizard, an Enterprise Manager job is scheduled to configure the master site. After the job runs successfully, the master site is configured, and it has a replication administrator. By default, the user name of the replication administrator is `repadmin`. You specified the password for this user when you completed the wizard.

3. Add a `time` column to each master table for latest timestamp conflict resolution.

   a. In SQL*Plus, connect to the database as an administrative user, such as `SYSTEM`. Alternatively, you can connect as the user who owns the table to which the time column will be added.

      See *Oracle Database 2 Day DBA* for more information about starting SQL*Plus.

   b. Use the `ALTER TABLE` SQL statement to add the `time` column to the table. For example, the following SQL statement adds the `time` column to the `hr.departments` table.

      ```
      ALTER TABLE hr.departments ADD (time TIMESTAMP WITH TIME ZONE);
      ```

4. Create a trigger to update the `time` column in each master table with the current time when a change occurs.

   Tip: Instead of using a trigger to update the `time` column, an application can populate the `time` column each time it modifies or inserts a row into a table.

   a. In Oracle Enterprise Manager, log in to the database that contains the master tables as an administrative user, such as `SYSTEM`.

      If you did not log out of Enterprise Manager after completing Step 3, then you can remain logged in to the master site as an administrative user.

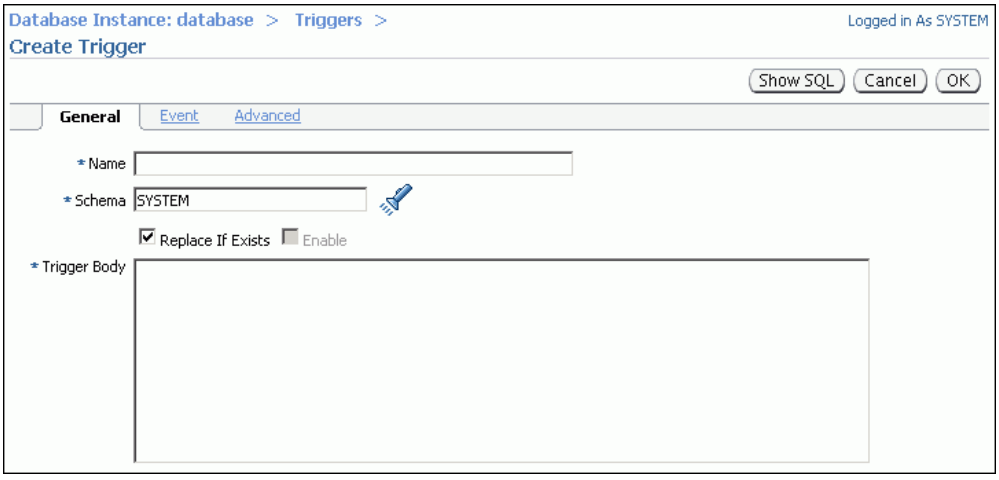
   b. Go to the Database Home page.

   c. Click **Schema** to open the Schema subpage.

   d. Click **Triggers** in the Programs section.

   e. On the Triggers page, click **Create**.

      The Create Trigger page appears, showing the General subpage.



   f. Enter the name of the trigger in the **Name** field.

   g. Retain the administrative user name in the **Schema** field.

h. Enter the following in the **Trigger Body** field:

```
BEGIN
   -- The IF/THEN statement ensures that the trigger does not fire during
   -- materialized view refresh.
   IF (DBMS_REPUTIL.FROM_REMOTE = FALSE AND
       DBMS_SNAPSHOT.I_AM_A_REFRESH = FALSE)
   THEN
      :NEW.TIME := SYSTIMESTAMP;
   END IF;
END;
```

i. Click **Event** to open the Event subpage.

j. Ensure that **Table** is selected in the Trigger On list.

k. Enter the table name in the form *schema.table* in the **Table (Schema.Table)** field, or use the flashlight icon to find the database object.

l. Ensure that **Before** is selected for **Fire Trigger**.

m. Select **Insert** and **Update of Columns** for **Event**.

The columns in the table appear.

n. Select every column in the table except for the new `time` column.

o. Click **Advanced**.

p. Select **Trigger for each row**.

q. Click **OK** to create the trigger.

r. Repeat Steps e through q to create a trigger for each master table.

s. Log out of Enterprise Manager.

Note: You can also use the `CREATE TRIGGER` SQL statement to create a trigger.

---

5. Create the master group.

a. In Enterprise Manager, log in to the master site as the replication administrator. By default, the user name of the replication administrator is `repadmin`.

b. Go to the Database Home page.

c. Click **Data Movement** to open the Data Movement subpage.

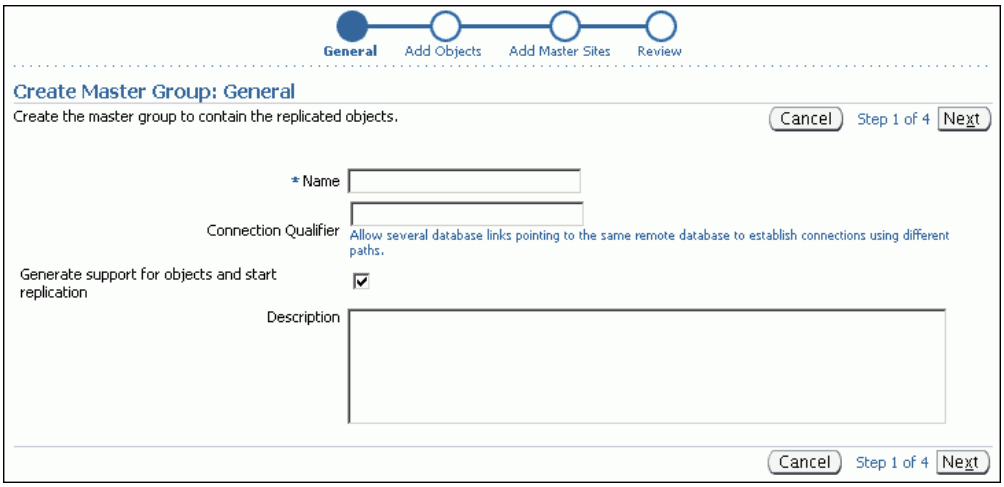
d. Click **Setup** in the Advanced Replication section.

The Advanced Replication: Setup page appears.

e. Expand **Multi-master Replication**.

f. Select **Create Master Group**.

g. Click **Continue** to open the Create Master Group Wizard.



Complete the pages in the wizard to configure the master group. For more information about using the wizard, click **Help** for each page of the wizard.

Tip:

- On the Add Objects page, add the tables that will be master tables for the materialized views.
- On the Add Master Sites page, do not add any additional master sites.

When you finish the wizard, an Enterprise Manager job is scheduled to configure the master group.

6. Configure conflict resolution for each master table:
   a. While still logged in as the replication administrator at the master site, go to the Database Home page of the database that contains the new master group.
   b. Click **Data Movement** to open the Data Movement subpage.
   c. Click **Manage** in the Advanced Replication section.

      The Advanced Replication: Administration page appears, showing the Overview subpage.
   d. Click the number associated with **Master Groups** in the Multimaster Replication section.
   e. On the Master Groups page, if the master group status is `NORMAL`, then click **Quiesce**. If the status is `QUIESCED`, then move on to the next step.
   f. On a command line, open SQL*Plus and connect to the master site as the replication administrator.

      For example, if the replication administrator is `repadmin` and the master site is `ii1.example.com`, then enter the following:

      ```
      sqlplus repadmin@ii1.example.com
      Enter password: password
      ```

      See *Oracle Database 2 Day DBA* for more information about starting SQL*Plus.
   g. In SQL*Plus, run the `MAKE_COLUMN_GROUP` procedure in the `DBMS_REPCAT` package to create a column group for a replicated table.

For example, to create a column group for the `hr.departments` table, run the following procedure:

```
BEGIN
   DBMS_REPCAT.MAKE_COLUMN_GROUP(
      sname                => 'hr',
      oname                => 'departments',
      column_group         => 'dep_time_cg',
      list_of_column_names => 'department_id,
                               department_name,
                               manager_id,
                               location_id,
                               time');
END;
/
```

Include all of the table columns in the `list_of_columns` parameter.

See *Oracle Database Advanced Replication Management API Reference* for more information about the `MAKE_COLUMN_GROUP` procedure.

h. In SQL*Plus, run the `ADD_UPDATE_RESOLUTION` procedure in the `DBMS_REPCAT` package to specify `LATEST TIME` conflict resolution for the table.

For example, to specify `LATEST TIME` conflict resolution using the column group you created in Step g for the `hr.departments` table, run the following procedure:

```
BEGIN
   DBMS_REPCAT.ADD_UPDATE_RESOLUTION (
      sname                => 'hr',
      oname                => 'departments',
      column_group         => 'dep_time_cg',
      sequence_no          => 1,
      method               => 'LATEST TIMESTAMP',
      parameter_column_name => 'time');
END;
/
```

Specify the column group you created in Step g for the `column_group` parameter, and specify the time column for the `parameter_column_name` column.

See *Oracle Database Advanced Replication Management API Reference* for more information about the `ADD_UPDATE_RESOLUTION` procedure.

i. Repeat Steps g through h for each table in the master group.

j. In Enterprise Manager, log in to the master site as the replication administrator. By default, the user name of the replication administrator is `repadmin`.

k. Go to the Database Home page.

l. Click **Data Movement** to open the Data Movement subpage.

m. Click **Manage** in the Advanced Replication section.

The Advanced Replication: Administration page appears, showing the Overview subpage.

n. Click the number associated with **Master Groups** in the Multimaster Replication section.

o. Select the master group.

p. Click **Edit** to open the General subpage of the Edit Master Group page.

q. Click **Objects** to open the Objects subpage.

r. Ensure that **Generate Replication Support** is selected for each object to which you added conflict resolution.

s. Click **Apply** to save your changes.

t. Click **Master Groups** at the top of the page to return to the Master groups page.

u. Click **Resume** for the master group.

7. Create the materialized view group.

a. In Enterprise Manager, log in to the materialized view site as the materialized view administrator. The materialized view site is the database in which you want to create materialized views. The default user name for the materialized view administrator is `mvadmin`.

b. Go to the Database Home page.

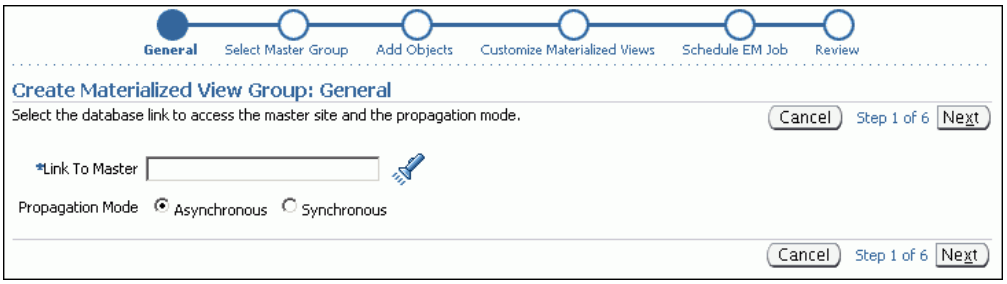
c. Click **Data Movement** to open the Data Movement subpage.

d. Click **Setup** in the Advanced Replication section.

e. On the Advanced Replication: Setup page, expand **Updateable Materialized View Replication**.

f. Select **Create Materialized View Group**.

g. Click **Continue** to open the Create Materialized View Group Wizard.



Complete the pages in the wizard to configure the materialized group. For more information about using the wizard, click **Help** for each page of the wizard.

Tip: For each materialized view on the Customize Materialized Views page:

- Select **Min. Communications** for each table that uses conflict resolution at the master site. This option reduces the amount of data required to support conflict resolution mechanisms.
- Select **Updatable** for each materialized view that you want to be updatable.
- Select **Fast Refresh** for each materialized view on which you want to perform fast refreshes.

When you finish the wizard, an Enterprise Manager job is scheduled to configure the materialized view group.

8. Create a trigger to update the `time` column in each materialized view with the current time when a change occurs. The trigger is required for latest timestamp conflict resolution.
   a. In Enterprise Manager, log in to the materialized view site as the materialized view administrator. The materialized view site is the database in which you want to create materialized views. The default user name for the materialized view administrator is `mvadmin`.
   b. Complete Steps 4b through 4q for each materialized view.
9. If you did not specify an existing refresh group during materialized view group configuration, then create a refresh group and add the materialized views to it so that they are consistent to a single point in time when they are refreshed. See "Configuring a Refresh Group" on page 7-23.

See Also:

- "When to Replicate Data with Materialized Views" on page 1-5
- "About Replicating Read/Write Data Using Materialized Views" on page 7-12
- "About Replication Groups and Updatable Materialized Views" on page 7-13
- "About Conflicts and Updatable Materialized Views" on page 7-16
- Chapter 8, "Administering a Materialized View Replication Environment"

Configuring a Refresh Group

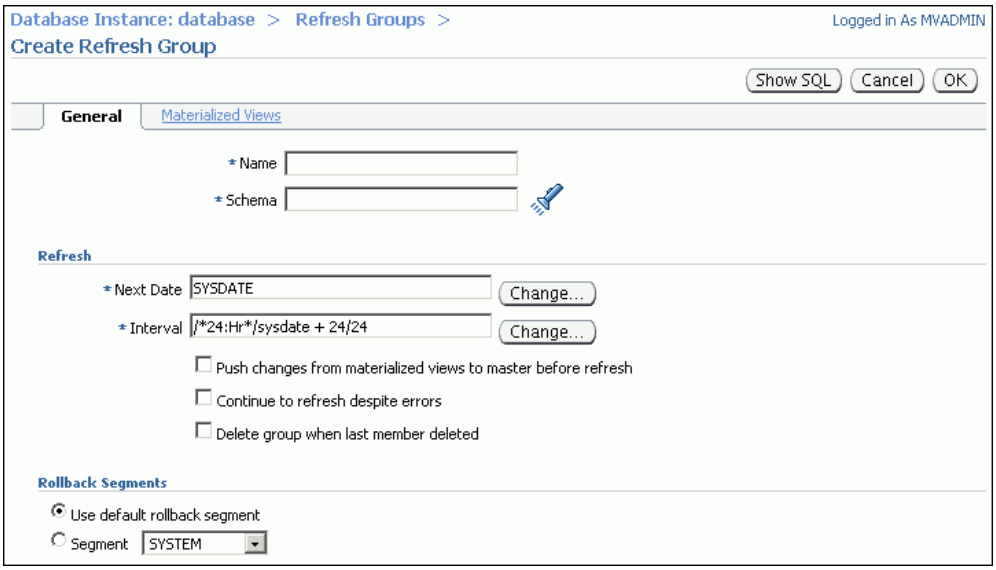
When two or more materialized views must be consistent to a point in time, the materialized views should belong to the same refresh group.

To create a refresh group and add materialized views to it:

1. In Enterprise Manager, log in to the materialized view site as the materialized view administrator. The default user name for the materialized view administrator is `mvadmin`.
2. Go to the Database Home page.
3. Click **Data Movement** to open the Data Movement subpage.
4. Click **Manage** in the Advanced Replication section.

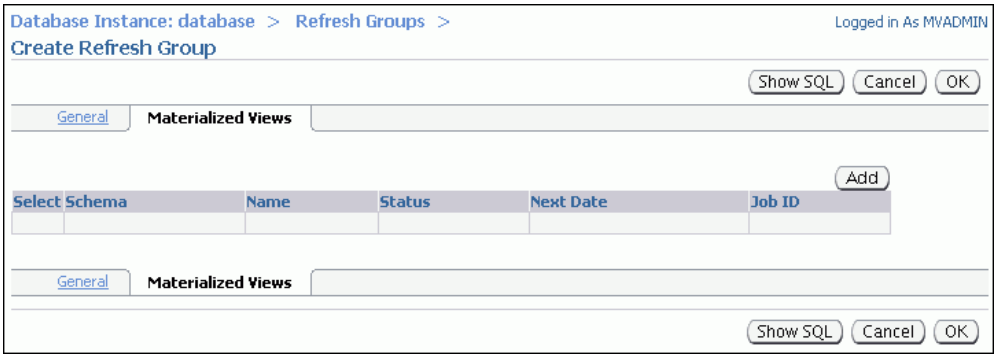
   The Advanced Replication: Administration page appears, showing the Overview subpage.
5. Click the number associated with **Refresh Groups** in the Materialized View Site section.
6. On the Refresh Groups page, click **Create**.

   The Create Refresh Group page appears, showing the General subpage.



7. Complete the following actions:
   - Enter a name for the refresh group in the **Name** field.
   - Enter the owner of the refresh group in the **Schema** field. Typically, the materialized view administrator owns refresh groups.
   - If necessary, then adjust the **Next Date** and **Interval** settings. The **Next Date** setting determines when the next refresh of the refresh group will occur. The **Interval** setting determines how often the refresh group is refreshed automatically. You can click **Change** to open a new page that adjusts each setting. Click **Help** for more information.

     If you have limited connectivity between the materialized view site and the master site, then you might want to refresh on demand. In this case, clear the **Interval** field.
   - Select **Push changes from materialized views to master before refresh**. This option specifies that, during refresh, the materialized view site pushes changes made to updatable materialized views to the master tables before refreshing the materialized views.
8. Click **Materialized Views** to open the Materialized Views subpage.



9. Click **Add** to open the Search and Select: Materialized View page.
10. Use the search tool to list the materialized views that you want to add to the refresh group.
11. Select the materialized views that you want to add to the refresh group.
12. Click **OK** to return to the Materialized Views subpage. The selected materialized views should be listed in the refresh group.
13. Click **OK** to create the refresh group.

Note: You can also use the following procedures in the `DBMS_REFRESH` package to create a refresh group and add objects to it:

- `MAKE`
- `ADD`

See Also:

- "About Refresh Groups" on page 7-3
- "Refreshing a Refresh Group" on page 8-2
- "Adding Materialized Views to a Refresh Group" on page 8-5
- "Viewing Information About Refresh Groups" on page 8-17

8

Administering a Materialized View Replication Environment

This chapter describes how to manage, monitor, and troubleshoot a materialized view replication environment.

This chapter contains the following sections:

- Managing a Materialized View Replication Environment
- Monitoring a Materialized View Replication Environment
- Troubleshooting a Materialized View Replication Environment

See Also:

- Chapter 7, "Replicating Data Using Materialized Views"
- Chapter 1, "Introduction to Data Replication and Integration"

Managing a Materialized View Replication Environment

After materialized views are configured, they can be refreshed to synchronize them with their master tables. This section includes instructions for refreshing materialized views and adding materialized views to a refresh group.

Sometimes, materialized views are no longer needed at a materialized view site. This section also includes information about dropping materialized views and cleaning up materialized view support at a master site.

The following topics describe managing a materialized view replication environment:

- Refreshing Materialized Views
- Adding Materialized Views to a Refresh Group
- Dropping a Materialized View
- Tutorial: Cleaning Up Materialized View Support at a Master Site

See Also:

- "Monitoring a Materialized View Replication Environment" on page 8-9
- "Troubleshooting a Materialized View Replication Environment" on page 8-21
- Chapter 7, "Replicating Data Using Materialized Views"

Refreshing Materialized Views

Refreshing materialized views synchronizes them with their master tables. Changes made to a master table since the last refresh of a materialized view are applied to the materialized view during refresh. If a materialized view is updatable, then changes made to the materialized view since its last refresh are also applied to the master table during refresh.

If a group of materialized views is used for a specific purpose, such as an application, then it is usually best to place these materialized views in a refresh group. When you refresh the refresh group, the materialized views in the refresh group are refreshed to a consistent point in time. You can also refresh materialized views individually, but then the materialized views are not consistent to a point in time.

The following topics describe refreshing materialized views:

- Refreshing a Refresh Group
- Refreshing a Materialized View

See Also:

- "About Materialized View Refresh" on page 7-2
- "About Refresh Groups" on page 7-3
- "About Replication Groups and Updatable Materialized Views" on page 7-13

Refreshing a Refresh Group

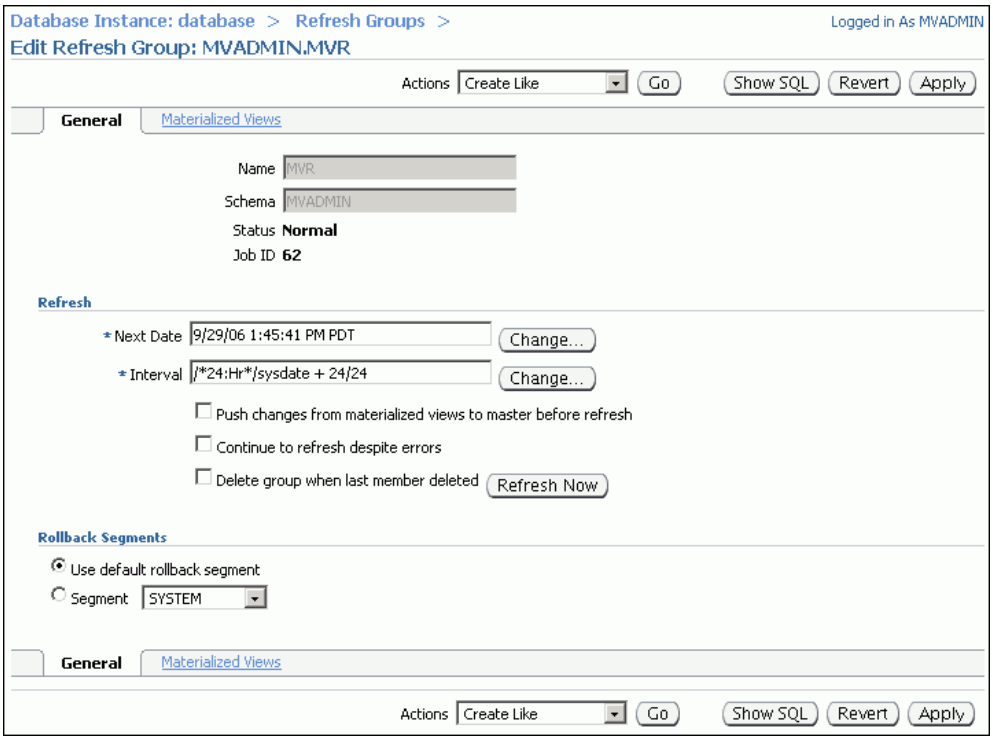
When you refresh a refresh group, all of the materialized views in the refresh group are refreshed to a consistent point in time.

To refresh a refresh group:

1. In Enterprise Manager, log in to the materialized view site as the materialized view administrator. The default user name for the materialized view administrator is `mvadmin`.
2. Go to the Database Home page.
3. Click **Data Movement** to open the Data Movement subpage.
4. Click **Manage** in the Advanced Replication section.

