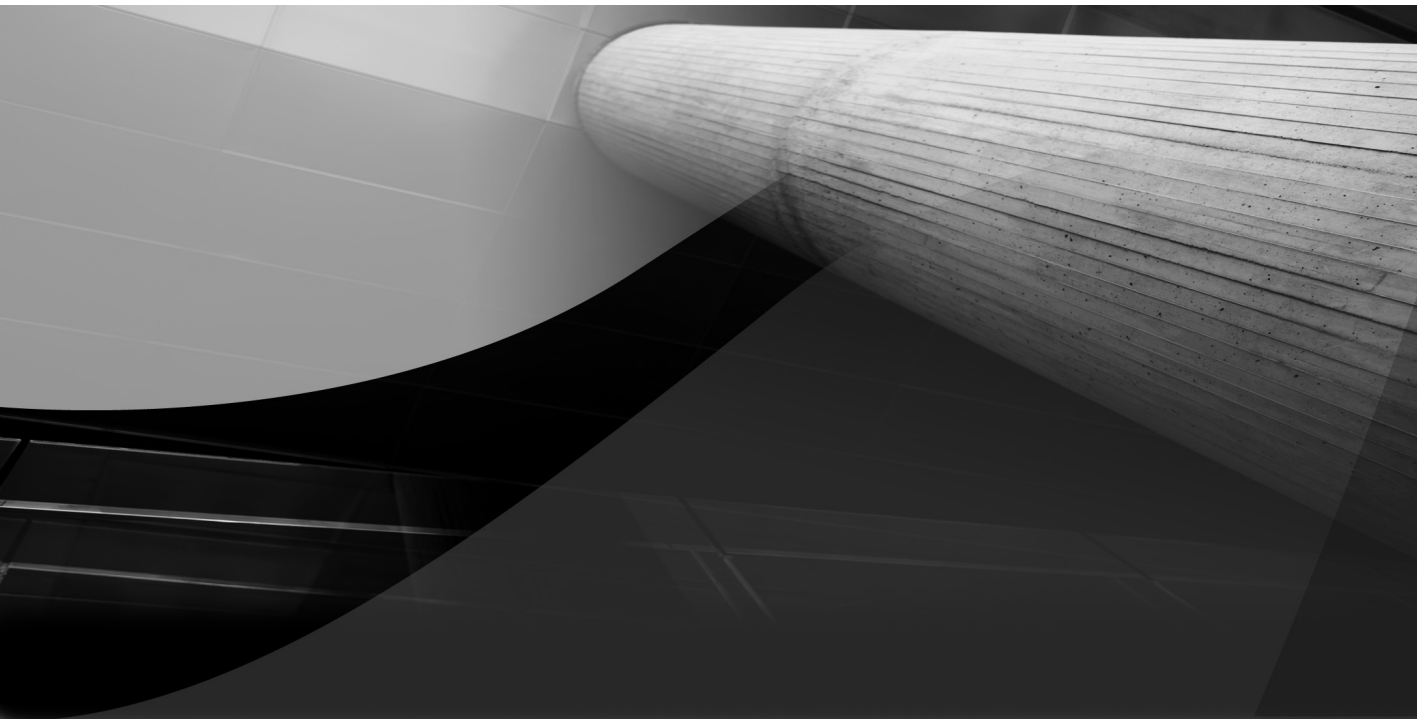


PART I

Introduction to Oracle Streams



CHAPTER 1

What Is Oracle Streams

4 Oracle Streams 11g Data Replication



Businesses operating in a distributed environment typically maintain several databases instead of one large data store. As a result, sharing data between the distributed databases and applications becomes a necessity. Applications and users expect access to the data in near real time. Over a period of time, businesses find themselves using a variety of products and customized applications to share the information. Oracle Streams provides the single information sharing solution.

Simply put, Oracle Streams is the managed flow of information. This information flow can be from one application to another, one database to another, within the same application, or within the same database. The applications and databases can be on the same machine or can be far apart. The databases can all be Oracle databases with different Oracle releases, different platforms (homogeneous environment), or a combination of Oracle and non-Oracle databases such as DB2 or Microsoft SQL-Server (heterogeneous environment).