   The Advanced Replication: Administration page appears, showing the Overview subpage.
5. Click the number associated with **Refresh Groups** in the Materialized View Site section.
6. On the Refresh Groups page, use the search tool to list the refresh group that you want to refresh.
7. Select the refresh group in the list.
8. Click **Edit**.

   The Edit Refresh Group page appears, showing the General subpage.



9. Click **Refresh Now** in the Refresh section.

Note: You can also use the following procedures in the `DBMS_REFRESH.REFRESH` procedure to refresh a refresh group.

See Also:

- "About Materialized View Refresh" on page 7-2
- "About Refresh Groups" on page 7-3
- "Configuring a Refresh Group" on page 7-23
- "Viewing Information About Refresh Groups" on page 8-17

Refreshing a Materialized View

Refreshing a materialized view makes it consistent with its master table. If the materialized view is part of a refresh group, then it is usually best to refresh the refresh group instead of the materialized view individually. Refreshing the refresh group ensures that all of the materialized views in the refresh group are consistent to a point in time.

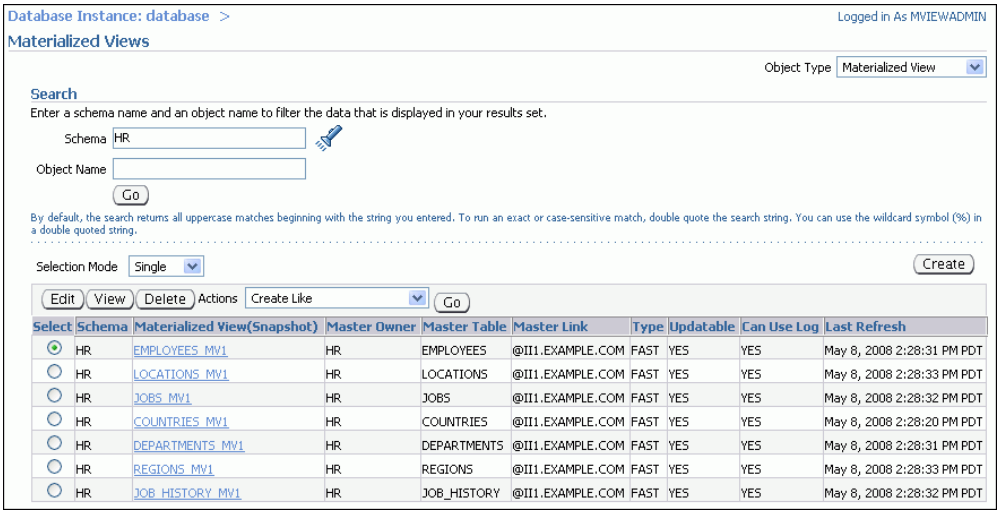
To refresh a materialized view:

1. In Enterprise Manager, log in to the materialized view site as the materialized view administrator. The default user name for the materialized view administrator is `mvadmin`.
2. Go to the Database Home page.
3. Click **Data Movement** to open the Data Movement subpage.
4. Click **Manage** in the Advanced Replication section.

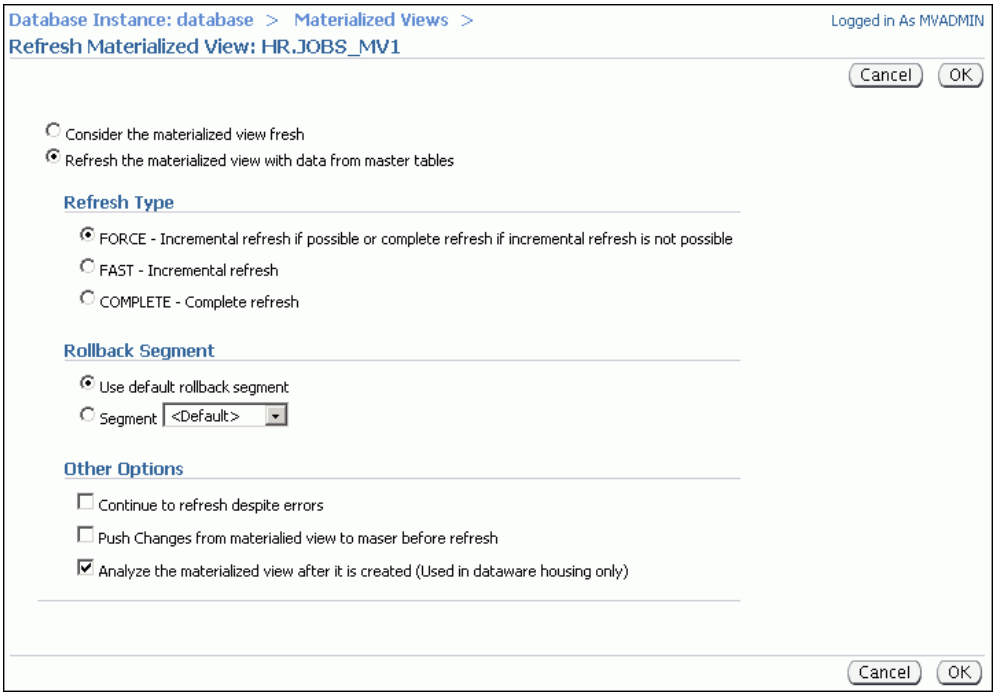
The Advanced Replication: Administration page appears, showing the Overview subpage.

5. Click the number associated with **Materialized Views** in the Materialized View Site section.
6. On the Materialized Views page, use the search tool to list the materialized view that you want to refresh.

   To list all materialized views, leave the **Schema** and **Object Name** fields blank and click **Go**.
7. Select the materialized view in the list.



8. Select **Refresh** in the Actions list.
9. Click **Go** to open the Refresh Materialized View page for the selected materialized view.



10. Adjust the options if necessary.

    Typically, the default values should be used. Click **Help** for information about this page.

11. Click **OK**.

12. On the confirmation page, click **Yes** to refresh the materialized view.

Note: You can also use the `DBMS_MVIEW.REFRESH` procedure to refresh a materialized view.

See Also:

- "About Materialized View Refresh" on page 7-2
- "About Refresh Groups" on page 7-3

Adding Materialized Views to a Refresh Group

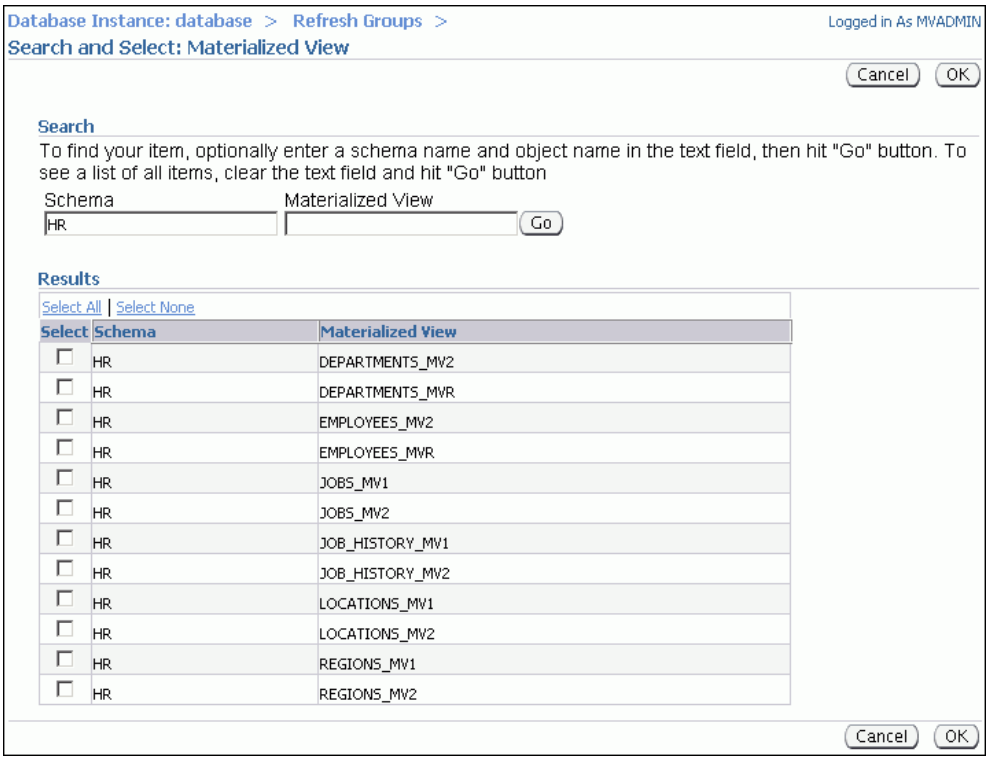
When two or more materialized views must be consistent to a point in time, the materialized views should belong to the same refresh group.

To add materialized views to a refresh group:

1. In Enterprise Manager, log in to the materialized view site as the materialized view administrator. The default user name for the materialized view administrator is `mvadmin`.
2. Go to the Database Home page.
3. Click **Data Movement** to open the Data Movement subpage.
4. Click **Manage** in the Advanced Replication section.

   The Advanced Replication: Administration page appears, showing the Overview subpage.

5. Click the number associated with **Refresh Groups** in the Materialized View Site section.
6. On the Refresh Groups page, use the search tool to list the refresh group that you want to modify.
7. Select the refresh group.
8. Click **Edit**.
9. On the Edit Refresh Group page, click **Materialized Views**.
10. On the Materialized Views subpage, click **Add**.
11. On the Search and Select: Materialized View page, use the search tool to list the materialized views that you want to add to the refresh group.



12. Select the materialized views that you want to add to the refresh group.
13. Click **OK**.
14. On the Edit Refresh Group page, click **Apply** to finish adding the materialized views to the refresh group.

Note: You can also use the `DBMS_REFRESH.ADD` procedure to add materialized views to a refresh group.

See Also:

- "About Materialized View Refresh" on page 7-2
- "About Refresh Groups" on page 7-3
- "Refreshing a Refresh Group" on page 8-2
- "Configuring a Refresh Group" on page 7-23
- "Viewing Information About Refresh Groups" on page 8-17

Dropping a Materialized View

When a materialized view is no longer needed at a materialized view site, you can drop the materialized view. If possible, then you should drop the materialized view when the materialized view site can connect to the master site. Information about the materialized view is removed automatically from the master site when a network connection is established.

If you must drop a materialized view when the materialized view site cannot connect to the master site, then information about the materialized view must be removed from the master site manually. See "Tutorial: Cleaning Up Materialized View Support

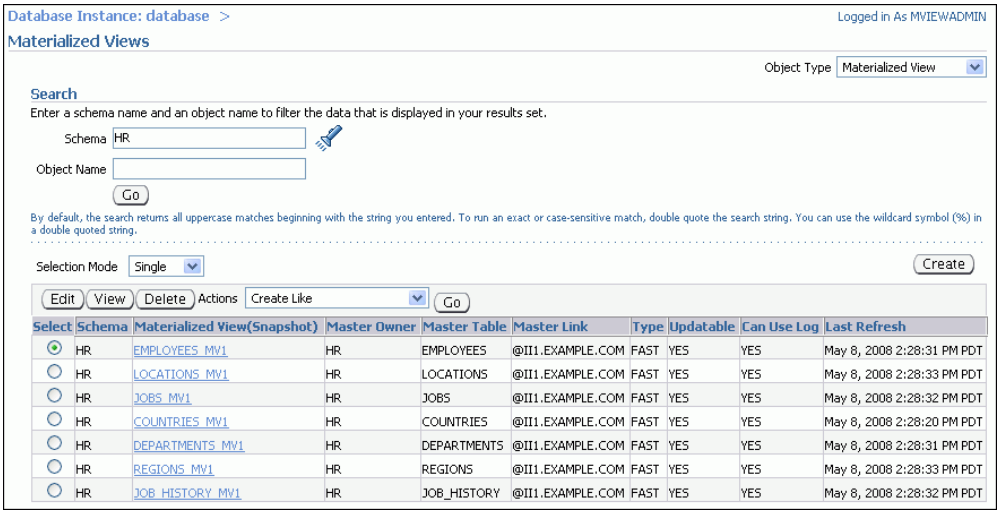
at a Master Site" on page 8-8 to learn how to remove this information after you drop the materialized view.

To drop a materialized view:

1. On Enterprise Manager, log in to the materialized view site as the materialized view administrator. The default user name for the materialized view administrator is `mvadmin`.
2. Go to the Database Home page.
3. Click **Data Movement** to open the Data Movement subpage.
4. Click **Manage** in the Advanced Replication section.

   The Advanced Replication: Administration page appears, showing the Overview subpage.
5. Click the number associated with **Materialized Views** in the Materialized View Site section.
6. On the Materialized Views page, use the search tool to list the materialized view that you want to drop.

   To list all materialized views, leave the **Schema** and **Object Name** fields blank and click **Go**.
7. Select the materialized view in the list.



8. Click **Delete**.
9. On the confirmation page, click **Yes** to delete the materialized view.

Note: You can also use the `DROP MATERIALIZED VIEW` SQL statement to drop a materialized view.

See Also:

- "About Materialized View Replication" on page 7-1

Tutorial: Cleaning Up Materialized View Support at a Master Site

If you dropped a materialized view while the materialized view site was not connected to the master site over the network, then you should clean up materialized view support at the master site. Cleaning up materialized view support includes the following actions:

- Unregistering the materialized view

  Unregistering the materialized view removes information about the materialized view in the data dictionary at the master site.

- Either purging the materialized view log or dropping the materialized view log

  If the master table for the materialized view is the master table for other materialized views, then you should purge the materialized view log of information for the materialized view that was dropped. If the master table for the materialized view is not a master table for any other materialized views, then you can drop the materialized view log.

Oracle Database automatically tracks which rows in a materialized view log have been used during the refreshes of materialized views, and purges these rows from the log so that the log size does not increase endlessly. Because multiple materialized views can use the same materialized view log, rows already used to refresh one materialized view might still be needed to refresh another materialized view. Oracle Database does not delete rows from the log until all materialized views have used them. If you drop a materialized view without cleaning up the master site, then the materialized view log for the materialized view can become very large.

The example in this topic cleans up the following materialized view support:

- The master site is `ii1.example.com`.
- The materialized view site is `ii2.example.com`.
- The name of the materialized view is `employees_mvr`.
- The owner of the materialized view is `hr`.

To clean up support for this materialized view at the master site:

1. On a command line, open SQL*Plus and connect to the master site `ii1.example.com` as an administrative user, such as the replication administrator or `SYSTEM`. By default, the user name of the replication administrator is `repadmin`.

   See *Oracle Database 2 Day DBA* for more information about starting SQL*Plus.

2. Either purge or drop the materialized view log for the master table of the materialized view:

   - If the master table is also the master table for other materialized views in addition to the one that was dropped, then purge the materialized view log of information for the materialized view that was dropped:

     ```
     BEGIN
        DBMS_MVIEW.PURGE_MVIEW_FROM_LOG (
           mviewowner => 'hr',
           mviewname  => 'employees_mvr',
           mviewsite  => 'ii2.example.com');
     END;
     /
     ```

- If the master table is not a master table for any materialized views other than the one that was dropped, then drop the materialized view log on the master table:

```
DROP MATERIALIZED VIEW LOG ON hr.employees;
```

3. Unregister the materialized view:

```
BEGIN
   DBMS_MVIEW.UNREGISTER_MVIEW (
      mviewowner => 'hr',
      mviewname  => 'employees_mvr',
      mviewsite  => 'ii2.example.com');
END;
/
```

If you are not sure about the materialized view owner, name, or materialized view site, then you can query the `ALL_REGISTERED_MVIEWS` data dictionary view to obtain this information.

See Also:

- Dropping a Materialized View on page 8-6
- "About Materialized View Refresh" on page 7-2
- "About Materialized View Replication" on page 7-1

Monitoring a Materialized View Replication Environment

This section describes using Enterprise Manager and SQL*Plus to view information about a materialized view replication environment. It describes viewing this information at both materialized view sites and master sites.

The following topics describe monitoring a materialized view replication environment:

- Viewing an Overview of the Replication Components at a Database
- Viewing Information About Materialized Views
- Determining Which Materialized Views Are Currently Refreshing
- Viewing Information About Materialized View Groups
- Viewing Information About Deferred Transactions for Updatable Materialized Views
- Viewing Information About Refresh Groups
- Viewing Materialized View Logs at a Master Site
- Viewing the Materialized Views for a Master Site

See Also:

- Chapter 7, "Replicating Data Using Materialized Views"
- "Managing a Materialized View Replication Environment" on page 8-1
- "Troubleshooting a Materialized View Replication Environment" on page 8-21

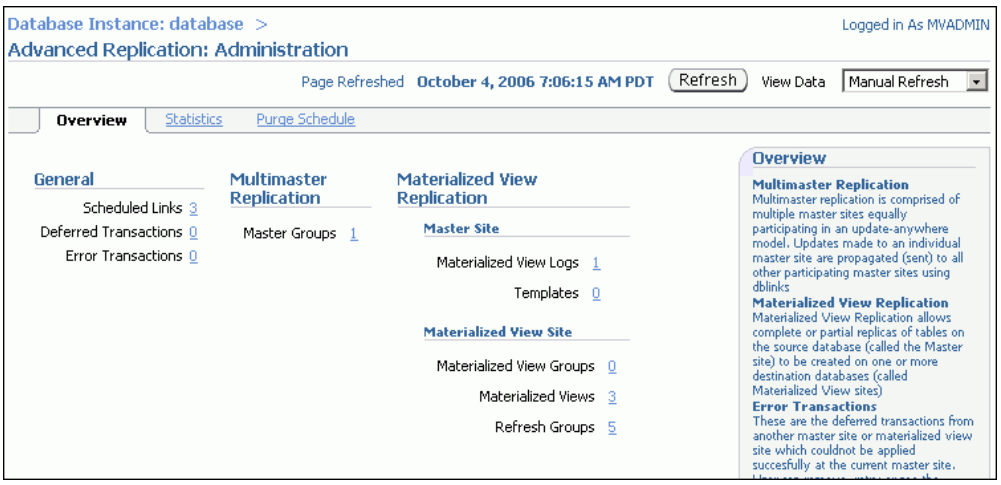
Viewing an Overview of the Replication Components at a Database

The Advanced Replication: Administration page in Enterprise Manager provides an overview of the replication components at a database.

To view an overview of the replication components at a database:

1. In Enterprise Manager, log in to the materialized view site as the materialized view administrator. The default user name for the materialized view administrator is `mvadmin`.
2. Go to the Database Home page.
3. Click **Data Movement** to open the Data Movement subpage.
4. Click **Manage** in the Advanced Replication section.

   The Advanced Replication: Administration page appears, showing the Overview subpage.



Each replication component is associated with a number that shows the quantity of the component at the current database. Click a number to view more information about these components and to manage these components.

If you are viewing a materialized view site, then you can monitor the following components:

- **Scheduled Links** in the General section shows the number of scheduled links at the materialized view site. Scheduled links are used to push deferred transactions automatically. See "About Scheduled Links and Deferred Transactions" on page 7-15.
- **Materialized View Groups** in the Materialized View Site section shows the number of materialized view groups at the materialized view site. Organizing materialized views into groups makes it easier to manage them. In addition, materialized views must be placed in a materialized view group for them to be updatable. See "About Replication Groups and Updatable Materialized Views" on page 7-13 for information about materialized view groups.
- **Materialized Views** in the Materialized View Site section shows the number of materialized views at the materialized view site. See "About Materialized View Replication" on page 7-1 for information about materialized views.

- **Refresh Groups** in the Materialized View Site section shows the number of refresh groups at the materialized view site. When a refresh group is refreshed, the materialized views in the refresh group are consistent to the same point in time. See "About Refresh Groups" on page 7-3.

If you are viewing a master site for one or more materialized view sites, then you can monitor the following components:

- **Master Groups** in the Multimaster Replication section shows the number of master groups at the master site. A materialized view group at a materialized view site can be based on a master group. See "About Replication Groups and Updatable Materialized Views" on page 7-13 for information about master groups and materialized view groups.
- **Materialized View Logs** in the Master Site section shows the number of materialized view logs at a master site. Materialized view logs enable a fast refresh of materialized views. See "About Materialized View Refresh" on page 7-2 for information about materialized view logs.
- **Error Transactions** in the General section shows the number of error transactions at the master site. Error transactions can occur when updatable materialized views are refreshed and conflicts are encountered that cannot be resolved. See "About Conflicts and Updatable Materialized Views" on page 7-16.

Click **Help** for information about the other replication components listed on the Overview subpage of the Advanced Replication: Administration page.

See Also:

- Chapter 7, "Replicating Data Using Materialized Views"
- "Managing a Materialized View Replication Environment" on page 8-1

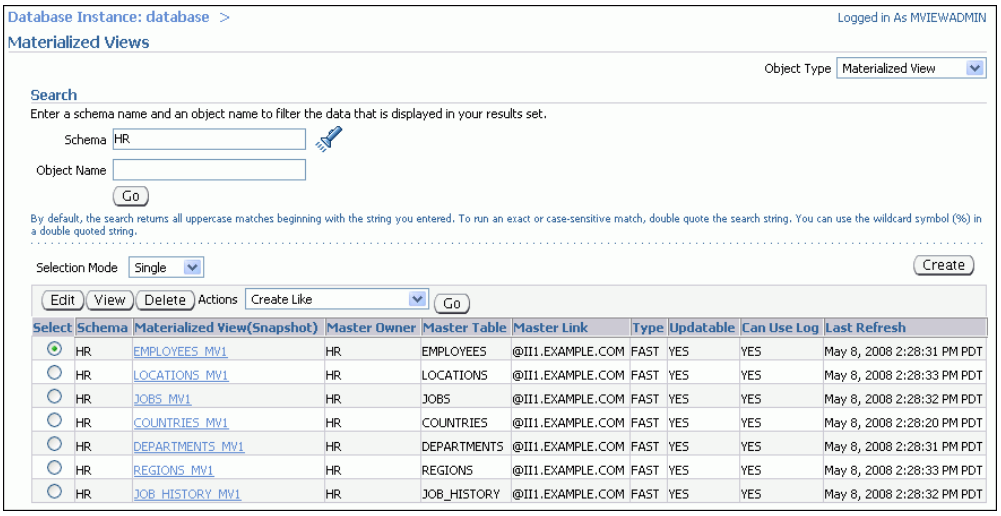
Viewing Information About Materialized Views

A materialized view contains a complete or partial copy of a table from a single point in time. In a replication environment, a materialized view is typically in a different database than the table on which it is based.

To view information about materialized views:

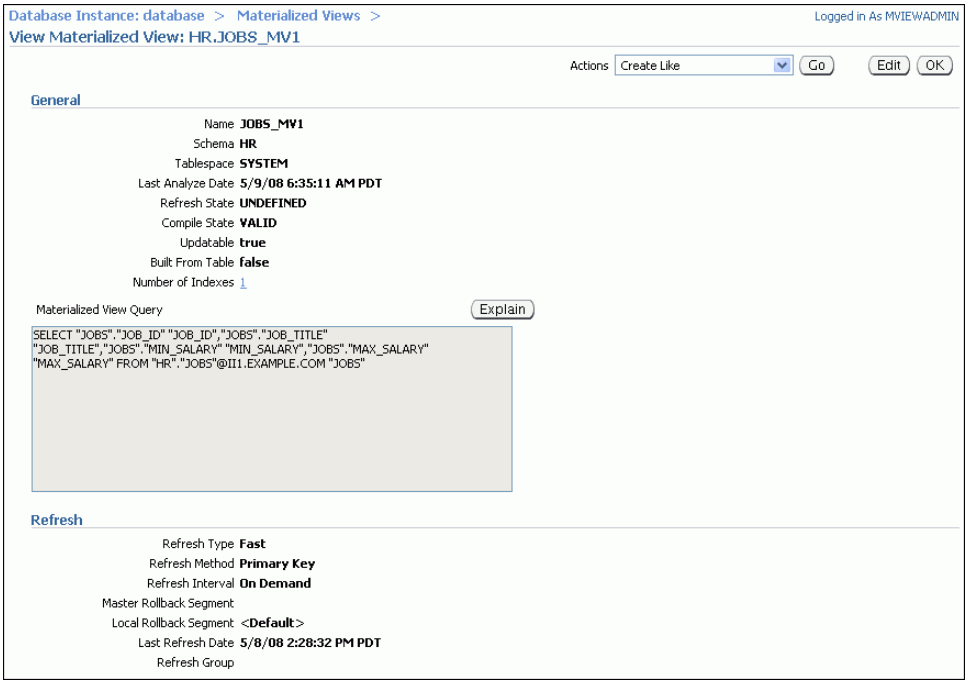
1. In Enterprise Manager, log in to the materialized view site as the materialized view administrator. The default user name for the materialized view administrator is `mvadmin`.
2. Go to the Database Home page.
3. Click **Data Movement** to open the Data Movement subpage.
4. Click **Manage** in the Advanced Replication section.

   The Advanced Replication: Administration page appears, showing the Overview subpage.
5. Click the number associated with **Materialized Views** in the Materialized View Site section to open the Materialized Views page.
6. Use the search tool to list the materialized views that you want to view. To list all materialized views, leave the **Schema** and **Object Name** fields blank and click **Go**.



The Materialized Views page contains important information about each materialized view listed, including its master table, master site (in the **Master Link** columns), and the last time it was refreshed. Click **Help** for information about the columns on this page.

7. For detailed information about a materialized view, select it and click **View** to open the View Materialized View page.



The View Materialized View page contains detailed information about the materialized view. The information is organized into the following sections: General, Refresh, Storage, Index Storage, Options, and Master Information. Click **Help** for information about the sections on this page.

Note: You can also query the `ALL_MVIEWS` data dictionary view for information about materialized views.

See Also:

- "About Materialized View Replication" on page 7-1
- "Managing a Materialized View Replication Environment" on page 8-1

Determining Which Materialized Views Are Currently Refreshing

Refreshing a materialized view synchronizes data in the materialized view with data in its master table. You can query the V$MVREFRESH dynamic performance view to determine which materialized views are currently refreshing.

Before shutting down a database that contains materialized views, it is best to check if any materialized views are currently refreshing. If any materialized views are refreshing, then wait until all of the refreshes are complete before shutting down the database.

To determine which materialized views at a materialized view site are currently refreshing:

1. Open SQL*Plus and connect to the materialized view site as the materialized view administrator.

   See *Oracle Database 2 Day DBA* for more information about starting SQL*Plus.

2. Run the following query:

```
COLUMN CURRMVOWNER HEADING 'Owner' FORMAT A15
COLUMN CURRMVNAME HEADING 'Materialized|View' FORMAT A25

SELECT CURRMVOWNER, CURRMVNAME FROM V$MVREFRESH;
```

   The output will be similar to the following:

```
                Materialized
Owner           View
--------------- -------------------------
HR              COUNTRIES_MV
HR              EMPLOYEES_MV
```

The V$MVREFRESH dynamic performance view does not contain information about updatable materialized views when the deferred transactions of the materialized views are being pushed to their masters.

See Also:

- "About Materialized View Refresh" on page 7-2
- "Refreshing Materialized Views" on page 8-2

Viewing Information About Materialized View Groups

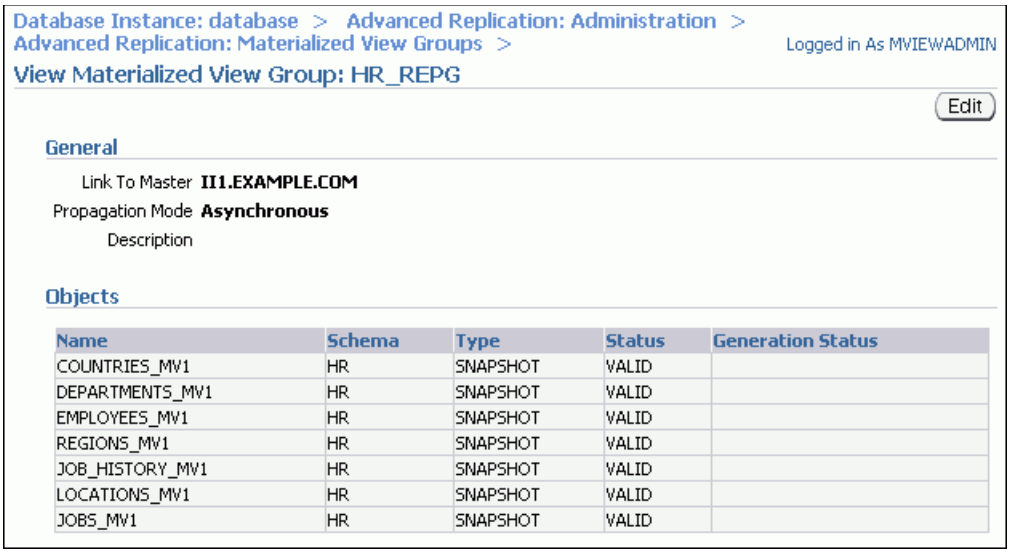
Materialized view groups make it easy to manage materialized views that are logically related. For example, you might add all of the materialized views in a schema or all of the materialized views used by an application to a materialized view group. Also, for materialized views to be updatable, they must belong to a materialized view group.

To view information about the materialized view groups at a materialized view site:

1. In Enterprise Manager, log in to the materialized view site as the materialized view administrator. The default user name for the materialized view administrator is `mvadmin`.
2. Go to the Database Home page.
3. Click **Data Movement** to open the Data Movement subpage.
4. Click **Manage** in the Advanced Replication section.

   The Advanced Replication: Administration page appears, showing the Overview subpage.
5. Click the number associated with **Materialized View Groups** in the Materialized View Site section.

   The Materialized View Groups page appears, listing the names of the materialized view groups at the materialized view site.
6. To view detailed information about a materialized view group, click the name of the materialized view group to open the View Materialized View Group page.



The View Materialized View Group page includes the following information:

- In the **Link To Master** field, the name of the database link to the master site. The master site contains the master tables of the materialized views in the materialized view group.
- In the **Propagation Mode** field, **Asynchronous** is displayed because the propagation sends messages according to a propagation schedule.
- In the **Description** field, a description of the materialized view group if the group has a description.
- In the Objects section, a list of the database objects in the materialized view group and information about each database object

Click **Help** for more information about this page.

Note: You can also query the following data dictionary views for information about materialized view groups:

- `ALL_REPGROUP`
- `ALL_REPSITES`

See Also:

- "About Replication Groups and Updatable Materialized Views" on page 7-13
- "About Master Sites, Master Tables, and Materialized View Sites" on page 7-2
- "Replicating Read/Write Data Using Materialized Views" on page 7-12

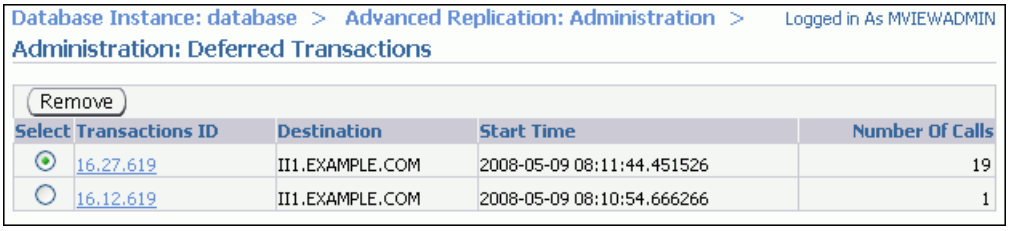
Viewing Information About Deferred Transactions for Updatable Materialized Views

At a materialized view site, deferred transactions include changes to updatable materialized views. The deferred transactions are stored at the materialized view site so that they can be sent to the master site and applied to master tables. Deferred transactions for a materialized view are always pushed when a materialized view is refreshed. A scheduled link determines when the deferred transactions are pushed to the master site independent of the refresh process.

To view information about deferred transactions for updatable materialized views:

1. In Enterprise Manager, log in to the materialized view site as the materialized view administrator. The default user name for the materialized view administrator is `mvadmin`.
2. Go to the Database Home page.
3. Click **Data Movement** to open the Data Movement subpage.
4. Click **Manage** in the Advanced Replication section.

   The Advanced Replication: Administration page appears, showing the Overview subpage.
5. Click the number associated with **Deferred Transactions** in the General section to open the Administration: Deferred Transactions page.

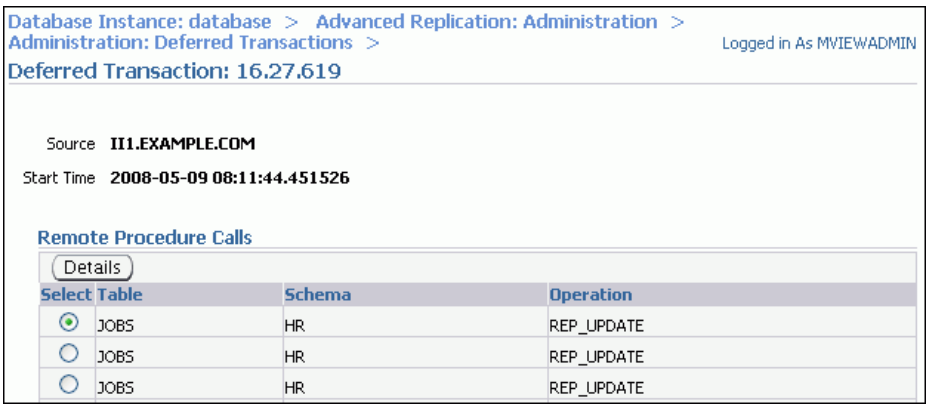


The Administration: Deferred Transactions page contains the transaction ID and the following information for each deferred transaction:

- The **Destination** field shows the master site to which the deferred transaction will be pushed.
- The **Start Time** field shows when the deferred transaction was placed in the deferred transaction queue at the materialized view site.
- The **Number of Calls** field typically shows the number of individual rows changed in the transaction.

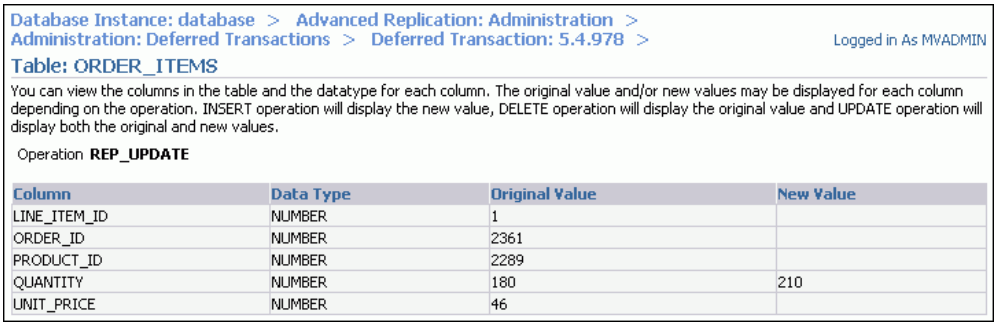
Click **Help** for more information about this page.

6. Click a transaction ID in the **Transaction ID** column to open the Deferred Transaction page and view more information about the transaction.



The Deferred Transaction page shows the materialized view and the type of operation that changed it. For example, `REP_UPDATE` designates an update operation.

7. To view detailed information about a specific change in the deferred transaction, select the change and click **Details**.



This page shows each column in the materialized view, along with its original value and, where appropriate, its new value. When there is a new value for a column, then the change modified the column value from the original value to the new value.

See Also:

- "About Scheduled Links and Deferred Transactions" on page 7-15
- "About Replication Groups and Updatable Materialized Views" on page 7-13
- "Replicating Read/Write Data Using Materialized Views" on page 7-12
- *Oracle Database Advanced Replication Management API Reference* for information about the data dictionary views related to deferred transactions

Viewing Information About Refresh Groups

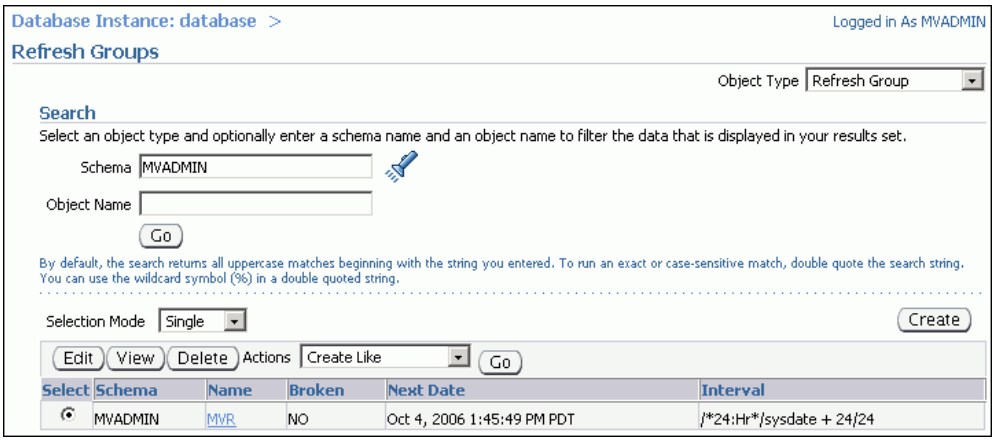
When you refresh a refresh group, all of the materialized views in the refresh group are refreshed to a consistent point in time.

To view information about the refresh groups at a materialized view site:

1. In Enterprise Manager, log in to the materialized view site as the materialized view administrator. The default user name for the materialized view administrator is `mvadmin`.
2. Go to the Database Home page.
3. Click **Data Movement** to open the Data Movement subpage.
4. Click **Manage** in the Advanced Replication section.

   The Advanced Replication: Administration page appears, showing the Overview subpage.
5. Click the number associated with **Refresh Groups** in the Materialized View Site section.

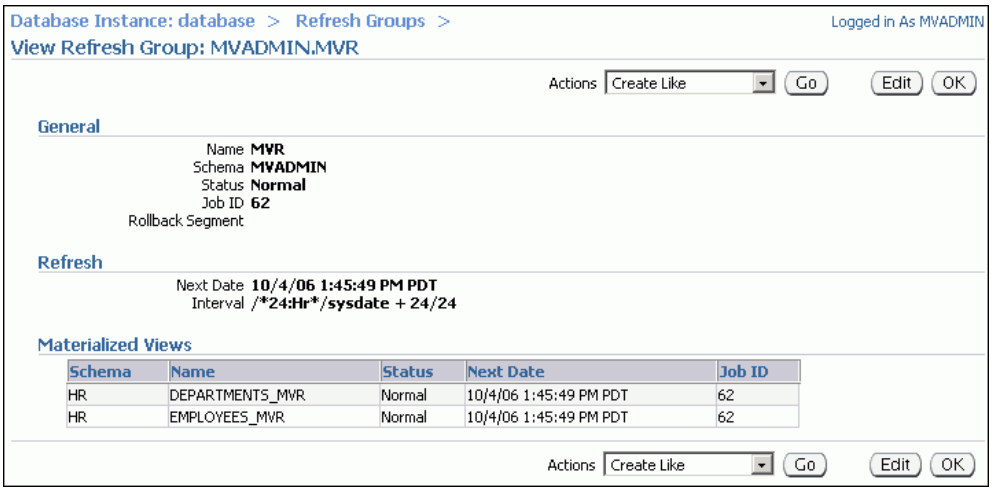
   The Refresh Groups page appears.
6. Use the search tool to list the refresh groups that you want to view. To list all refresh groups, ensure that the **Schema** and **Object Name** fields are blank, and click **Go**.



The Refresh Groups page lists the names of the refresh groups at the materialized view site. It also contains the following information for each refresh group:

- The **Broken** column shows whether the refresh job is working properly. **No** means that the job is working properly. **Yes** means that the job is broken. If 16 attempts to refresh a refresh group fail, then the refresh job is considered broken. If the refresh job is broken, see "Correcting Problems with Materialized View Refresh" on page 8-21.
- The **Next Date** column shows the next date and time when the refresh group will be automatically refreshed.
- The **Interval** column shows the time interval between automatic refreshes of the refresh group.

7. For detailed information about a refresh group, select it and click **View** to open the View Refresh Group page.



The View Refresh Group page contains detailed information about the refresh group. The information is organized into the following sections: General, Refresh, and Materialized Views. Click **Help** for information about all of the sections on this page.

Note: You can also query the `ALL_REFRESH` data dictionary view for information about refresh groups.

See Also:

- "About Materialized View Refresh" on page 7-2
- "About Refresh Groups" on page 7-3
- "Configuring a Refresh Group" on page 7-23
- "Refreshing a Refresh Group" on page 8-2

Viewing Materialized View Logs at a Master Site

Materialized view logs keep track of changes to master tables so that a fast refresh of materialized views is possible.

To view information about the materialized view logs at a master site:

1. In Enterprise Manager, log in to the master site as an administrative user, such as the replication administrator or `SYSTEM`. By default, the user name of the replication administrator is `repadmin`.

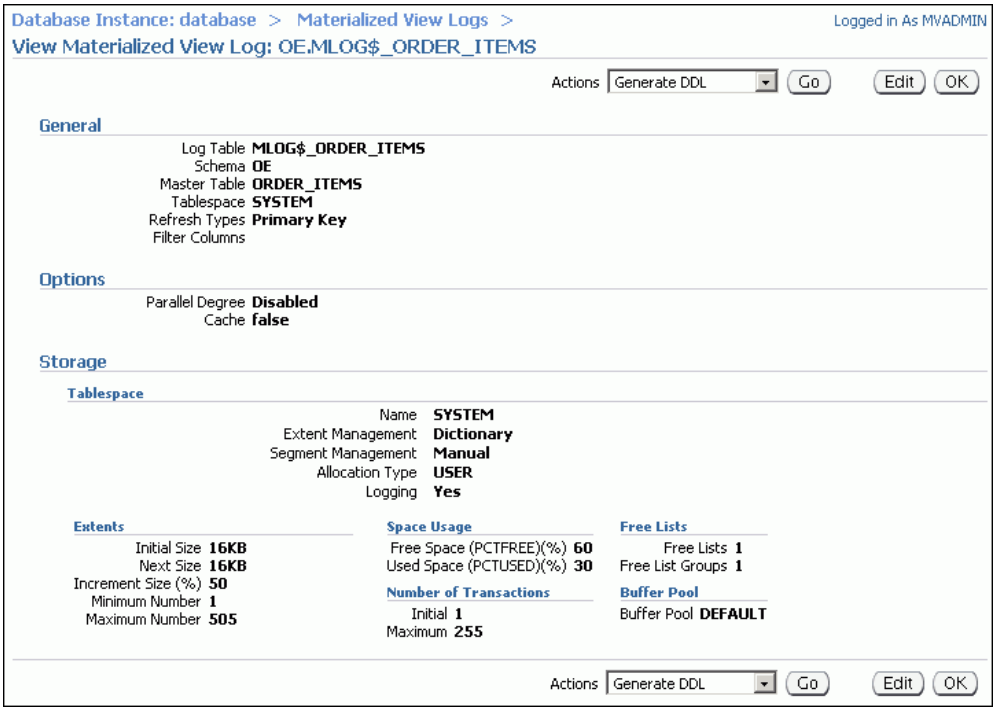
2. Click **Data Movement** to open the Data Movement subpage.
3. Click **Manage** in the Advanced Replication section.

   The Advanced Replication: Administration page appears, showing the Overview subpage.
4. Click the number associated with **Materialized View Logs** in the Master Site section.

   The Materialized View Logs page appears.
5. Use the search tool to list the materialized view logs that you want to view.

   To list all materialized view logs, ensure that the **Schema** and **Object Name** fields are blank, and click **Go**.

   The Materialized View Logs page lists the schema, underlying log table, and master table for each materialized view log listed.
6. For detailed information about a materialized view log, select it and click **View** to open the View Materialized View Log page.



The View Materialized View Log page contains detailed information about the materialized view log. The information is organized into the following sections: General, Options, and Storage. Click **Help** for information about the sections on this page.

Note: You can also query the `ALL_MVIEW_LOGS` data dictionary view for information about materialized view logs.

See Also:

- "About Materialized View Refresh" on page 7-2
- "Configuring Materialized View Logs at the Master Site" on page 7-6
- "Preventing Materialized View Logs From Becoming Too Large" on page 8-22
- "Tutorial: Cleaning Up Materialized View Support at a Master Site" on page 8-8

Viewing the Materialized Views for a Master Site

When you configure a materialized view at a materialized view site, it registers the materialized view at the master site. You can query the `ALL_REGISTERED_MVIEWS` data dictionary view to show the following information about each materialized view registered at a master site:

- The owner of the materialized view
- The name of the materialized view
- The materialized view site that contains the materialized view
- Whether the materialized view is updatable or read-only

To show this information:

1. Open SQL*Plus and connect to the master site as an administrative user, such as the replication administrator or `SYSTEM`. By default, the user name of the replication administrator is `repadmin`.

   See *Oracle Database 2 Day DBA* for more information about starting SQL*Plus.