Oracle Streams provides a solid infrastructure for such information flow. Oracle introduced Streams as an information sharing solution in Oracle9i Database Release 2. It has since been enhanced and improved significantly. Oracle Streams offers a number of features to match your information sharing requirements.

In Oracle Streams, the smallest unit of shared information is called a *message*. A message can be captured from the changes made to the database, or it can be generated by an event in the database. Typically, this includes data changes from the insert, update, and delete operations on the tables in the database. In addition, it can be a Data Definition Language (DDL) operation such as adding, altering, or dropping a table, an index, or a schema. However, certain other changes made to the database, such as adding a data file, starting a backup, or taking a tablespace offline, are not candidates for sharing.

The message can also be created by the user application and put in the Streams information flow. These application messages are generated outside the database. For example, when an application wants to shutdown, it can send a message to other applications. The application programs simply use Streams infrastructure to send messages to other applications, which will receive the message and process it. The message that is written can then be propagated to other databases or applications. You can manage the message as it flows through Oracle Streams.

Using Oracle Streams, you can control what messages to capture, how to propagate those messages, and how to consume, or apply, those messages when they reach their intended destination. Oracle Streams can capture database changes as a result of Data Manipulation Language (DML) and DDL commands. You can have total control over the message and its contents at each step of the way. It is possible for you to modify data in these messages using supplied procedures or by writing your own procedures. Such powerful flexibility enables you to build robust distributed databases and applications, replicated systems, and other high-availability solutions.

Prior to Oracle Database 11g, Oracle Streams captured information in an asynchronous manner by scanning the database redo information. Oracle Database 11g offers you an additional feature of *synchronous capture*. With synchronous capture, Oracle Streams captures database DML changes as they happen, and not afterward from the redo logs. Oracle Streams does not use database triggers to capture such information.

Although Oracle Streams has numerous applications, such as message queuing, data protection, data warehouse loading, and so on, this book discusses data replication using Oracle Streams.

Oracle Streams is a standard and integrated feature of Oracle Database 11g. However, the Standard Edition of Oracle Database 11g offers synchronous capture as the only method to automatically capture database changes. In Standard Edition, asynchronous capture is not available. You must have Enterprise Edition for that. You do not need to install any additional software and you do not have to license Oracle Streams separately.

Information Flow in Oracle Streams

In a simplistic form, here is how a message, or a change, flows through Oracle Streams following the standard publish-subscribe model. First, the capture process of Oracle Streams captures the message. The message is reformatted by the capture process and placed (staged or queued) in a staging area, which is typically an in-memory structure, called the *Streams queue*. The capture process thus publishes the message to the Streams queue. The processes that read these messages from the Streams queue are called consumer processes. These processes need to register an interest in receiving messages from the Streams queue. In other words, the processes subscribe to receive the messages. These can also be referred to as subscriber processes. The consumer or subscriber can either be a Streams process, such as an apply process, or an external application process. From the Streams queue, a message can be read, or dequeued, by the local consumer that will process the message, or it can be propagated to another Streams queue on another system to be consumed by a consumer or an apply process, for example.

If the same message is not put in another Streams queue, then its flow in Oracle Streams ends. The message remains in the flow until consumed by all intended consumers.

Figure 1-1 depicts the Streams information flow.

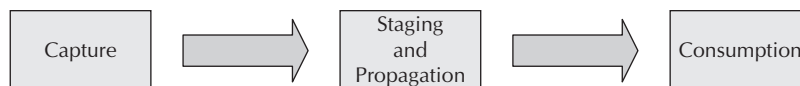


FIGURE 1-1. *Information flow in Oracle Streams*

6 Oracle Streams 11g Data Replication

The three major components responsible for the Streams information flow are

- Capture
- Staging and Propagation
- Consumption

In the Streams architecture, each of these major components has a few other subcomponents, processes, or configuration requirements. In Oracle Streams replication, these components coordinate automatic delivery of the message to its destination.