2. Run the following query:

```
COLUMN OWNER HEADING 'Owner' FORMAT A20
COLUMN NAME HEADING 'Name' FORMAT A20
COLUMN MVIEW_SITE HEADING 'Materialized|View Site' FORMAT A15
COLUMN UPDATABLE HEADING 'Updatable?' FORMAT A10

SELECT OWNER, NAME, MVIEW_SITE, UPDATABLE
  FROM ALL_REGISTERED_MVIEWS;
```

   The output will be similar to the following:

```
                                    Materialize
Owner                Name           View Site       Updatable?
-------------------- -------------------- --------------- ----------
HR                   COUNTRIES_MV        II2.EXAMPLE.COM YES
HR                   DEPARTMENTS_MV      II2.EXAMPLE.COM YES
HR                   EMPLOYEES_MV        II2.EXAMPLE.COM YES
HR                   JOBS_MV             II2.EXAMPLE.COM YES
HR                   JOB_HISTORY_MV      II2.EXAMPLE.COM YES
HR                   LOCATIONS_MV        II2.EXAMPLE.COM YES
HR                   REGIONS_MV          II2.EXAMPLE.COM YES
```

See Also:

- "About Master Sites, Master Tables, and Materialized View Sites" on page 7-2
- "About Replication Groups and Updatable Materialized Views" on page 7-13

Troubleshooting a Materialized View Replication Environment

This section describes the most common problems in a materialized view replication environment. It also describes how to correct these problems.

The following topics describe troubleshooting a materialized view replication environment:

- Correcting Problems with Materialized View Refresh
- Preventing Materialized View Logs From Becoming Too Large

See Also:

- *Oracle Database Advanced Replication*
- Chapter 7, "Replicating Data Using Materialized Views"
- "Managing a Materialized View Replication Environment" on page 8-1
- "Monitoring a Materialized View Replication Environment" on page 8-9

Correcting Problems with Materialized View Refresh

If one or more materialized views at a materialized view site cannot refresh, then the following are typical causes and solutions:

- A network problem is preventing one or more materialized views from refreshing. If you have multiple refresh groups and multiple materialized views at a materialized view site, and none of them can refresh, then a network problem is the most likely cause. In this case, correct the network problem, and refresh the relevant materialized views. See *Oracle Database 2 Day DBA* for information about configuring network connectivity between databases.
- A master site for the materialized views is down. A materialized view cannot refresh if its master site is not open. If you have a materialized view site with materialized views that use multiple master sites, and only the materialized views for a particular master site cannot refresh, then a master site problem is the most likely cause. In this case, check the database at the master site. If it is down, then start it, and refresh the relevant materialized views. See *Oracle Database 2 Day DBA* for information about starting a database.
- A database link is broken. If all of the materialized views in a materialized view group cannot refresh, then the database link used to refresh the materialized views might be broken. If a database link is broken, then you can correct the problem by deleting the database link and re-creating it.

To re-create a broken database link:

1. Log in to Enterprise Manager as an administrative user, such as `SYSTEM`.
2. Go to the Database Home page.

3. Click **Schema** to open the Schema subpage.
4. Click **Database Links** in the Database Objects section.

   The Database Links page appears.
5. Use the search tool to list the database link used by the materialized views.

   The database link name should be the same as the global name of the master site for the materialized views.
6. Select the database link.
7. Click **View**.
8. On the View Database Link page, click **Test** to see if the database link is active. If it is active, then the materialized view refresh problems are caused by a different issue. If the database link is not active, then follow Steps 9 through 13 to delete it and re-create it.
9. Click **OK** to return to the Database Links page.
10. Select the database link.
11. Click **Delete**.
12. Click **Yes** on the confirmation page to delete the database link.
13. Follow the instructions in "Tutorial: Creating a Database Link" on page 2-8 to re-create the database link.

See Also:

- "About Materialized View Refresh" on page 7-2
- "Refreshing Materialized Views" on page 8-2
- *Oracle Database Advanced Replication*

Preventing Materialized View Logs From Becoming Too Large

Materialized view logs keep track of changes to master tables at master sites. When a fast refresh is performed on a materialized view, the changes in the materialized view log are applied to the materialized view. After changes are applied to all of the materialized views that use the materialized view log, the changes are deleted from the materialized view log.

A materialized view log can become too large when one or more materialized views that use the materialized view log do not refresh for a relatively long period of time. This situation can slow refresh performance for other materialized views. The following are typical causes and solutions:

- A network problem is preventing one or more materialized views that use the materialized view log from refreshing. In this case, correct the network problem and refresh the relevant materialized views. See *Oracle Database 2 Day DBA* for information about configuring network connectivity between databases.
- A materialized view was dropped at a materialized view site when the materialized view site was not connected to the network. When a materialized view is dropped and a network connection is available, information about the materialized view is removed from the master site automatically. However, when a network connection is not available, information about the materialized view remains at the master site and must be cleaned up manually. See "Tutorial: Cleaning Up Materialized View Support at a Master Site" on page 8-8.

- Materialized views are not being refreshed often enough. Materialized views can be refreshed manually, or they can be refreshed automatically when they are in a refresh group.

  If you refresh materialized views manually, then you might need to refresh them more often if materialized view logs are becoming too large. See "Refreshing a Materialized View" on page 8-3.

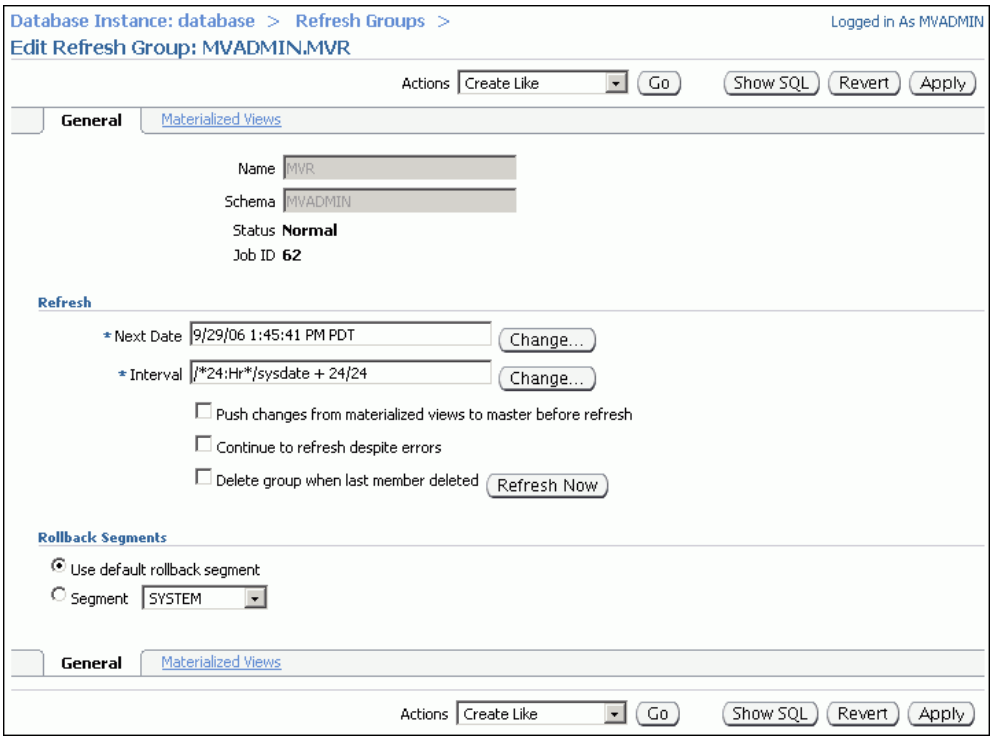
  If materialized views are in a refresh group that is not refreshing often enough, then you can make the refresh interval shorter for the refresh group.

To modify the refresh interval for a refresh group:

1. In Enterprise Manager, log in to the materialized view site as the materialized view administrator. The default user name for the materialized view administrator is `mvadmin`.
2. Go to the Database Home page.
3. Click **Data Movement** to open the Data Movement subpage.
4. Click **Manage** in the Advanced Replication section.

   The Advanced Replication: Administration page appears, showing the Overview subpage.
5. Click the number associated with **Refresh Groups** in the Materialized View Site section.
6. On the Refresh Groups page, use the search tool to list the refresh group that you want to modify.
7. Select the refresh group.
8. Click **Edit**.

   The Edit Refresh Group page appears, showing the General subpage.



9. In the Refresh section, click **Change** for the **Interval** field.
10. Shorten the interval using the **Every** field and list.
11. Click **OK**.
12. On the Edit Refresh Group page, click **Apply** to save your changes.

Note: You can also use the `DBMS_REFRESH.CHANGE` procedure to change the refresh interval for a refresh group.

See Also:

- "About Materialized View Refresh" on page 7-2
- "About Refresh Groups" on page 7-3
- "About Master Sites, Master Tables, and Materialized View Sites" on page 7-2
- "Configuring Materialized View Logs at the Master Site" on page 7-6
- *Oracle Database Advanced Replication*

9

Sending Messages Using Oracle Streams Advanced Queuing

This chapter describes how to configure an Oracle Streams Advanced Queuing (AQ) environment that sends messages between databases and applications. It also describes administering, monitoring, and troubleshooting a messaging environment once it is in place.

This chapter contains the following sections:

About Messaging

A messaging environment stores information in queues. **Enqueuing** is the process of placing messages into queues. **Dequeuing** is the process of retrieving messages from queues.

The information in queues can be used to complete tasks, or it can be processed by applications. A messaging environment allows applications to communicate with each other asynchronously. That is, one application does not need to wait for another application to complete a particular task. Asynchronous communication means that a messaging system has minimal impact on the functionality of the applications that use the system.

For example, when one application wants to communicate with another application, it can put messages in a queue. The messages can be stored in the queue until the other application retrieves them. In fact, one application might not be running while another application is enqueuing messages for it to process later. The messages might instruct the retrieving application to perform an action, or the messages might contain information that must be processed by the retrieving application.

When an organization has a number of different systems that must communicate with each other, a messaging environment can be a good solution. The various systems might be in different locations, some might be older than others, and some might run on different platforms. Messaging provides a standard, reliable way to transport critical information between these systems.

The messaging feature of Oracle Database is called Oracle Streams Advanced Queuing (AQ). Oracle Streams AQ provides the advantages of messaging, and it also integrates the messaging system with Oracle Database. Therefore, Oracle Streams AQ provides the reliability, scalability, security, and manageability of Oracle Database.

Oracle Streams AQ can be used to configure a messaging environment that sends messages between queues in a single database or between queues in different databases. A messaging environment that sends messages between queues includes the following components:

- **Queues:** Abstract storage units that store messages in a messaging environment
- **Producers:** Users or applications that enqueue messages
- **Propagations:** Oracle Scheduler jobs that copy messages from one queue to another according to a set schedule
- **Consumers:** Users or applications that dequeue messages

A single user or application can act as both a producer and a consumer. In a messaging environment, both **subscribers** and **messaging clients** are consumers. Both subscribers and messaging clients are mechanisms that are authorized to dequeue messages from a queue, and both can use rules to determine which messages to dequeue.

Oracle Streams AQ provides many features, some of which are beyond the scope of this guide. The following topics provide an overview of some of the features of Oracle Streams AQ:

- About Message Ordering
- About Message Modes
- About Message Notifications
- About Propagations
- About Oracle Messaging Gateway

See Also:

- "When to Send Messages Between Databases" on page 1-6
- *Oracle Streams Concepts and Administration* for more information about messaging clients
- *Oracle Streams Advanced Queuing User's Guide* for detailed information about subscribers and all of the other features of Oracle Streams AQ

About Message Ordering

Message ordering determines the order in which messages are dequeued. Oracle Streams Advanced Queuing (AQ) provides the following options for message ordering:

- **Enqueue time:** Messages are dequeued in the same order that they were enqueued. This option is sometimes called first in first out (FIFO) ordering. For example, banking applications often use this ordering so that financial transactions are always ordered according to the time when they occurred.
- **Priority:** A priority is specified for messages during enqueuing, and messages are dequeued based on their priority. For example, shipping companies might use this ordering for messages because some packages have higher priority than others.
- **Commit time:** Messages are dequeued based on when the transaction that enqueued the messages committed. Commit-time ordering is typically used when the messages result in database changes with transaction dependencies. For example, an application that handles sales for a company might use commit-time ordering when the messages result in changes to a number of database tables that have dependencies.

About Message Modes

Oracle Streams Advanced Queuing (AQ) supports the following **message modes**:

- **Persistent messaging:** Messages are always stored on disk in a database table called a **queue table**. This type of storage is sometimes called **persistent queue** storage.
- **Buffered messaging:** Messages are stored in memory but can spill to a queue table under certain conditions. This type of storage is sometimes called **buffered queue** storage.

Buffered messaging provides better performance, but it does not support some messaging features, such as message retention. Message retention lets you specify the amount of time a message is retained in the queue table after being dequeued. In addition, when a database goes down unexpectedly, persistent messages are retained on disk, but buffered messages can be lost because they are stored in memory.

Message retention provides an audit trail, which might be required for some organizations. For example, while messages are stored in the queue table, users and applications can run queries to gather information about the messages. For example, a sales application might gather this information to generate a report on orders.

See Also:

- *Oracle Streams Advanced Queuing User's Guide* for information about the differences between persistent messaging and buffered messaging

About Message Notifications

Oracle Streams Advanced Queuing (AQ) can notify an application and users when a message of interest is enqueued. **Notifications** enable applications and users to do their work without constantly checking a queue for new messages. Notifications can be sent to an application using PL/SQL, Java Message Service (JMS), or Oracle Call Interface (OCI) callback functions. Notification can also be sent to a specified e-mail address or HTTP post. In addition, notifications can be presented in either `RAW` data type form or XML form. Applications and users do not need to be connected to a database to receive notifications. When they are notified that a message of interest has appeared in a queue, they can connect to the database and check the queue.

See Also:

- "Preparing for Messaging" on page 9-5
- "Tutorial: Configuring Message Notifications" on page 9-16

About Propagations

Propagation is the process of sending messages from one queue to another. These queues can be in the same database or in different databases. Propagations are not a required part of a messaging system. Propagations enable applications to communicate with each other even if they are not connected to the same database or same queue. Also, using multiple propagations, you can send the same messages from one queue to several other queues.

Propagations use database links to send messages between queues at different databases. A **database link** is a pointer that defines a one-way communication path from an Oracle database to another database. Propagation is performed by an Oracle Scheduler job called a **propagation job**. A propagation schedule determines how often a propagation sends messages, and you control the schedule for each propagation job.

Oracle Streams Advanced Queuing (AQ) supports both queue-to-queue or queue-to-database link (queue-to-dblink) propagations. A **queue-to-queue propagation** always has its own exclusive propagation job to send messages from the source queue to the destination queue. Therefore, the propagation schedule of each queue-to-queue propagation can be managed separately. A single database link can be used by multiple queue-to-queue propagations. Use queue-to-queue propagation when you want fine-grained control over multiple propagations that send messages from a single queue, and each propagation should use a different schedule.

In contrast, a **queue-to-dblink propagation** shares a propagation job with other queue-to-dblink propagations from the same source queue that use the same database link. If several propagations send messages from a single source queue to several queues at the remote database, then these propagations share the same propagation schedule. Any change to the propagation schedule affects all of the queue-to-dblink propagations from the same source queue that use the database link. Use queue-to-dblink propagation when you want bulk control over multiple propagations that send messages from a single queue, and all of the propagations should use the same schedule.

See Also:

- "Preparing for Messaging" on page 9-5
- "Tutorial: Sending Messages Between Oracle Databases" on page 9-6
- "Modifying Propagations" on page 9-28
- "Tutorial: Creating a Database Link" on page 2-8
- *Oracle Streams Concepts and Administration*

About Oracle Messaging Gateway

Oracle Messaging Gateway integrates an Oracle Streams Advanced Queuing (AQ) messaging system with other messaging systems, such as IBM Websphere MQ (formerly called MQSeries) and TIBCO Rendezvous. This integration enables Oracle Database applications that use Oracle Streams AQ to communicate bidirectionally and seamlessly with other applications that use non-Oracle messaging systems.

Messaging Gateway uses a gateway agent to send messages from Oracle Streams AQ queues to queues in non-Oracle messaging systems. The gateway agent also enables Oracle Streams AQ queues to receive messages from non-Oracle messaging systems.

Messaging Gateway supports automatic type conversion for Oracle Streams AQ types that can be directly mapped to a non-Oracle messaging system. For types that cannot be directly mapped, you can define custom transformations to map the types. Once defined, custom transformations are performed automatically during propagation.

Messaging Gateway supports the native message formats of the messaging systems. Oracle Streams AQ messages can have `RAW` or any Oracle object type payload supported by Oracle Streams AQ. IBM Websphere MQ messages can be text or byte messages. TIBCO Rendezvous messages can be any TIBCO Rendezvous wire format data type.

Oracle Streams AQ queues can be securely accessed through the Internet. Messaging Gateway provides secure, Internet-enabled messaging between Oracle Streams AQ queues and queues in a non-Oracle messaging system.

See Also:

- *Oracle Streams Advanced Queuing User's Guide* for detailed information about the Oracle Messaging Gateway and for information about how to use it

Preparing for Messaging

This topic describes actions that are required to prepare your databases for a messaging environment.

To prepare your databases for a messaging environment:

1. Set the `GLOBAL_NAMES` initialization parameter to `TRUE` at each database involved in the messaging environment. See "Setting the GLOBAL_NAMES Initialization Parameter to TRUE" on page 2-1 for instructions.
2. Configure network connectivity so that a database that sends messages can communicate with a database that receives messages. See *Oracle Database 2 Day DBA* for information about configuring network connectivity between databases.
3. Configure an Oracle Streams administrator at each database involved in the messaging environment. See "Tutorial: Creating an Oracle Streams Administrator" on page 2-2 for instructions.
4. Ensure that the Oracle Streams pool is large enough to accommodate the queues created for the messaging environment. Each queue requires at least 10 MB of memory. The Oracle Streams pool is part of the System Global Area (SGA). You can manage the Oracle Streams pool by setting the `MEMORY_TARGET` initialization parameter (Automatic Memory Management), the `SGA_TARGET` initialization parameter (Automatic Shared Memory Management), or the `STREAMS_POOL_SIZE` initialization parameter. See *Oracle Streams Concepts and Administration* for more information about the Oracle Streams pool.

See Also:

- "About Messaging" on page 9-1
- "Tutorial: Sending Messages Between Oracle Databases" on page 9-6
- "Tutorial: Configuring Message Notifications" on page 9-16

Tutorial: Sending Messages Between Oracle Databases

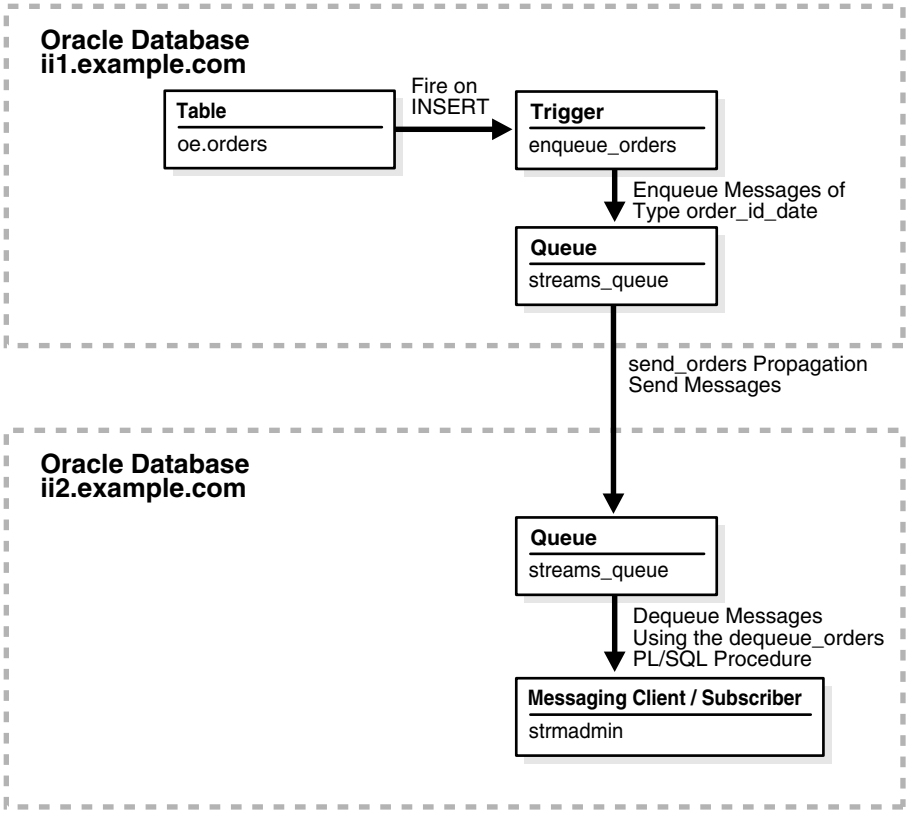
The example in this topic involves a business that enters orders in an Oracle database at headquarters. When an order is entered, the business wants to use a messaging system to send the order ID and order date to an Oracle database in a warehouse in a different location. These messages alert employees at the warehouse about the orders so that they can fill and ship them. The employees at the warehouse have access to the data at headquarters to find detailed information about any particular order.

This example uses the `oe` sample schema for its order-entry schema. The `oe` sample schema is installed by default with Oracle Database.

In this example, the global name of the database at headquarters is `ii1.example.com`, and the global name of the database at the warehouse is `ii2.example.com`. However, you can substitute any two databases in your environment to complete the example.

Figure 9–1 provides an overview of the messaging environment created in this example.

Figure 9–1 Sample Messaging Environment That Sends Messages Between Databases



Before you start this example, complete the tasks described in "Preparing for Messaging" on page 9-5.

To configure a messaging system that sends messages about orders from the ii1.example.com database to the ii2.example.com database:

- Task 1: Creating the Message Type at Each Database
- Task 2: Configuring the Queues and Propagation Between Them

- Task 3: Configuring a Message Enqueuing Mechanism
- Task 4: Configuring a Messaging Client to Dequeue Messages
- Task 5: Enqueuing Messages
- Task 6: Dequeuing Messages

See Also:

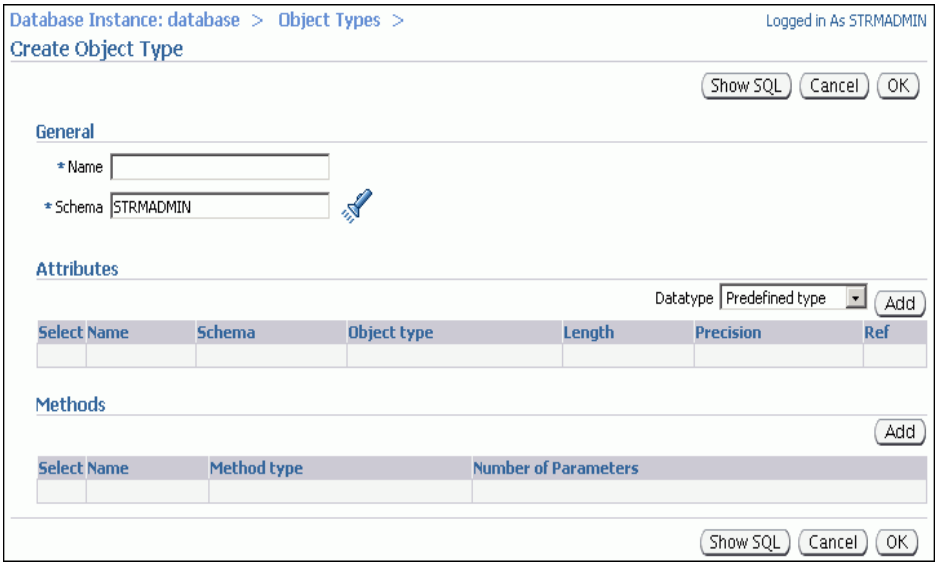
- "When to Send Messages Between Databases" on page 1-6

Task 1: Creating the Message Type at Each Database

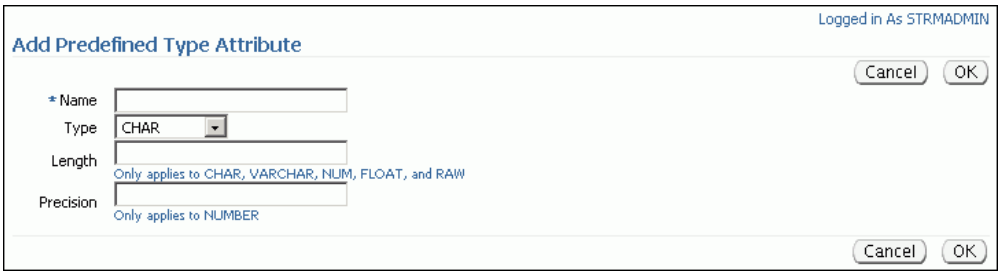
Create a user-defined type to define the information that you want to send about orders. This example creates the `strmadmin.order_id_date` type for messages that will contain the order ID and order date.

To create the strmadmin.order_id_date type at each database:

1. Prepare your environment for messaging if you have not already done so. See "Preparing for Messaging" on page 9-5.
2. In Oracle Enterprise Manager, log in to the `ii1.example.com` database as the Oracle Streams administrator.
3. Go to the Database Home page.
4. Click **Schema** to open the Schema subpage.
5. Click **Object Types** in the User Defined Types section.
6. On the Object Types page, click **Create** to open the Create Object Type page.



7. Enter `order_id_date` in the **Name** field.
8. Ensure that `strmadmin` is selected in the **Schema** field.
9. Ensure that **Predefined Type** is selected for **Datatype** in the Attributes section.
10. Click **Add** in the Attributes section to open the Add Predefined Type Attributes page.



11. To add an attribute:
    a. Enter `order_id` in the **Name** field.
    b. Select `NUMBER` for **Type**.
    c. Enter `12` in the **Length** field.
    d. Click **OK**.
12. On the Create Object Type page, ensure that **Predefined Type** is selected in the Attributes section.
13. Click **Add** in the Attributes section to open the Add Predefined Type Attributes page.
14. To add an attribute:
    a. Enter `order_date` in the **Name** field.
    b. Select `VARCHAR2` for **Type**.
    c. Enter `100` in the **Length** field.
    d. Click **OK**.
15. On the Create Object Type page, click **OK** to create the type.
16. Log in to the `ii2.example.com` database as the Oracle Streams administrator in Enterprise Manager.
17. Complete Steps 3 through 15 at the `ii2.example.com` database so that both databases have the `strmadmin.order_id_date` type.
18. Complete the steps in "Task 2: Configuring the Queues and Propagation Between Them" to continue this extended example.

Note: You can also use the `CREATE TYPE` SQL statement to create a type.

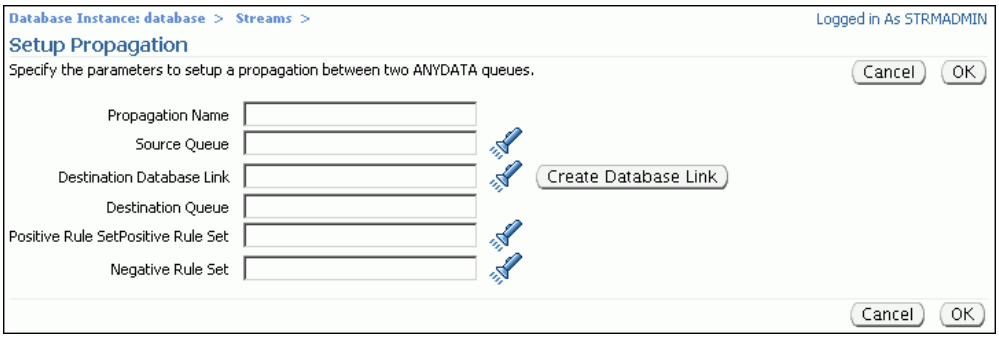
Task 2: Configuring the Queues and Propagation Between Them

Create a queue at each database, and create a propagation to send messages about orders from the queue at the `ii1.example.com` database to the queue at the `ii2.example.com` database.

To create a propagation:

1. Create a queue named `streams_queue` in the schema of the Oracle Streams administrator at both the `ii1.example.com` and `ii2.example.com` databases. See "Creating an ANYDATA Queue" on page 2-7 for instructions.

2. Create a database link named `ii2.example.com` in the Oracle Streams administrator schema at `ii1.example.com`. Configure the database link to connect to the Oracle Streams administrator schema at `ii2.example.com`. The service name of the database link must be `ii2.example.com`. See "Tutorial: Creating a Database Link" on page 2-8 for instructions.
3. In Oracle Enterprise Manager, log in to the `ii1.example.com` database as the Oracle Streams administrator.
4. Go to the Database Home page.
5. Click **Data Movement** to open the Data Movement subpage.
6. Click **Manage** in the Streams section.
7. On the Streams page, click **Propagation** to open the Propagation subpage.
8. Click **Setup Propagation** to open the Setup Propagation page.



9. Enter `send_orders` in the **Propagation Name** field.
10. Enter `strmadmin.streams_queue` in the **Source Queue** field. This queue is the queue at the `ii1.example.com` database into which messages about orders will be enqueued.
11. Enter the name of the database link that you created in Step 2 in the **Destination Database Link** field. In this example, the database link name is `ii2.example.com`.
12. Enter `strmadmin.streams_queue` in the **Destination Queue** field. This is the queue at the `ii2.example.com` database to which messages about orders will be sent.
13. Leave the **Positive Rule Set** and **Negative Rule Set** fields empty. When a propagation does not have a rule set, it sends all of the messages in the source queue to the destination queue.
14. Click **OK** to create the propagation.
15. Complete the steps in "Task 3: Configuring a Message Enqueuing Mechanism" to continue this extended example.

Note: You can also use the `DBMS_PROPAGATION_ADM.CREATE_PROPAGATION` procedure to create a propagation.

Task 3: Configuring a Message Enqueuing Mechanism

Configure a mechanism to enqueue the messages in your messaging system. Typically, an application creates and enqueues messages that will be dequeued and processed by another application. For simplicity, this example creates a trigger called `enqueue_orders` to enqueue a message that includes the order ID and order date of an order. The trigger fires when an order is inserted into the `oe.orders` table.

To configure a message enqueuing mechanism:

1. Grant the `EXECUTE` privilege on the `DBMS_STREAMS_MESSAGING` package to the Oracle Streams administrator.

   This example configures a trigger that runs the `ENQUEUE` procedure in the `DBMS_STREAMS_MESSAGING` package. The user who runs this procedure in a trigger must have explicit `EXECUTE` privilege on the package that contains the procedure. The privilege cannot be granted through a role. Therefore, the Oracle Streams administrator must be granted explicit `EXECUTE` privilege on the package.

   a. Log in to Enterprise Manager as an administrative user who can grant privileges to the `strmadmin` user. For example, you can log in as a user with `SYSDBA` privilege. In this example, log in to the `ii1.example.com` database.

   b. Go to the Database Home page.

   c. Click **Server** to open the Server subpage.

   d. Click **Users** in the Security section.

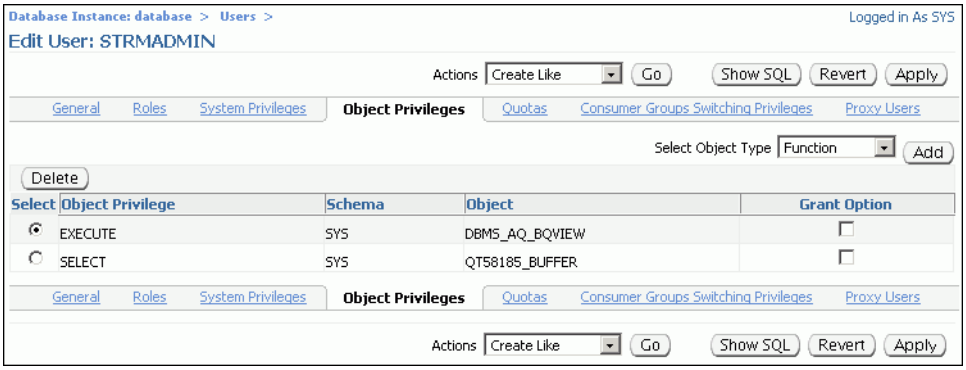
      The Users page appears.

   e. Select the `STRMADMIN` user.

   f. Click **Edit**.

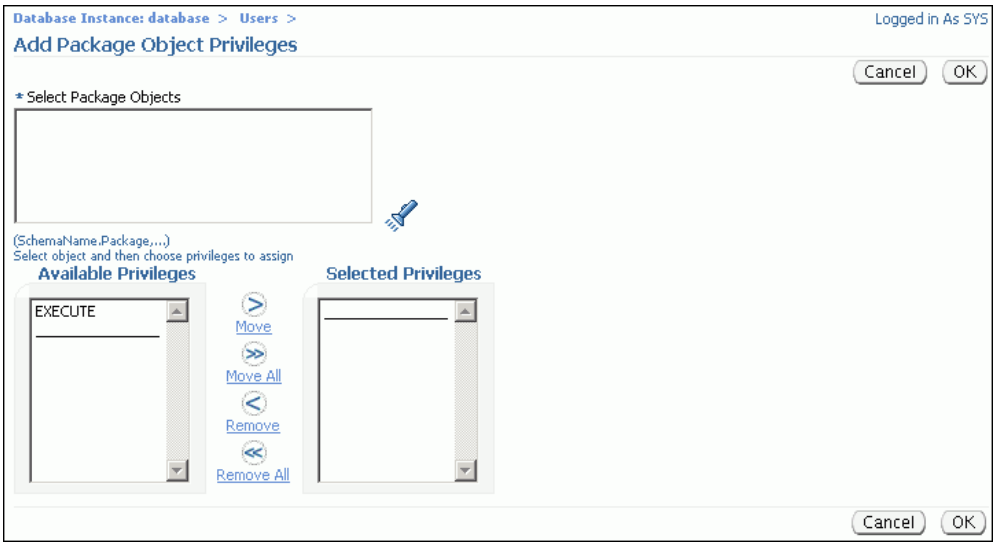
      The Edit User page appears, showing the General subpage.

   g. Click **Object Privileges** to open the Object Privileges subpage.



   h. Select **Package** in the Select Object Type list.

   i. Click **Add** to open the Add Package Object Privileges page.



j. Enter `SYS.DBMS_STREAMS_MESSAGING` in the **Select Package Objects** field.

k. Move **EXECUTE** from the Available Privileges list to the Selected Privileges list.

l. Click **OK** to add the privilege.

m. Click **Apply** to grant the privilege.

n. Log out of Enterprise Manager.

Note: You can also use the `GRANT` SQL statement to grant privileges to a user.

---

2. Create a trigger in the `ii1.example.com` database to enqueue a message automatically when a change is made to the `oe.orders` table.

   a. Log in to `ii1.example.com` database in Enterprise Manager as the Oracle Streams administrator.

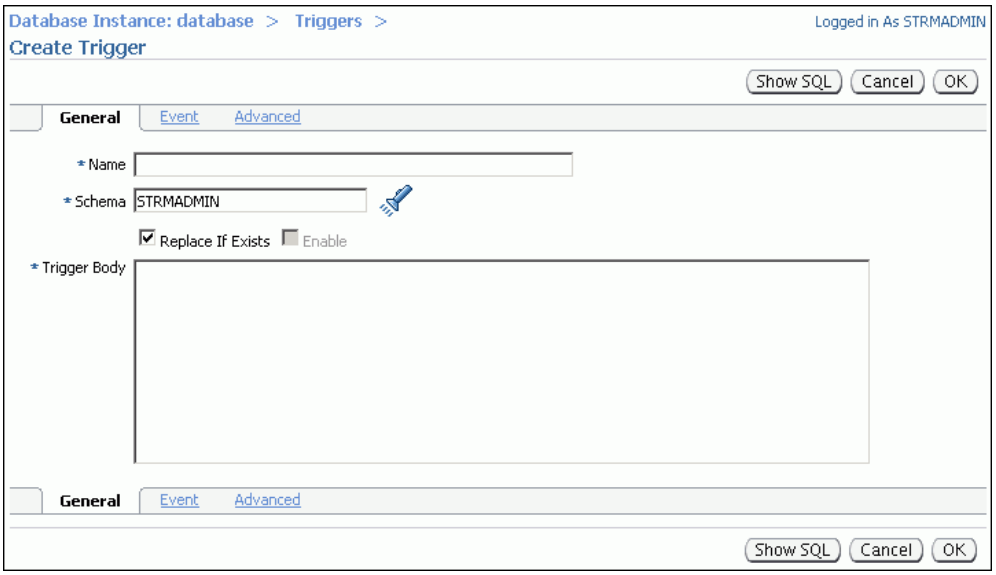
   b. Go to the Database Home page.

   c. Click **Schema** to open the Schema subpage.

   d. Click **Triggers** in the Programs section.

   e. On the Triggers page, click **Create**.

   The Create Trigger page appears, showing the General subpage.



f. Enter `enqueue_orders` in the **Name** field.

g. Ensure that `strmadmin` is entered in the **Schema** field.

h. Enter the following in the **Trigger Body** field:

```
DECLARE
  message  strmadmin.order_id_date;
BEGIN
  message := strmadmin.order_id_date(
               order_id   => :NEW.order_id,
               order_date => TO_CHAR(:NEW.order_date));
  DBMS_STREAMS_MESSAGING.ENQUEUE (
    queue_name => 'strmadmin.streams_queue',
    payload    => ANYDATA.CONVERTOBJECT(message));
END;
```

i. Click **Event** to open the Event subpage.

j. Ensure that **Table** is selected in the Trigger On list.

k. Enter `oe.orders` in the **Table (Schema.Table)** field.

l. Select **After** for **Fire Trigger**.

m. Select **Insert** for **Event**. In this example, only **Insert** should be selected because the messages track new orders, not changes to existing orders.

n. Click **Advanced**.

o. Select **Trigger for each row**.

p. Enter `OLD` in the **Old as** field in the Referencing section.

q. Enter `NEW` in the **New as** field in the Referencing section.

r. Click **OK** to create the trigger.

Note: You can also use the `CREATE TRIGGER` SQL statement to create a trigger.

---

3. Complete the steps in "Task 4: Configuring a Messaging Client to Dequeue Messages" to continue this extended example.

Task 4: Configuring a Messaging Client to Dequeue Messages

Configure a mechanism to dequeue the messages in your messaging system. Typically, an application dequeues and processes messages that were created by another application. For simplicity, this example creates a PL/SQL procedure called `dequeue_orders` to dequeue messages that include the order ID and order date of an order. You call the procedure to dequeue messages in this example, but an application can run such a procedure periodically.

To configure a messaging client to dequeue messages:

1. Grant `EXECUTE` privilege on the `SYS.DBMS_STREAMS_MESSAGING` package to the Oracle Streams administrator at the `ii2.example.com` database. See Step 1 in "Task 3: Configuring a Message Enqueuing Mechanism" on page 9-10 for an example that grants this privilege to the Oracle Streams administrator in the `ii1.example.com` database.
2. On a command line, open SQL*Plus and connect to the `ii2.example.com` database as the Oracle Streams administrator.

   See *Oracle Database 2 Day DBA* for more information about starting SQL*Plus.
3. Create a messaging client to enable the Oracle Streams administrator to dequeue messages from the `streams_queue` queue:

```
BEGIN
  DBMS_STREAMS_ADM.ADD_MESSAGE_RULE (
    message_type   => 'strmadmin.order_id_date',
    rule_condition => ':MSG.ORDER_ID > 0',
    streams_type   => 'DEQUEUE',
    streams_name   => 'strmadmin',
    queue_name     => 'strmadmin.streams_queue');
END;
/
```

   The user name of the Oracle Streams administrator must be specified for the `streams_name` parameter. In this example, the user name of the Oracle Streams administrator is `strmadmin`.

   A messaging client uses rules to determine which messages to dequeue. In this example, the rule for the messaging client specifies that all messages with an `order_id` greater than zero should be dequeued. So, with this rule, the messaging client will dequeue all new messages of type `strmadmin.order_id_date` that appear in the `strmadmin.streams_queue` queue.
4. In Oracle Enterprise Manager, log in to the `ii2.example.com` database as the Oracle Streams administrator.
5. Go to the Database Home page.
6. Click **Schema** to open the Schema subpage.
7. Click **Procedures** in the Programs section.
8. On the Procedures page, click **Create**.
9. On the Create Procedure page, enter `dequeue_orders` in the **Name** field.
10. Ensure that `strmadmin` is entered in the **Schema** field.

11. Delete the sample text in the **Source** field.
12. Enter the following in the **Source** field:

```
AS
  msg            ANYDATA;
  user_msg       strmadmin.order_id_date;
  num_var        PLS_INTEGER;
  more_messages  BOOLEAN := TRUE;
  navigation     VARCHAR2(30);
BEGIN
  navigation := 'FIRST MESSAGE';
  WHILE (more_messages) LOOP
    BEGIN
      DBMS_STREAMS_MESSAGING.DEQUEUE(
        queue_name   => 'strmadmin.streams_queue',
        streams_name => 'strmadmin',
        payload      => msg,
        navigation   => navigation,
        wait         => DBMS_STREAMS_MESSAGING.NO_WAIT);
      IF msg.GETTYPENAME() = 'STRMADMIN.ORDER_ID_DATE' THEN
        num_var := msg.GETOBJECT(user_msg);
        DBMS_OUTPUT.PUT_LINE('Order ID: ' || user_msg.order_id);
        DBMS_OUTPUT.PUT_LINE('Order Date: ' || user_msg.order_date);
      END IF;
      navigation := 'NEXT MESSAGE';
      COMMIT;
    EXCEPTION WHEN SYS.DBMS_STREAMS_MESSAGING.ENDOFCURTRANS THEN
                navigation := 'NEXT TRANSACTION';
              WHEN DBMS_STREAMS_MESSAGING.NOMOREMSGS THEN
                more_messages := FALSE;
                DBMS_OUTPUT.PUT_LINE('No more messages.');
              WHEN OTHERS THEN
                RAISE;
    END;
  END LOOP;
END;
```

13. Click **OK**.
14. Complete the steps in "Task 5: Enqueuing Messages" to continue this extended example.

Note: You can also use the `CREATE PROCEDURE` SQL statement to create a procedure.

Task 5: Enqueuing Messages

The example in "Task 3: Configuring a Message Enqueuing Mechanism" on page 9-10 created a trigger. The trigger enqueues a message with information about an order that was inserted into the `oe.orders` table into the `strmadmin.streams_queue` queue. So, you can enqueue messages into the `strmadmin.streams_queue` queue by inserting rows into the `oe.orders` table.

To insert rows into the oe.orders table:

1. On a command line, open SQL*Plus and connect to the `ii1.example.com` database as `oe` user.

   See *Oracle Database 2 Day DBA* for more information about starting SQL*Plus.

2. In SQL*Plus, insert rows into the `oe.orders` table:

```
INSERT INTO oe.orders VALUES(3000, SYSDATE, 'direct', 116, 0, 4000.00, 153,
NULL);

INSERT INTO oe.orders VALUES(3001, SYSDATE, 'direct', 117, 5, 5000.00, 163,
NULL);

INSERT INTO oe.orders VALUES(3002, SYSDATE, 'direct', 118, 7, 6000.00, 159,
NULL);
```

3. In SQL*Plus, commit your changes:

```
COMMIT;
```

4. Complete the steps in "Task 6: Dequeuing Messages" to continue this extended example.

Task 6: Dequeuing Messages

The example in "Task 5: Enqueuing Messages" on page 9-14 enqueues messages into the `strmadmin.streams_queue` queue in the `ii1.example.com` database. After messages are enqueued, the propagation configured in "Task 2: Configuring the Queues and Propagation Between Them" on page 9-8 sends the messages to the `strmadmin.streams_queue` queue in the `ii2.example.com` database.

This example dequeues the messages from the queue in the `ii2.example.com` database using the `dequeue_orders` procedure created in "Task 4: Configuring a Messaging Client to Dequeue Messages" on page 9-13.

To dequeue messages:

1. Optionally, check to see if the messages have arrived in the `strmadmin.streams_queue` queue at `ii2.example.com` before you attempt to dequeue them. It might take time for the propagation to send messages from the queue at `ii1.example.com` to the queue at `ii2.example.com`. See "Viewing the Messages in a Queue" on page 9-31 for instructions.

   Three messages should appear on the Messages page. Each message contains information about an order you inserted into the `oe.orders` table in "Task 5: Enqueuing Messages" on page 9-14.

2. On a command line, open SQL*Plus and connect to the `ii2.example.com` database as the Oracle Streams administrator.

   See *Oracle Database 2 Day DBA* for more information about starting SQL*Plus.

3. In SQL*Plus, run the `dequeue_orders` procedure:

```
SET SERVEROUTPUT ON
exec dequeue_orders;
```

   The output will be similar to the following:

```
Order ID: 3000
Order Date: 01-FEB-07 01.47.48.000000000 PM
Order ID: 3001
Order Date: 01-FEB-07 01.47.57.000000000 PM
Order ID: 3002
Order Date: 01-FEB-07 01.48.04.000000000 PM
```

```
No more messages.

PL/SQL procedure successfully completed.
```

Tutorial: Configuring Message Notifications

You can configure message notifications that alert applications or users when a message of interest is enqueued.