Architectural Overview of Oracle Streams

This section briefly reviews the Streams components, subcomponents, processes, configuration options, and configuration requirements. This will provide enough foundation for the in-depth discussion of how to configure and manage Streams replication environments in subsequent chapters.

Capture

As shown in Figure 1-1, the message flow begins at capture when Oracle Streams creates the message. Typically, the capture process, which is a database background process, creates the messages. This capture process is an asynchronous process that reads the redo log files to extract the database changes to create the messages. The synchronous capture process, available from Oracle Database 11g, uses a different mechanism to create the messages. The synchronous capture process captures the DML changes in real time to create the messages.

The messages are created in a particular data format and are called *Logical Change Records (LCRs)*. The messages created by the capture process are called *implicitly captured LCRs* and are enqueued into a buffer queue. The messages created by the synchronous capture process are also called implicitly captured LCRs but are enqueued into a persistent queue stored on the disk.

When messages are automatically captured, the process is referred to as *implicit capture*. The database that created the information contained in the message is called the *source database*.

Your external applications can also create messages in the LCR format, or your own format. When messages are created by user applications, the process is referred to as *explicit capture*. When you don't want to capture each and every change, you can specify which changes to capture. These instructions are called *rules* in the Streams environment. In other words, the Streams rules associated with the capture process

determine which changes the capture process captures. Such rules can be created automatically or manually. You can modify existing rules or define your own rules.

Log-Based Capture

Oracle Streams utilizes the Log Miner functionality to mine database redo logs to capture changes made to the database. Using the redo log information to capture changes guarantees data recoverability from database crashes or media failures. You do not lose changes that you want to capture, as long as the redo logs (or archived logs) are available. The mined information from the redo logs is presented to the capture process to identify changes to capture and convert the change information into an LCR.

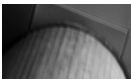
Local Capture

Typically, the capture process runs on the source database as a database background process. This is called the *local capture process*. It is local to the source database. It seamlessly scans, or mines, the in-memory redo log buffer, the online redo logs, and, when necessary, the archived logs to capture changes to the local database. The changes that satisfy the selection criteria defined by the capture rules are captured and converted into LCRs to be placed in an in-memory staging area called the Streamspool. This is the default behavior of the local capture.

Downstream Capture

Oracle Streams also provides an option to capture changes for the source database by running the capture process on another database server. In this case, the log files are written to the remote database server in addition to the source database server. Oracle Streams leverages Log Transport Services, Oracle Data Guard functionality, to write logs to the remote server. The capture process on the remote server mines the logs from the source database and stages them locally. These changes can be applied to the remote database if its apply process is the subscriber to these changes; otherwise, the changes can be propagated to another staging area for consumption.

Downstream capture offers a couple of benefits. First, you can offload the process of capturing changes from the production database to another database. Second, since remote writing of the redo log files is achieved using Data Guard protection modes (maximum availability, maximum performance, maximum protection), you have a choice to select an appropriate mode for your environment. It is possible to use one remote database to capture changes from multiple source databases.



NOTE

The capture process does not capture certain types of DML and DDL changes. Changes made to SYS, SYSTEM, and CTXSYS schema are ignored.

Synchronous Capture

Introduced in Oracle Database 11g, synchronous capture operates differently. Instead of mining the redo information to capture changes, synchronous capture captures the changes to the table as a result of a DML statement. As soon as the table data changes, this change or message is captured in real time by the synchronous capture process and converted to an LCR. The LCR is then written to the disk queue instead of the in-memory staging area. In Oracle Database 11g, synchronous capture does not capture changes as a result of DDL statements. Synchronous capture can be a better option when you want to replicate low-volume DML activity on a small number of tables.



NOTE

Synchronous capture only captures DML changes.

Staging and Propagation

The captured LCRs must be made available to the subscribers and consumers. Oracle Streams achieves this by staging the LCRs for propagation.