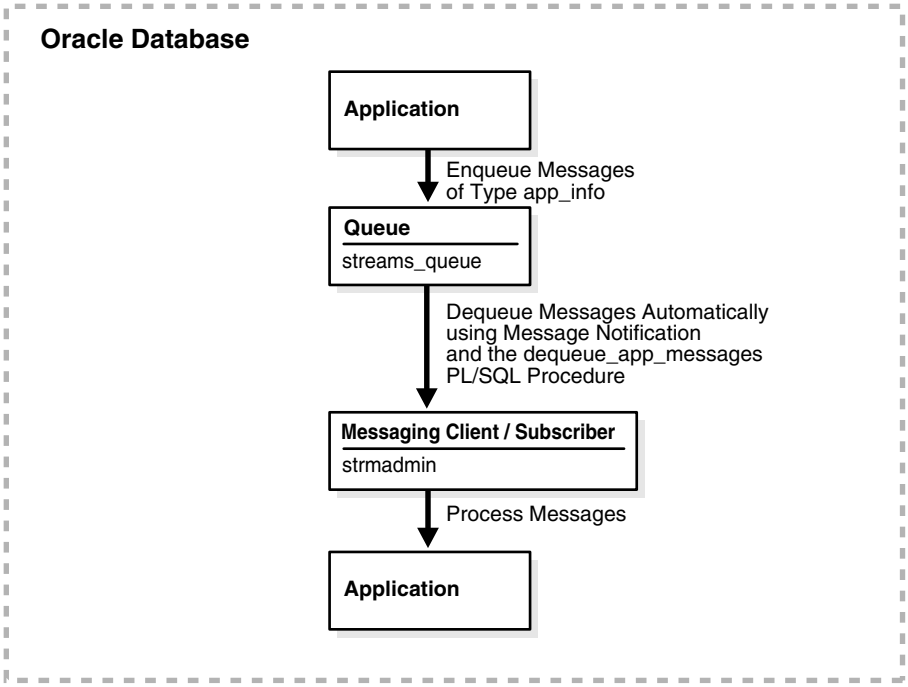
The example in this topic illustrates how you can use message notifications to enable two applications to communicate with each other. In this example, the applications communicate in the following way:

1. The first application determines various parameters that must be set in the second application and the values for these parameters.
2. The first application enqueues messages that contain the following attributes:
   - `parameter`: Specifies the parameter to set in the second application
   - `value`: Specifies the parameter value
3. Message notification alerts the second application that there are new messages in the queue.
4. The second application dequeues the messages and sets the parameters to the values in the messages.

For simplicity, this example does not create the two applications. Instead, it illustrates how to configure a message notification and how message notification can be used to dequeue messages of interest automatically when they are enqueued.

Figure 9–2 provides an overview of the messaging environment created in this example.

Figure 9–2 Sample Messaging Environment That Uses Message Notification



Note: You can also configure message notification in an environment that sends messages between databases. See "Tutorial: Sending Messages Between Oracle Databases" on page 9-6 for an example of an environment that sends messages between databases.

To configure a message notification:

- Task 1: Creating the Message Type
- Task 2: Configuring a Queue and a Messaging Client
- Task 3: Configuring a Mechanism for Dequeuing Messages
- Task 4: Configuring Message Notification
- Task 5: Enqueuing Messages and Checking for Message Notification

See Also:

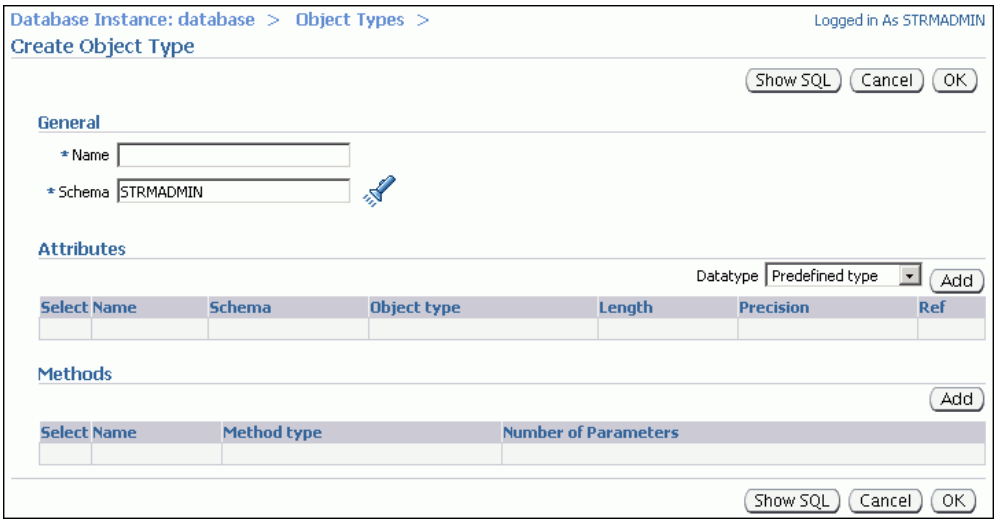
- "About Message Notifications" on page 9-3
- *Oracle Streams Advanced Queuing User's Guide* for detailed information about message notification features
- *Oracle Streams Concepts and Administration* for an example that configures e-mail message notifications

Task 1: Creating the Message Type

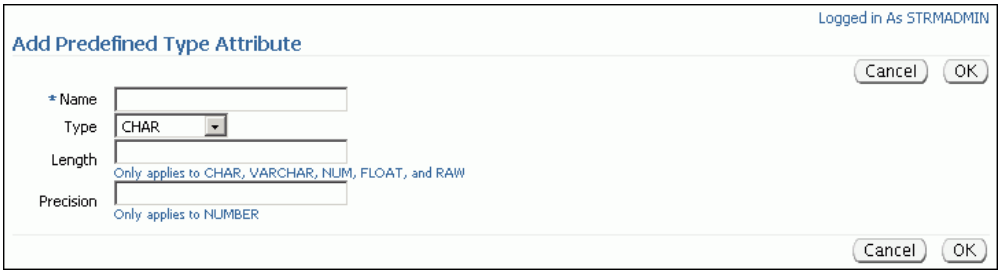
Create a user-defined type to define the information that you want to track for the applications. This example creates the `strmadmin.app_info` type for messages that will contain a parameter and a value.

To create the strmadmin.app_info type:

1. Prepare your environment for messaging if you have not already done so. See "Preparing for Messaging" on page 9-5.
2. In Oracle Enterprise Manager, log in to the database as the Oracle Streams administrator.
3. Go to the Database Home page.
4. Click **Schema** to open the Schema subpage.
5. Click **Object Types** in the User Defined Types section.
6. On the Object Types page, click **Create** to open the Create Object Type page.



7. Enter `app_info` in the **Name** field.
8. Ensure that `strmadmin` is selected in the **Schema** field.
9. Ensure that **Predefined Type** is selected for **Datatype** in the Attributes section.
10. Click **Add** in the Attributes section to open the Add Predefined Type Attributes page.



11. To add the first attribute to the `app_info` type:
    a. Enter `parameter` in the **Name** field.
    b. Select `VARCHAR2` for **Type**.
    c. Enter `20` in the **Length** field.
    d. Click **OK**.
12. On the Create Object Type page, ensure that **Predefined Type** is selected for **Datatype** in the Attributes section.
13. Click **Add** in the Attributes section to open the Add Predefined Type Attributes page.
14. To add the next attribute to the `app_info` type:
    a. Enter `value` in the **Name** field.
    b. Select `NUMBER` for **Type**.

c. Enter `12` in the **Length** field.

d. Click **OK**.

15. On the Create Object Type page, click **OK** to create the type.

16. Complete the steps in "Task 2: Configuring a Queue and a Messaging Client" to continue this extended example.

Note: You can also use the `CREATE TYPE` SQL statement to create a type.

See Also:

- "Tutorial: Configuring Message Notifications" on page 9-16
- "About Messaging" on page 9-1

Task 2: Configuring a Queue and a Messaging Client

Create a queue table and a queue to store the messages that the first application will generate for the second application. After the queue is created, message notification requires a consumer who can dequeue messages from the queue. In this example, a messaging client is the consumer who can dequeue messages from the `streams_queue` queue.

To configure a queue and a messaging client:

1. Create a queue named `streams_queue` in the schema of the Oracle Streams administrator. See "Creating an ANYDATA Queue" on page 2-7 for instructions.

2. On a command line, open SQL*Plus and connect to the database as the Oracle Streams administrator.

   See *Oracle Database 2 Day DBA* for more information about starting SQL*Plus.

3. Create a messaging client to enable the Oracle Streams administrator to dequeue messages from the `streams_queue` queue:

```
BEGIN
  DBMS_STREAMS_ADM.ADD_MESSAGE_RULE (
    message_type   => 'strmadmin.app_info',
    rule_condition => ':MSG.VALUE >= 0',
    streams_type   => 'DEQUEUE',
    streams_name   => 'strmadmin',
    queue_name     => 'strmadmin.streams_queue');
END;
/
```

   The user name for the Oracle Streams administrator must be specified for the `streams_name` parameter. In this example, the user name for the Oracle Streams administrator is `strmadmin`. The name of the new messaging client is also `strmadmin`.

   A messaging client uses rules to determine which messages to dequeue. In this example, the rule for the messaging client specifies that all messages with a `value` greater than zero should be dequeued. So, with this rule, the messaging client will dequeue all new messages of type `strmadmin.app_info` that appear in `strmadmin.streams_queue` because all parameter values are greater than or equal to zero.

4. Complete the steps in "Task 3: Configuring a Mechanism for Dequeuing Messages" to continue this extended example.

See Also:

- "Tutorial: Configuring Message Notifications" on page 9-16
- "About Messaging" on page 9-1

Task 3: Configuring a Mechanism for Dequeuing Messages

In this extended example, the second application must be able to dequeue messages from the `streams_queue` queue. For simplicity, this example creates a PL/SQL procedure called `dequeue_app_messages` to dequeue messages of type `strmadmin.app_info`. This procedure uses the messaging client created in "Task 2: Configuring a Queue and a Messaging Client" on page 9-19 to dequeue messages.

To configure a mechanism for dequeuing messages:

1. Grant the `EXECUTE` privilege on the `DBMS_AQ` package to the Oracle Streams administrator.

   This example configures a procedure that runs the `DEQUEUE` procedure in the `DBMS_AQ` package. The user who runs this procedure must have explicit `EXECUTE` privilege on the package that contains the procedure. The privilege cannot be granted through a role. Therefore, the Oracle Streams administrator must be granted explicit `EXECUTE` privilege on the package.

   a. Log in to Enterprise Manager as an administrative user who can grant privileges to the `strmadmin` user.

   b. Go to the Server subpage.

   c. Click **Users** in the Security section.

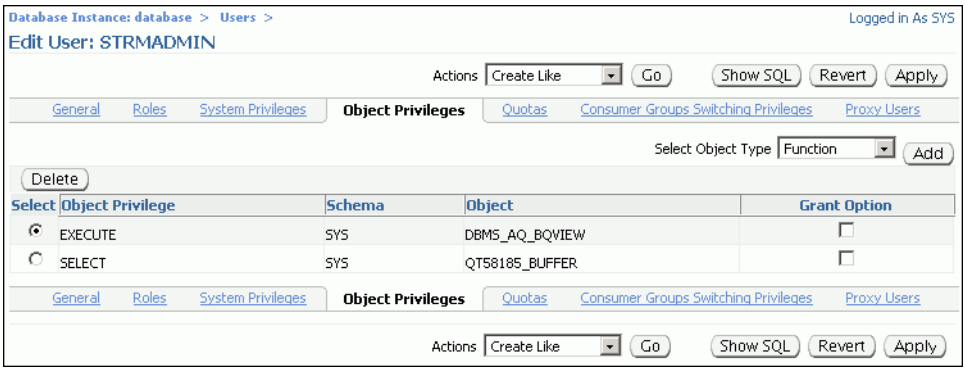
      The Users page appears.

   d. Select the `STRMADMIN` user.

   e. Click **Edit**.

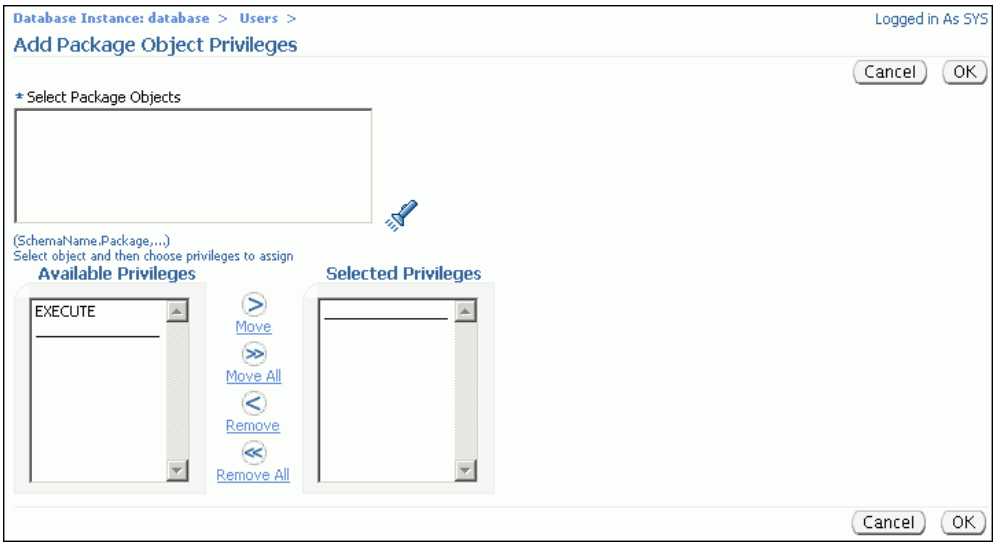
      The General subpage of the Edit User page appears.

   f. Click **Object Privileges** to open the Object Privileges subpage.



   g. Select **Package** in the Select Object Type list.

   h. Click **Add** to open the Add Package Object Privileges page.



i. Enter `SYS.DBMS_AQ` in the **Select Package Objects** field.

j. Move **EXECUTE** from the Available Privileges list to the Selected Privileges list.

k. Click **OK** to add the privilege.

l. Click **Apply** to grant the privilege.

Note: You can also use the `GRANT` SQL statement to grant privileges to a user.

---

2. Create a PL/SQL procedure that dequeues messages.

   a. In Oracle Enterprise Manager, log in to the database as the Oracle Streams administrator.

   b. Go to the Database Home page.

   c. Click **Schema** to open the Schema subpage.

   d. Click **Procedures** in the Programs section.

   e. On the Procedures page, click **Create**.

   f. On the Create Procedure page, enter `dequeue_app_messages` in the **Name** field.

   g. Ensure that `strmadmin` is entered in the **Schema** field.

   h. Delete the sample text in the **Source** field.

   i. Enter the following in the **Source** field:

```
(
  context  ANYDATA,
  reginfo  SYS.AQ$_REG_INFO,
  descr    SYS.AQ$_DESCRIPTOR)
AS
  dequeue_options     DBMS_AQ.DEQUEUE_OPTIONS_T;
  message_properties  DBMS_AQ.MESSAGE_PROPERTIES_T;
  message_handle      RAW(16);
  message             ANYDATA;
```

```
  app_message         strmadmin.app_info;
  rc                  PLS_INTEGER;
BEGIN
  -- Get the message identifier and consumer name from the descriptor
  dequeue_options.msgid := descr.msg_id;
  dequeue_options.consumer_name := descr.consumer_name;
  -- Dequeue the message
  DBMS_AQ.DEQUEUE(
    queue_name         => descr.queue_name,
    dequeue_options    => dequeue_options,
    message_properties => message_properties,
    payload            => message,
    msgid              => message_handle);
  rc := message.getobject(app_message);
  COMMIT;
END;
```

A message notification PL/SQL procedure must have the following signature:

```
PROCEDURE procedure_name(
  context  IN  ANYDATA,
  reginfo  IN  SYS.AQ$_REG_INFO,
  descr    IN  SYS.AQ$_DESCRIPTOR);
```

Here, *procedure_name* stands for the name of the procedure. The procedure is a `PLSQLCALLBACK` data structure that specifies the user-defined PL/SQL procedure to be invoked on message notification.

The procedure in this example is a simple notification procedure that dequeues a message of type `strmadmin.app_info` type using the message identifier and consumer name sent by the notification.

j. Click **OK** to create the procedure.

Note: You can also use the `CREATE PROCEDURE` SQL statement to create a procedure.

---

3. Complete the steps in "Task 4: Configuring Message Notification" to continue this extended example.

See Also:

- "Tutorial: Configuring Message Notifications" on page 9-16
- "About Messaging" on page 9-1

Task 4: Configuring Message Notification

In this extended example, message notification invokes the `strmadmin.dequeue_app_messages` procedure when a new message is enqueued into the `strmadmin.streams_queue` queue. This procedure dequeues the new message or messages. After dequeuing, an application can process the messages in any customized way. In this example, the second application uses the information in the messages to set application parameter values. For simplicity, this example does not configure the second application.

To configure message notification that invokes a procedure:

1. On a command line, open SQL*Plus and connect to the database as the Oracle Streams administrator.

   See *Oracle Database 2 Day DBA* for more information about starting SQL*Plus.

2. Set the message notification to invoke the `strmadmin.dequeue_app_messages` procedure when messages are enqueued and they can be dequeued by the `strmadmin` messaging client:

```
BEGIN
  DBMS_STREAMS_ADM.SET_MESSAGE_NOTIFICATION (
    streams_name         => 'strmadmin',
    notification_action  => 'strmadmin.dequeue_app_messages',
    notification_type    => 'PROCEDURE',
    include_notification => TRUE,
    queue_name           => 'strmadmin.streams_queue');
END;
/
```

3. Complete the steps in "Task 5: Enqueuing Messages and Checking for Message Notification" to continue this extended example.

See Also:

- "Tutorial: Configuring Message Notifications" on page 9-16
- "About Messaging" on page 9-1

Task 5: Enqueuing Messages and Checking for Message Notification

In this extended example, one application enqueues messages that are processed by a second application. For simplicity, this example enqueues messages manually. After the messages are enqueued, you can ensure that they have been dequeued automatically through message notification.

To enqueue messages and check for message notification:

1. On a command line, open SQL*Plus and connect to the database as the Oracle Streams administrator.

   See *Oracle Database 2 Day DBA* for more information about starting SQL*Plus.

2. Enqueue messages into the `strmadmin.streams_queue` queue:

```
DECLARE
  msg    strmadmin.app_info;
BEGIN
  msg := strmadmin.app_info(
           parameter => 'THRESHOLD',
           value     => 1000);
  DBMS_STREAMS_MESSAGING.ENQUEUE (
    queue_name   => 'strmadmin.streams_queue',
    payload      => ANYDATA.CONVERTOBJECT(msg));
  COMMIT;
END;
/
```

```
DECLARE
  msg    strmadmin.app_info;
BEGIN
  msg := strmadmin.app_info(
           parameter => 'MIN_WAIT',
           value     => 1);
  DBMS_STREAMS_MESSAGING.ENQUEUE (
    queue_name   => 'strmadmin.streams_queue',
    payload      => ANYDATA.CONVERTOBJECT(msg));
  COMMIT;
END;
/

DECLARE
  msg    strmadmin.app_info;
BEGIN
  msg := strmadmin.app_info(
           parameter => 'BUFFER_SIZE',
           value     => 30);
  DBMS_STREAMS_MESSAGING.ENQUEUE (
    queue_name   => 'strmadmin.streams_queue',
    payload      => ANYDATA.CONVERTOBJECT(msg));
  COMMIT;
END;
/
```

3. Optionally, view the messages in the `streams_queue` queue, and check to see if they have been dequeued automatically using message notification. See "Viewing the Messages in a Queue" on page 9-31 for instructions.

   When the message notification configured in this example is working properly, you should see one of the following on the Messages page:

   - One or more of the enqueued messages might appear with `Processed` in the **Consumers in Different States** field. A `Processed` value in this field means that the message has been dequeued by all consumers.
   - No messages appear in the queue. Eventually, processed messages are deleted from a queue automatically. If no messages appear, then the enqueued messages were dequeued by message notification and then deleted automatically.

     See Also:

     - "Tutorial: Configuring Message Notifications" on page 9-16
     - "About Messaging" on page 9-1

Modifying Queues

You can use Enterprise Manager to modify some of the properties of an existing queue, including the following properties:

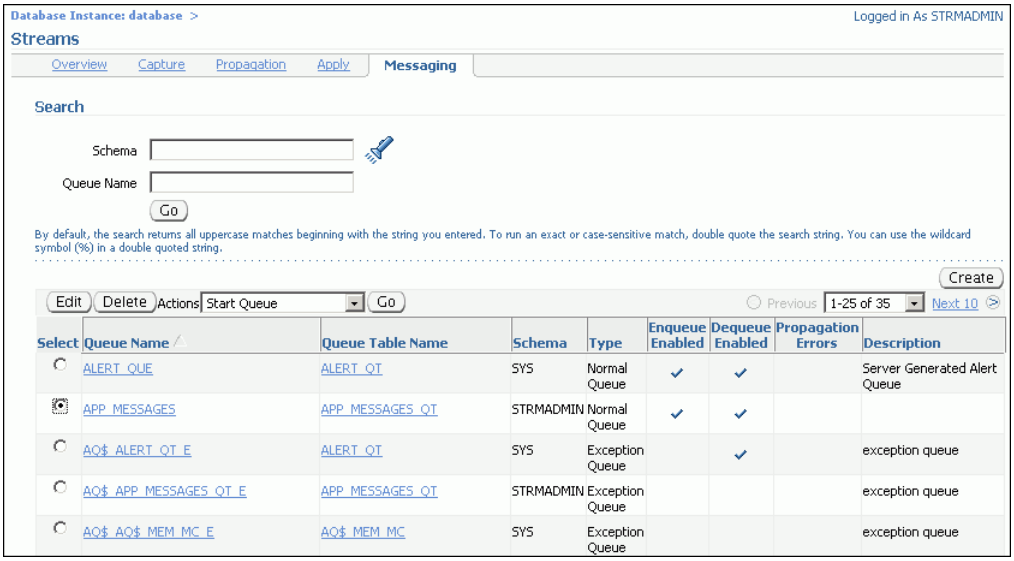
- **Maximum Retries:** The number of dequeuing attempts allowed on a message before the message is moved to the exception queue. If there is poor connectivity in your messaging environment, then a connection might be lost while an application is in the process of dequeuing messages. In such environment, you might increase the number of maximum retries allowed so that applications can try to dequeue messages a number of times before the messages are moved to the exception queue.

- **Retry Delay:** The amount of time after which a message is scheduled for processing again after an application rollback. If there is poor connectivity in your messaging environment, then you might want applications to wait for a set period of time after a failed attempt to dequeue messages. In such environments, you might increase the retry delay setting to specify the amount of time to wait.
- **Retention Time:** The amount of time that a message is retained in a queue after the message has been dequeued by all of its consumers. If you want to track messages that have been consumed, then you might set a retention time that enables you to record information about the messages that have been stored in the queue.

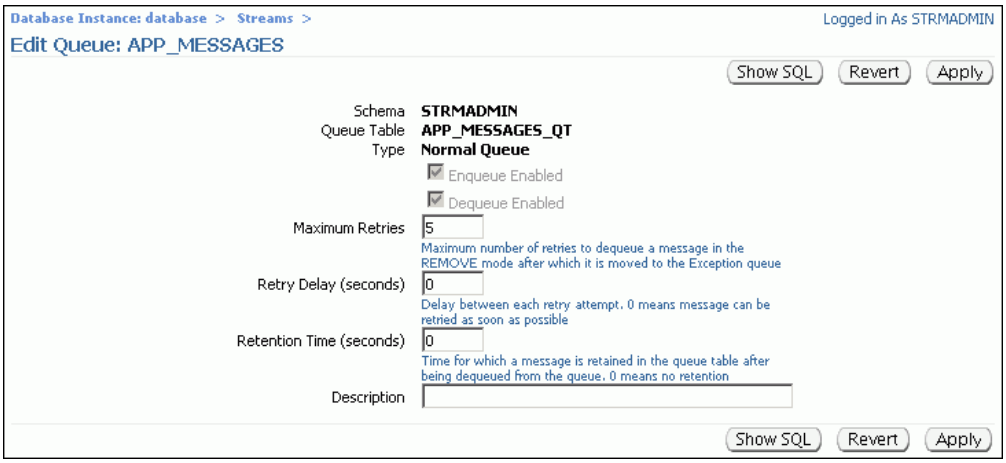
You can set these properties to values that work best with the applications that dequeue messages from the queue.

To modify a queue:

1. In Oracle Enterprise Manager, log in to the database as the Oracle Streams administrator.
2. Go to the Database Home page.
3. Click **Data Movement** to open the Data Movement subpage.
4. Click **Manage** in the Streams section.
5. On the Streams page, click **Messaging** to open the Messaging subpage.



6. Select the queue that you want to modify.
7. Click **Edit** to open the Edit Queue page.



8. Modify one or more queue properties. For information about this page, click **Help**.
9. Click **Apply** to save your changes.

Note: You can also use the `DBMS_AQADM.ALTER_QUEUE` procedure to modify a queue.

See Also:

- About Messaging on page 9-1
- "Tutorial: Sending Messages Between Oracle Databases" on page 9-6
- "Tutorial: Configuring Message Notifications" on page 9-16
- "Modifying Queue Tables" on page 9-26
- "Monitoring a Messaging Environment" on page 9-30

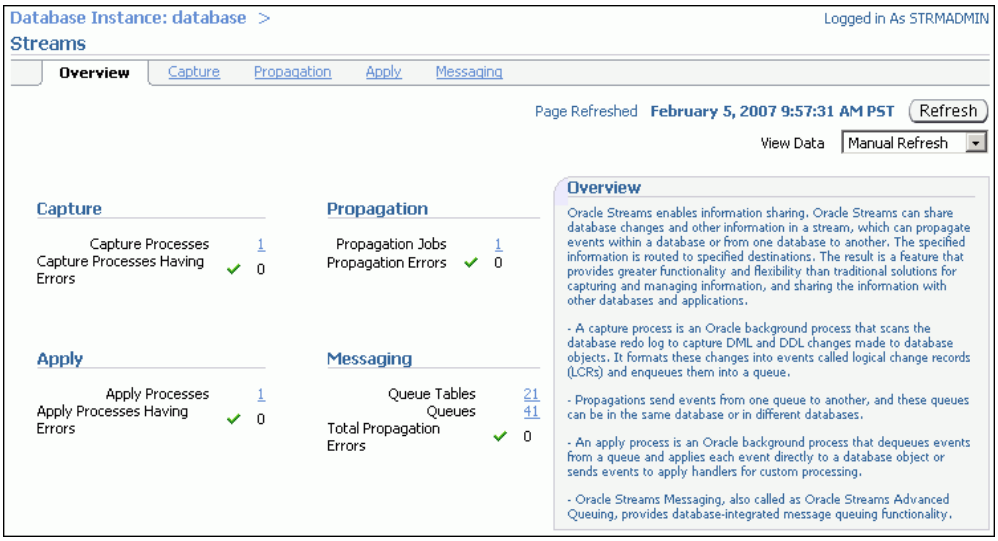
Modifying Queue Tables

You can use Enterprise Manager to modify some of the storage options of an existing queue table. Changing the storage options for a queue table can improve the performance of the queues that use the queue table.

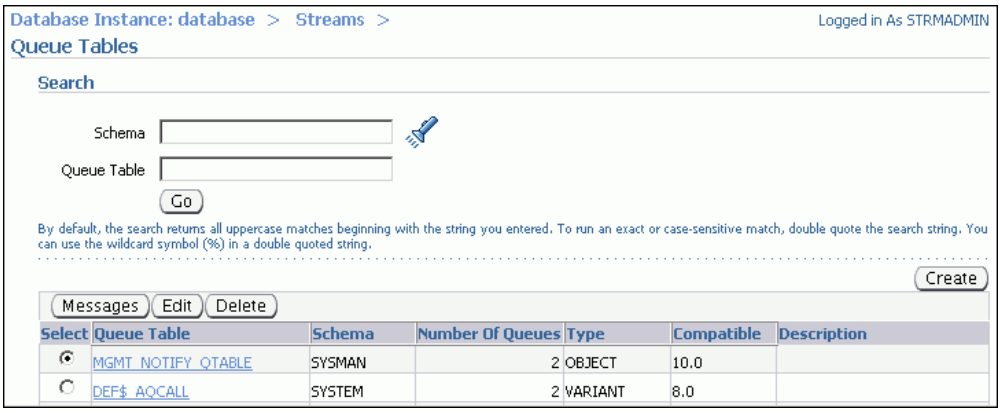
To modify a queue table:

1. In Oracle Enterprise Manager, log in to the database as the Oracle Streams administrator.
2. Go to the Database Home page.
3. Click **Data Movement** to open the Data Movement subpage.
4. Click **Manage**.

   The Streams page appears, showing the Overview subpage.

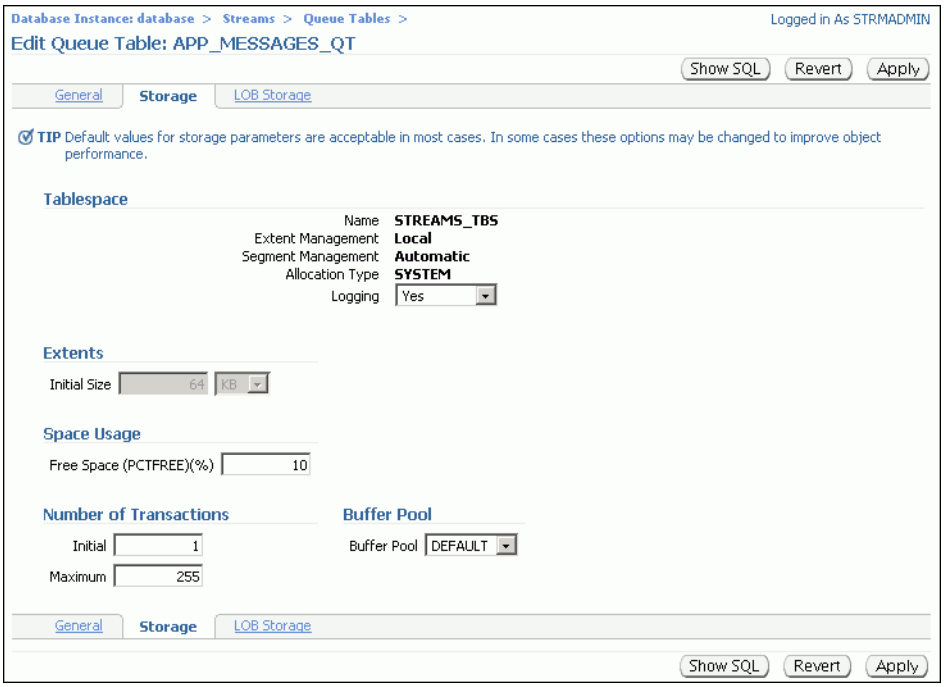


5. In the Messaging section, click the number associated with **Queue Tables** to open the Queue Tables page.



6. Select the queue table that you want to modify.
7. Click **Edit**.

   The Edit Queue Table page appears, showing the General subpage.
8. Click **Storage** to open the Storage subpage or **LOB Storage** to open the LOB Storage subpage.



9. Modify one or more queue table properties. For information about the subpage, click **Help**.
10. Click **Apply** to save your changes.

Note: You can also use the `DBMS_AQADM.ALTER_QUEUE_TABLE` procedure to modify a queue table.

See Also:

- "About Messaging" on page 9-1
- "Tutorial: Sending Messages Between Oracle Databases" on page 9-6
- "Tutorial: Configuring Message Notifications" on page 9-16
- "Modifying Queues" on page 9-24
- "Monitoring a Messaging Environment" on page 9-30

Modifying Propagations

You can use Enterprise Manager to modify the schedule of an existing propagation. A propagation schedule determines when and how often the propagation sends messages from one queue to another and how long each propagation lasts. You can modify the following schedule options:

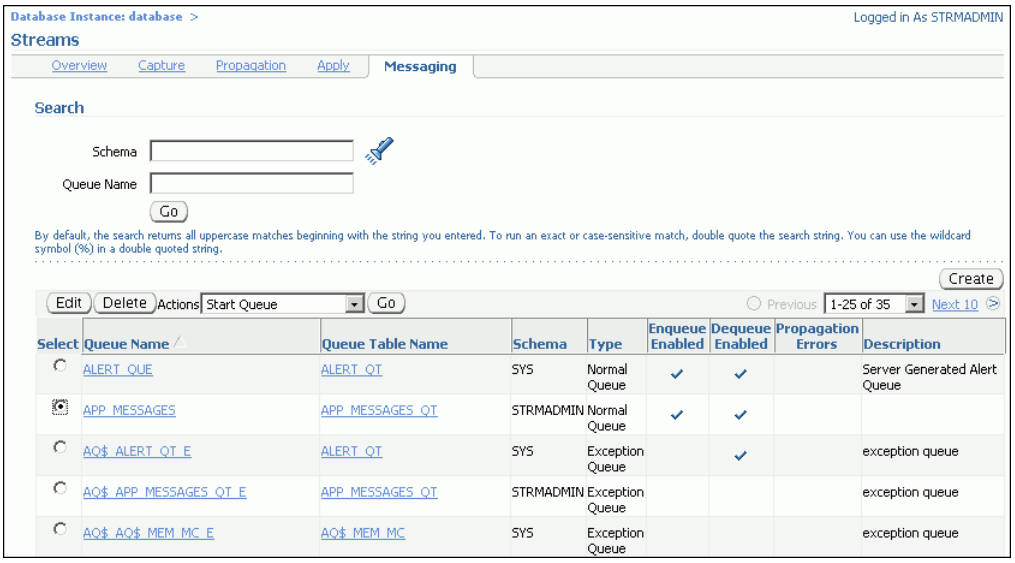
- **Latency:** The amount of time to wait before new messages in a completely propagated queue are propagated
- **Duration of the Propagation:** The amount of time that each individual propagation lasts
- **Next Time:** The amount of time in between individual propagations

Typically, you modify a propagation schedule to improve the performance of a propagation in your messaging environment. The default settings for these options are usually the best settings. However, you can try different settings for these options to achieve the best performance.

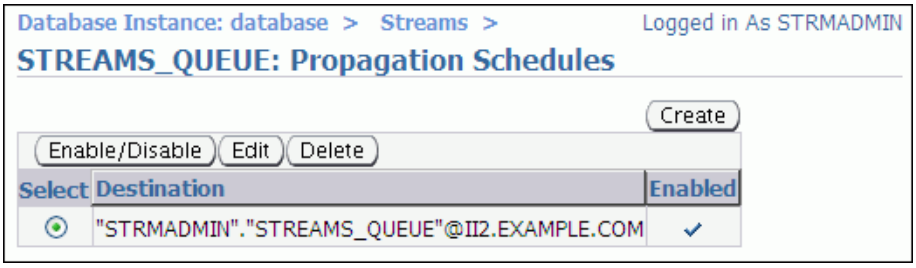
To modify a propagation:

1. In Oracle Enterprise Manager, log in to the database as the Oracle Streams administrator.
2. Go to the Database Home page.
3. Click **Data Movement** to open the Data Movement subpage.
4. Click **Manage** in the Streams section.

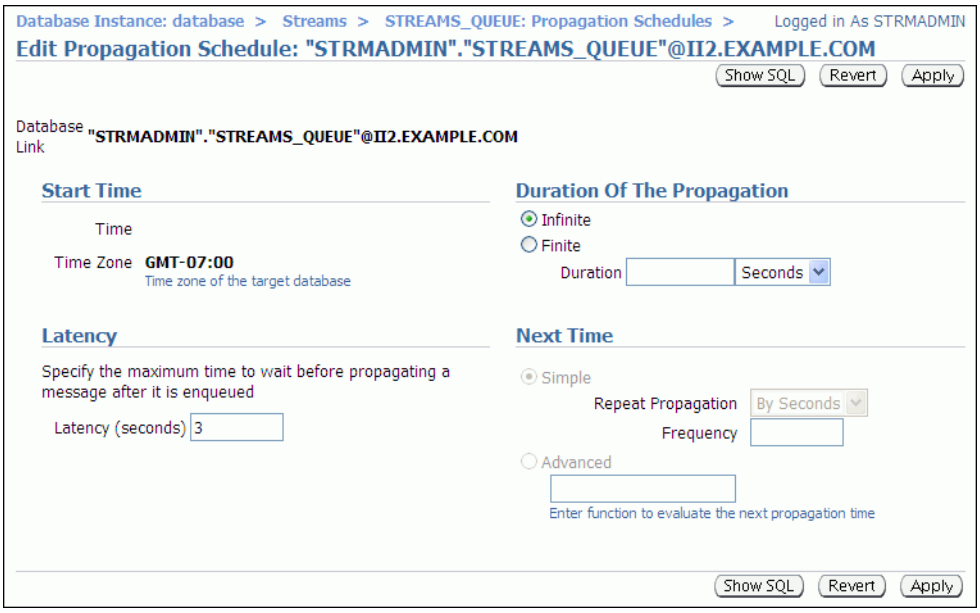
   The Streams page appears, showing the Overview subpage.
5. Click **Messaging** to open the Messaging subpage.



6. Select the source queue for the propagation that you want to modify. The source queue is the queue from which the propagation sends messages to a destination queue.
7. Select **Propagation Schedules** in the Actions list.
8. Click **Go** to open the Propagation Schedules page.



9. Select the propagation schedule that you want to modify.
10. Click **Edit** to open the Edit Propagation Schedule page.



11. Modify one or more schedule options. For information about the current page, click **Help**.
12. Click **Apply** to save your changes.

Note: You can also use the `DBMS_AQADM.ALTER_PROPAGATION_SCHEDULE` procedure to modify a propagation schedule.

See Also:

- "About Propagations" on page 9-4
- "Tutorial: Sending Messages Between Oracle Databases" on page 9-6

Monitoring a Messaging Environment

This section describes using Enterprise Manager to display information about a messaging environment. This section contains instructions for viewing the messages in a queue, queue statistics, and queue consumers.

The following topics describe monitoring a messaging environment:

- Viewing the Messages in a Queue
- Viewing Persistent Queue Statistics
- Viewing the Consumers Who Can Dequeue Messages

See Also:

- "About Messaging" on page 9-1
- "Preparing for Messaging" on page 9-5
- "Tutorial: Sending Messages Between Oracle Databases" on page 9-6
- "Tutorial: Configuring Message Notifications" on page 9-16
- "Troubleshooting a Messaging Environment" on page 9-34

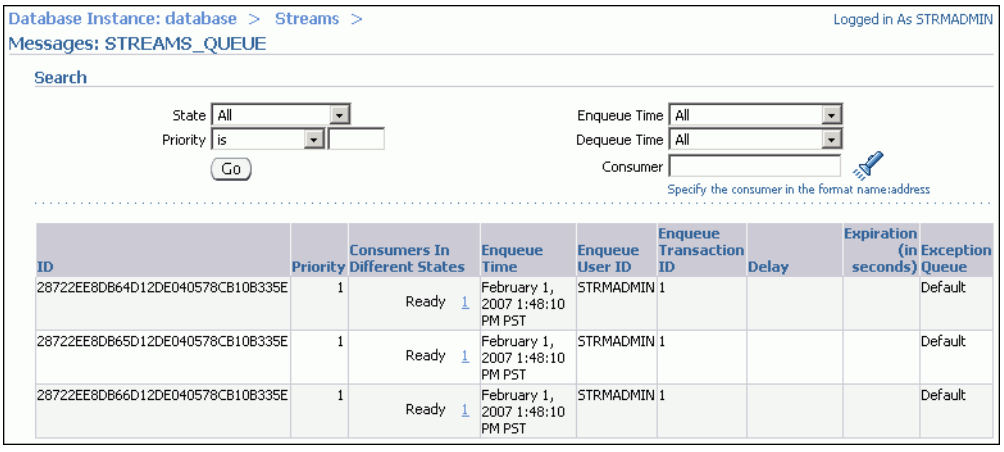
Viewing the Messages in a Queue

You can view the messages in a queue to ensure that an application is enqueuing messages correctly or to see which messages are currently ready to be dequeued or propagated to another queue.

To view the messages currently stored in a queue:

1. In Oracle Enterprise Manager, log in to the database as the Oracle Streams administrator.
2. Go to the Database Home page.
3. Click **Data Movement** to open the Data Movement subpage.
4. Click **Manage** in the Streams section.

   The Streams page appears, showing the Overview subpage.
5. Click **Messaging** to open the Messaging subpage.
6. Select the queue that contains the messages.
7. Select **Messages** in the Actions list.
8. Click **Go**.
9. On the Messages page, click **Go**. The messages are listed.



10. Click the number link in the **Consumers In Different States** field to view the consumers of a message. A consumer might be a propagation that sends the message to a different queue or an application that dequeues the message.

Click **Help** on a page for more information about the page.

See Also:

- "Modifying Queues" on page 9-24

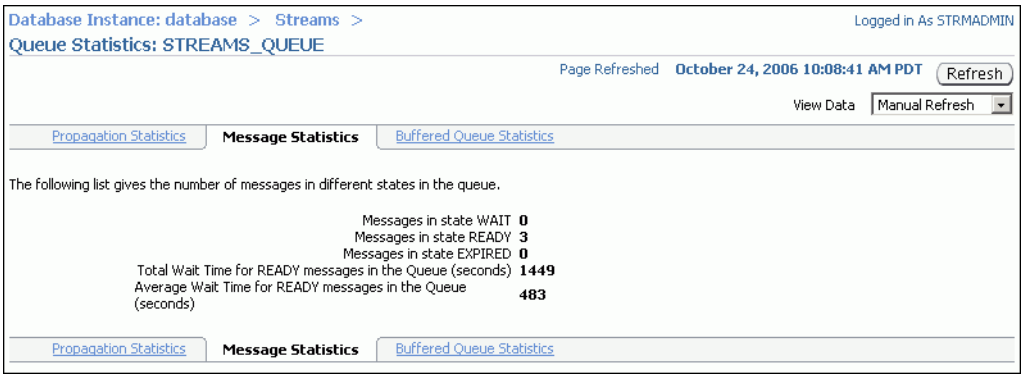
Viewing Persistent Queue Statistics

If you are using persistent messaging mode, then messages are stored in queue table on hard disk. This topic describes viewing statistics for messages that were enqueued in persistent mode into a persistent queue.

To view persistent queue statistics:

1. In Oracle Enterprise Manager, log in to the database as the Oracle Streams administrator.
2. Go to the Database Home page.
3. Click **Data Movement** to open the Data Movement subpage.
4. Click **Manage** in the Streams section.

   The Streams page appears, showing the Overview subpage.
5. Click **Messaging** to open the Messaging subpage.
6. Select the queue that contains the messages.
7. Select **Queue Statistics** in the Actions list.
8. Click **Go**.
9. On the Queue Statistics page, click **Message Statistics** to view the persistent queue statistics for the queue.



The message statistics show the number of messages in each state in the persistent queue. They also show the total wait time and the average wait time for messages to be dequeued.

Note:

- Messages can also be enqueued in buffered mode. In this case, the messages are stored in memory in a buffered queue. See "Viewing Buffered Queue Statistics" on page 5-23 for information about viewing statistics for these messages.
- You can also query the `V$AQ` dynamic performance view for queue statistics.

See Also:

- "Modifying Queues" on page 9-24

Viewing the Consumers Who Can Dequeue Messages

Consumers are configured to dequeue messages from a specific queue. In a messaging environment, consumers are represented by subscribers to a queue, and more than one consumer might use a single subscriber.

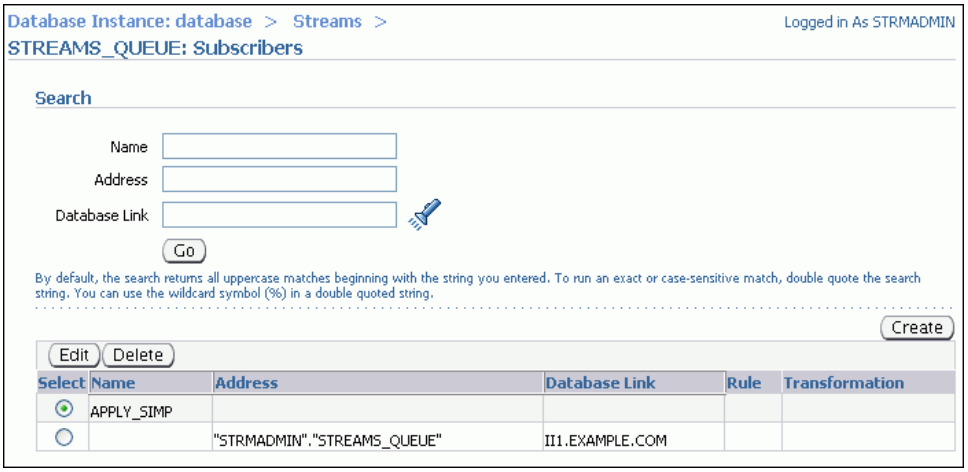
Consumers can be:

- Applications or users that dequeue and process messages
- Propagations that send messages from one queue to another
- Apply processes that can dequeue messages for custom processes in a messaging environment or dequeue changes to database objects in a replication environment

To view the consumers who can dequeue messages from a specific queue:

1. In Oracle Enterprise Manager, log in to the database as the Oracle Streams administrator.
2. Go to the Database Home page.
3. Click **Data Movement** to open the Data Movement subpage.
4. Click **Manage** in the Streams section.

   The Streams page appears, showing the Overview subpage.
5. Click **Messaging** to open the Messaging subpage.
6. Select the queue.
7. Select **Subscribers** in the Actions list.
8. Click **Go** to open the Subscribers page.



The Subscribers page includes the following information about each subscriber to the queue:

- The **Name** field contains the name of the subscriber.
- The **Address** field is typically populated if the subscriber is a propagation that sends messages to another queue. The **Address** field shows the name of the queue that receives the messages.
- The **Database Link** field shows the database link to a remote database if the subscriber is at the remote database.
- The **Rule** field shows the rule used by the subscriber. Rules can determine which messages are dequeued by the subscriber. Some subscribers do not use rules.
- The **Transformation** field shows the transformation used by the subscriber. Transformations modify messages while they are being dequeued. Some subscribers do not use transformations.

Click **Help** for more information about this page.

Note: You can also query the `ALL_QUEUE_SUBSCRIBERS` data dictionary to view the consumers who can dequeue messages.

See Also:

- "About Messaging" on page 9-1
- "About Change Apply" on page 4-6

Troubleshooting a Messaging Environment

This section describes the most common problems in a messaging environment. It also describes how to correct these problems.

The following topics describe troubleshooting a messaging environment:

- Correcting an ORA-01031 Error While Enqueuing or Dequeuing Messages
- Correcting an ORA-24033 Error While Enqueuing Messages
- Correcting an ORA-02019 Error for a Propagation
- Understanding Why Dequeued Messages Remain in a Queue

See Also:

- *Oracle Streams Advanced Queuing User's Guide*
- "About Messaging" on page 9-1

Correcting an ORA-01031 Error While Enqueuing or Dequeuing Messages

If a user who does not have the required privileges attempts to enqueue or dequeue messages, then Oracle Database returns the following error:

```
ORA-01031: insufficient privileges
```

The enqueue or dequeue operation fails when Oracle Database returns this message. To correct the problem, grant `ENQUEUE` or `DEQUEUE` privileges on the queue to the user.

To grant ENQUEUE or DEQUEUE privileges to a user:

1. On a command line, open SQL*Plus and connect to the database as an administrative user, such as the Oracle Streams administrator or `SYSTEM`.

   See *Oracle Database 2 Day DBA* for more information about starting SQL*Plus.

2. Run the `GRANT_QUEUE_PRIVILEGE` procedure in the `DBMS_AQADM` package to grant the required queue privileges to the user.

   For example, to grant the `ENQUEUE` privilege to the `hr` user on the `strmadmin.streams_queue` queue, run the following procedure:

```
BEGIN
   DBMS_AQADM.GRANT_QUEUE_PRIVILEGE (
      privilege  => 'ENQUEUE',
      queue_name => 'strmadmin.streams_queue',
      grantee    => 'hr');
END;
/
```

   To grant the `DEQUEUE` privilege to the `hr` user on the `strmadmin.streams_queue` queue, run the following procedure:

```
BEGIN
   DBMS_AQADM.GRANT_QUEUE_PRIVILEGE (
      privilege  => 'DEQUEUE',
      queue_name => 'strmadmin.streams_queue',
      grantee    => 'hr');
END;
/
```

See Also:

- *Oracle Streams Advanced Queuing User's Guide*
- "About Messaging" on page 9-1

Correcting an ORA-24033 Error While Enqueuing Messages

If there are no consumers for a message when a user or application tries to enqueue it, then Oracle Database returns the following error:

```
ORA-24033: no recipients for message
```

The message is not enqueued when Oracle Database returns this message.

To correct the problem:

1. Configure at least one consumer for the messages.
2. Enqueue the messages.

Consumers can be:

- Applications or users that dequeue and process messages
- Propagations that send messages from one queue to another
- Apply processes that can dequeue messages for custom processes in a messaging environment or dequeue changes to database objects in a replication environment
- Messaging clients that dequeue messages and pass them to an application

See Also:

- *Oracle Streams Advanced Queuing User's Guide*
- "About Messaging" on page 9-1

Correcting an ORA-02019 Error for a Propagation

A propagation uses the privileges of the database user who owns its source queue when it sends messages. If the source queue owner cannot access the database link used by the propagation, then Oracle Database returns the following error:

```
ORA-02019: connection description for remote database not found
```

The messages are not propagated when Oracle Database returns this message. The easiest way to correct this problem is to delete and re-create the database link so that the source queue owner can access it.

To delete and re-create a database link:

1. In Oracle Enterprise Manager, log in to the database as an administrative user, such as the Oracle Streams administrator or `SYSTEM`.
2. Go to the Database Home page.
3. Click **Schema** to open the Schema subpage.
4. Click **Database Links** in the Database Objects section.

   The Database Links page appears.
5. Use the search tool to list the database link you want to delete and re-create.
6. Select the database link.
7. Click **Delete**.
8. Click **Yes** on the confirmation page to delete the database link.
9. Follow the instructions in "Tutorial: Creating a Database Link" on page 2-8 to re-create the database link.

To ensure that the source queue owner for the propagation has access to the database link, specify this user as the fixed user in the Connect As section when you create the database link.

See Also:

- "About Propagations" on page 9-4
- *Oracle Database Administrator's Guide* for more information about database links

Understanding Why Dequeued Messages Remain in a Queue

A message can remain in a queue after it has been dequeued for the following reasons:

- One or more consumers of the message have not yet dequeued it. A message is not removed from a queue until all of its consumers have dequeued it.
- The message has not been cleaned up by a background process. After all of the consumers of a message have dequeued it, the message remains in the queue until a background process removes it automatically.

If you see messages in a queue that were dequeued by a consumer, then you can use Enterprise Manager to check on the consumers for a message and to determine whether all of the consumers for a message have dequeued it.

To check on messages in a queue:

1. Follow the instructions in "Viewing the Consumers Who Can Dequeue Messages" on page 9-33 to view the consumers who can dequeue the messages. If there are multiple consumers for a message, then some consumers might not have dequeued the message yet.
2. Follow the instructions in "Viewing the Messages in a Queue" on page 9-31. If the **Consumers in Different States** field shows `PROCESSED` for a message, then all of the consumers of the message have dequeued it. In this case, the message remains in the queue until the background process removes it automatically.

See Also:

- *Oracle Streams Advanced Queuing User's Guide*
- "About Messaging" on page 9-1

10

Comparing and Converging Data

This chapter describes how to compare data in shared database objects at two different databases. It also describes how to converge divergent data in shared database objects.

This chapter contains the following sections:

See Also:

- Chapter 7, "Replicating Data Using Materialized Views"
- Chapter 4, "Replicating Data Using Oracle Streams"
- *Oracle Database PL/SQL Packages and Types Reference*
- *Oracle Streams Replication Administrator's Guide*

About Comparing and Converging Data in Different Databases

You can share database objects at two or more databases. When copies of the same database object exist at multiple databases, the database object is a **shared database object**. Shared database objects might be maintained by data replication. For example, materialized views or Oracle Streams components might replicate the database objects and maintain them at multiple databases. A custom application might also maintain shared database objects. Typically, replication environments share database objects that contain data, such as tables, as well as other types of databases objects, such as indexes.

When a change is made to a shared database object at one database, the change is transferred to and made at each of the other databases that share the database object. In this way, the replication environment keeps the shared database object synchronized at each database.

Sometimes, shared database objects that contain data can become inconsistent at different databases. That is, the data might diverge in the different instances of the shared database object. For example, if the database object is a table, then one instance of the table might have more rows than another instance of the table, or two instances of the table might have different data in the same rows.

When shared database objects diverge in an Oracle Streams replication environment, it is usually for one of the following reasons:

- Data changes are not being captured at one or more of the databases.
- Data changes are being captured, but they are not being transferred from one database to another.
- Data changes are being captured and transferred from one database to another, but they are not being made to shared database objects at the other databases.

Common causes of data divergence are network problems, incorrect configurations, or user errors. When shared database objects diverge in a replication environment that uses materialized views, it might be because there is a problem with the materialized view refresh.

The `DBMS_COMPARISON` package enables you to compare database objects at different databases and identify differences. This package also enables you to converge the database objects so that they are consistent at different databases. The `DBMS_COMPARISON` package is an Oracle-supplied PL/SQL package that is always installed with Oracle Database.

The `DBMS_COMPARISON` package can compare and converge the following types of database objects:

- Tables
- Single-table views
- Materialized views
- Synonyms for tables, single-table views, and materialized views

Database objects of different types can be compared and converged at different databases. For example, a table at one database and a materialized view at another database can be compared and converged.

In the examples in this guide, the shared database object has the same name at the two databases, and the entire database object is compared and converged. However, the `DBMS_COMPARISON` package provides flexibility for differences in the shared database object at different databases. The database objects being compared do not need to have the same name. In addition, column names can also be different in the database objects, as long as the corresponding columns are the same data type. You can compare and converge the entire shared database object or subsets of columns and rows.

To create a comparison, use the `CREATE_COMPARISON` procedure in the `DBMS_COMPARISON` package. This procedure identifies one or more index columns in the shared database object. The `DBMS_COMPARISON` package must be able to identify at least one column that it can use as an index column. If the specified database object does not have a column that can be used as an index column, then the `CREATE_COMPARISON` procedure cannot create a comparison for the database object.

Note: If your environment has shared database objects that have diverged, then you should investigate the cause of the problem and correct it. Although the `DBMS_COMPARISON` package can compare and converge shared database objects, it is better if these database objects do not diverge.

See Also:

- *Oracle Database PL/SQL Packages and Types Reference* for detailed information about the `DBMS_COMPARISON` package
- *Oracle Streams Replication Administrator's Guide* for information about using the advanced features of the `DBMS_COMPARISON` package

Tutorial: Preparing to Compare and Converge Data

Suppose you share the `hr.departments` table in two databases. You want to compare this table at these databases to see if their data is consistent. If the tables have diverged at the two databases, then you want to converge them to make them consistent.

Meet the following prerequisites to complete this tutorial:

- Configure network connectivity so that the two databases can communicate with each other. See *Oracle Database 2 Day DBA* for information about configuring network connectivity between databases.
- Ensure that the `hr` sample schema is installed on both databases. The `hr` sample schema is installed by default with Oracle Database.

In this example, the global names of the databases are `ii1.example.com` and `ii2.example.com`, but you can substitute any two databases in your environment that meet the prerequisites.

To prepare for comparison and convergence of the hr.departments table at the ii1.example.com and ii2.example.com databases:

1. For the purposes of this example, make the `hr.departments` table diverge at the two databases:
   a. On a command line, open SQL*Plus and connect to the `ii2.example.com` database as `hr` user.

      See *Oracle Database 2 Day DBA* for more information about starting SQL*Plus.

   b. Delete the department in the `hr.departments` table with the `department_id` equal to `270`:

      ```
      DELETE FROM hr.departments WHERE department_id=270;
      COMMIT;
      ```

   c. Modify the data in a row in the `hr.departments` table:

      ```
      UPDATE hr.departments SET manager_id=114 WHERE department_id=10;
      COMMIT;
      ```

   d. Insert a row into the `hr.departments` table:

      ```
      INSERT INTO hr.departments VALUES(280, 'Bean Counters', 108, 2700);
      COMMIT;
      ```

   e. Exit SQL*Plus:

      ```
      EXIT;
      ```

> **Note:** Usually, Step 1 is not required. It is included in this example to ensure that the `hr.departments` table diverges at the two databases.

2. Create a database link from the `ii1.example.com` database to the `ii2.example.com` database.

   The database link should connect from an administrative user in `ii1.example.com` to an administrative user schema in `ii2.example.com`. The administrative user at both databases should have the necessary privileges to access and modify the `hr.departments` table and the necessary privileges to run subprograms in the `DBMS_COMPARISON` package. If you are not sure which user has these privileges, then use `SYSTEM` user. Also, both the name and the service name of the database link must be `ii2.example.com`. See "Tutorial: Creating a Database Link" on page 2-8 for instructions.

Tutorial: Comparing Data in Two Different Databases

This example continues the scenario described in "Tutorial: Preparing to Compare and Converge Data" on page 10-3. Complete the steps in that topic before continuing.

You can use the `CREATE_COMPARISON` procedure in the `DBMS_COMPARISON` package to define a comparison of a shared database object at two different databases. Once the comparison is defined, you can use the `COMPARE` function in this package to compare the database object specified in the comparison at the current point in time. You can run the `COMPARE` function multiple times for a specific comparison. Each time you run the function, it results one or more scans of the database objects, and each scan has its own scan ID.

To compare the entire hr.departments table at the ii1.example.com and ii2.example.com databases:

1. On a command line, open SQL*Plus and connect to the `ii1.example.com` database as the administrative user who owns the database link created in "Tutorial: Preparing to Compare and Converge Data" on page 10-3. For example, if `SYSTEM` user owns the database link, then connect as `SYSTEM` user:

   ```
   sqlplus system@ii1.example.com
   Enter password: password
   ```

   See *Oracle Database 2 Day DBA* for more information about starting SQL*Plus.

2. Run the `CREATE_COMPARISON` procedure to create the comparison for the `hr.departments` table:

   ```
   BEGIN
     DBMS_COMPARISON.CREATE_COMPARISON(
       comparison_name => 'compare_departments',
       schema_name     => 'hr',
       object_name     => 'departments',
       dblink_name     => 'ii2.example.com');
   END;
   /
   ```

   Note that the name of the new comparison is `compare_departments`. This comparison is owned by the user who runs the `CREATE_COMPARISON` procedure.

3. Run the `COMPARE` function to compare the `hr.departments` table at the two databases:

```
SET SERVEROUTPUT ON
DECLARE
  consistent   BOOLEAN;
  scan_info    DBMS_COMPARISON.COMPARISON_TYPE;
BEGIN
  consistent := DBMS_COMPARISON.COMPARE(
                  comparison_name => 'compare_departments',
                  scan_info       => scan_info,
                  perform_row_dif => TRUE);
  DBMS_OUTPUT.PUT_LINE('Scan ID: '||scan_info.scan_id);
  IF consistent=TRUE THEN
    DBMS_OUTPUT.PUT_LINE('No differences were found.');
  ELSE
    DBMS_OUTPUT.PUT_LINE('Differences were found.');
  END IF;
END;
/

Scan ID: 1
Differences were found.

PL/SQL procedure successfully completed.
```

Specify the name of the comparison created in Step 2 for the `comparison_name` parameter.

The function prints the scan ID for the comparison. The scan ID is important when you are querying data dictionary views for information about the comparison and when you are converging the database objects.

The function also prints whether or not differences were found in the table at the two databases:

- If the function prints `'No differences were found'`, then the table is consistent at the two databases.
- If the function prints `'Differences were found'`, then the table has diverged at the two databases.

4. Make a note of the scan ID returned by the function in the previous step. In this example, assume the scan ID is `1`.
5. If differences were found in Step 3, then run the following query to show the number of differences found:

```
COLUMN OWNER HEADING 'Comparison Owner' FORMAT A16
COLUMN COMPARISON_NAME HEADING 'Comparison Name' FORMAT A20
COLUMN SCHEMA_NAME HEADING 'Schema Name' FORMAT A11
COLUMN OBJECT_NAME HEADING 'Object Name' FORMAT A11
COLUMN CURRENT_DIF_COUNT HEADING 'Differences' FORMAT 9999999

SELECT c.OWNER,
       c.COMPARISON_NAME,
       c.SCHEMA_NAME,
       c.OBJECT_NAME,
       s.CURRENT_DIF_COUNT
  FROM DBA_COMPARISON c, DBA_COMPARISON_SCAN_SUMMARY s
  WHERE c.COMPARISON_NAME = s.COMPARISON_NAME AND
        c.OWNER           = s.OWNER AND
        s.SCAN_ID         = 1;
```

Specify the scan ID you recorded in Step 4 in the `WHERE` clause of the query.

The output will be similar to the following:

```
Comparison Owner Comparison Name      Schema Name Object Name Differences
---------------- -------------------- ----------- ----------- -----------
SYSTEM           COMPARE_DEPARTMENTS  HR          DEPARTMENTS           3
```

6. To see which rows were different in the database object being compared, run the following query:

```
COLUMN COLUMN_NAME HEADING 'Index Column' FORMAT A15
COLUMN INDEX_VALUE HEADING 'Index Value' FORMAT A15
COLUMN LOCAL_ROWID HEADING 'Local Row Exists?' FORMAT A20
COLUMN REMOTE_ROWID HEADING 'Remote Row Exists?' FORMAT A20

SELECT c.COLUMN_NAME,
       r.INDEX_VALUE,
       DECODE(r.LOCAL_ROWID,
                NULL, 'No',
                      'Yes') LOCAL_ROWID,
       DECODE(r.REMOTE_ROWID,
                NULL, 'No',
                      'Yes') REMOTE_ROWID
  FROM DBA_COMPARISON_COLUMNS c,
       DBA_COMPARISON_ROW_DIF r,
       DBA_COMPARISON_SCAN s
  WHERE c.COMPARISON_NAME = 'COMPARE_DEPARTMENTS' AND
        r.SCAN_ID         = s.SCAN_ID AND
        s.PARENT_SCAN_ID  = 1 AND
        r.STATUS          = 'DIF' AND
        c.INDEX_COLUMN    = 'Y' AND
        c.COMPARISON_NAME = r.COMPARISON_NAME AND
        c.OWNER           = r.OWNER
  ORDER BY r.INDEX_VALUE;
```

In the `WHERE` clause, specify the name of the comparison and the scan ID for the comparison. In this example, the name of the comparison is `compare_departments` and the scan ID is `1`.

The output will be similar to the following:

```
Index Column    Index Value     Local Row Exists?    Remote Row Exists?
--------------- --------------- -------------------- --------------------
DEPARTMENT_ID   10              Yes                  Yes
DEPARTMENT_ID   270             Yes                  No
DEPARTMENT_ID   280             No                   Yes
```

This output shows the index column for the table being compared and the index value for each row that is different in the shared database object. In this example, the index column is the primary key column for the `hr.departments` table (`department_id`). The output also shows the type of difference for each row:

- If `Local Row Exists?` and `Remote Row Exists?` are both `Yes` for a row, then the row exists in both instances of the database object, but the data in the row is different.
- If `Local Row Exists?` is `Yes` and `Remote Row Exists?` is `No` for a row, then the row exists in the local database object but not in the remote database object.
- If `Local Row Exists?` is `No` and `Remote Row Exists?` is `Yes` for a row, then the row exists in the remote database object but not in the local database object.

Tutorial: Converging Divergent Data

This example continues the scenario described in "Tutorial: Comparing Data in Two Different Databases" on page 10-4. Complete the steps in that topic before continuing.

When a shared database object has diverged at two different databases, you can use the CONVERGE procedure in the DBMS_COMPARISON package to converge the two instances of the database object. After the CONVERGE procedure runs successfully, the shared database object is consistent at the two databases. To run the CONVERGE procedure, you must specify the following information:

- The name of an existing comparison created using the CREATE_COMPARISON procedure in the DBMS_COMPARISON package
- The scan ID of the comparison that you want to converge

The scan ID contains information about the differences that will be converged. In this example, the name of the comparison is compare_departments and the scan ID is 1.

Also, when you run the CONVERGE procedure, you must specify which database "wins" when the shared database object is converged. If you specify that the local database wins, then the data in the database object at the local database replaces the data in the database object at the remote database when the data is different. If you specify that the remote database wins, then the data in the database object at the remote database replaces the data in the database object at the local database when the data is different. In this example, the local database ii1.example.com wins.

To converge divergent data in the hr.departments table at the ii1.example.com and ii2.example.com databases:

1. On a command line, open SQL*Plus and connect to the ii1.example.com database as the administrative user who owns the database link created in "Tutorial: Preparing to Compare and Converge Data" on page 10-3. For example, if the SYSTEM user owns the database link, then connect as the SYSTEM user:

   ```
   sqlplus system@ii1.example.com
   Enter password: password
   ```

   See *Oracle Database 2 Day DBA* for more information about starting SQL*Plus.

2. Run the CONVERGE procedure to converge the hr.departments table at the two databases:

   ```
   SET SERVEROUTPUT ON
   DECLARE
     scan_info    DBMS_COMPARISON.COMPARISON_TYPE;
   BEGIN
     DBMS_COMPARISON.CONVERGE(
       comparison_name  => 'compare_departments',
       scan_id          => 1,
       scan_info        => scan_info,
       converge_options => DBMS_COMPARISON.CMP_CONVERGE_LOCAL_WINS);
     DBMS_OUTPUT.PUT_LINE('Local Rows Merged: '||scan_info.loc_rows_merged);
     DBMS_OUTPUT.PUT_LINE('Remote Rows Merged: '||scan_info.rmt_rows_merged);
     DBMS_OUTPUT.PUT_LINE('Local Rows Deleted: '||scan_info.loc_rows_deleted);
     DBMS_OUTPUT.PUT_LINE('Remote Rows Deleted: '||scan_info.rmt_rows_deleted);
   END;
   /

   Local Rows Merged: 0
   Remote Rows Merged: 2
   ```

```
Local Rows Deleted: 0
Remote Rows Deleted: 1

PL/SQL procedure successfully completed.
```

The CONVERGE procedure synchronizes the portion of the database object compared by the specified scan and returns information about the changes it made. Some scans might compare a subset of the database object. In this example, the specified scan compared the entire table. So, the entire table is synchronized, assuming no new differences appeared after the comparison scan completed.

The local table wins in this example because the converge_options parameter is set to DBMS_COMPARISON.CMP_CONVERGE_LOCAL_WINS in the procedure. That is, for the rows that are different in the two databases, the rows at the local database replace the corresponding rows at the remote database. If some rows exist at the remote database but not at the local database, then the extra rows at the remote database are deleted. If instead you want the remote database to win, then set the converge_options parameter to DBMS_COMPARISON.CMP_CONVERGE_REMOTE_WINS in the procedure.

In addition, if you run the CONVERGE procedure on a shared database object that is part of an Oracle Streams replication environment, then you might not want the changes made by the procedure to be replicated to other databases. In this case, you can set the following parameters in the CONVERGE procedure to values that will prevent the changes from being replicated:

- local_converge_tag
- remote_converge_tag

When one of these parameters is set to a non-NULL value, a tag is set in the session that makes the changes during convergence. The local_converge_tag parameter sets the tag in the session at the local database, while the remote_converge_tag parameter sets the tag in the session at the remote database. If you do not want the changes made by the CONVERGE procedure to be replicated, then set these parameters to a value that will prevent Oracle Streams capture processes and synchronous captures from capturing the changes.

See Also:

- "About Tags for Avoiding Change Cycling" on page 4-10

Index

A

B

C

D

E

F

G

H

I

L

M

N

O

P

Q

R

S

T

U

V

W