Staging

All captured messages are stored in a staging area. The staging area is an in-memory buffer and is part of the system global area (SGA) of the database instance. Messages created by user applications are also stored in this staging area. The LCRs created by the synchronous capture process are not stored in the memory queue, but they are stored in a disk queue table.

The messages remain in the staging area until consumed by the subscribers. The subscriber can read the contents of the staging area to select messages of their interest. The subscriber can be an application, a staging area, or the apply process of a different database. The applications can explicitly dequeue, or read, the message from the staging area to consume it. If the subscriber to this staging area is an apply process, then the messages will be dequeued and applied by the apply process.

Propagation

Messages in one staging area can be propagated to another staging area in another database using database links over Oracle Net. Streams offers a great deal of flexibility in choosing how messages are routed. Rules can be applied during propagation to select which messages will be propagated to another staging area. You can modify existing propagation rules or define your own rules.

In some cases, the propagation may not be required. The messages can be dequeued by the consumer from the same staging area where the capture process created the message. In this case, the publisher and consumer processes run in the same database.

Directed Networks Oracle Streams offers a capability to control how you propagate the messages within your network. Messages captured at one database can be published and propagated to or propagated through other databases anywhere in your network until they reach the intended subscriber or destination. This capability is referred to as *directed networks*.

Even if the source and the destination database do not have direct network communication, the messages can be directed through another, intermediary database that has network communication between the source and the destination database.

Figure 1-2 shows such a directed network. Messages from database A are sent to database C through database B. Database A propagates the message to the staging area in database B. Database B does not consume the messages locally, but only propagates those to the staging area in database C.

There could be more than one database in between A and C, or there could be more destinations than just database C.

Instead of sending the message to all destinations from the source database, the message can be sent only once to the intermediate database, which can send the same message to all the other destinations. The intermediate database simply forwards the contents of one Streams queue to another. This is called Queue Forwarding.

It is also possible to apply the message to the intermediate database and then capture it again using a capture process on the intermediate database to propagate it to other database. This is called Apply Forwarding.

Consumption

When a message is dequeued from the staging area, it is considered consumed. The apply process implicitly dequeues messages from the staging area. If the message is consumed by the apply process and applied to the objects in the database, then that database is called the *destination database*. The apply process runs locally on the destination database.

Your application, or process, can also explicitly dequeue messages from the staging area. The staging area can be local or remote to the user application.

Apply rules can determine which messages are dequeued and applied by the apply process at the destination database. By default, the apply process applies the

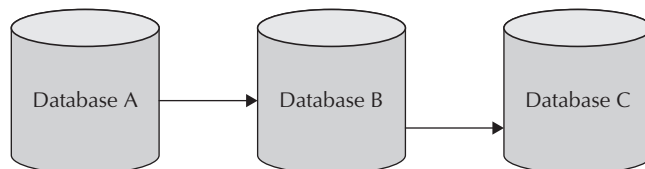


FIGURE 1-2. A directed network

captured LCRs, but you can intercept and process these LCRs with your own PL/SQL procedure. The apply process can also dequeue the message for the buffer queue and enqueue it into a persistent queue for your application to process it.

Default Apply Process

The default apply process is configured as multiple database background processes on the destination database. By default, it automatically applies the captured LCRs for the DML and DDL changes from the source database. The apply process can detect any data conflicts when applying changes to the destination database.

In a heterogeneous environment, you can configure the apply process to send the message to the remote non-Oracle database using appropriate Oracle Transparent Gateways.

Custom Apply Process

A custom apply process is similar to the default apply process in its functionality. The customization gives you total control over how the apply process handles the LCRs. You write the custom PL/SQL procedure. In Oracle Streams, this user-created procedure is called the *apply handler procedure*.

The apply handler procedure can be common to all LCRs or it can selectively apply to certain LCRs. You can also define different apply handlers to process the type of the DML change. For example, you can define separate apply handlers for INSERT, DELETE, and UPDATE operations against the same table. You can then choose to ignore all delete statements at the destination database for selected tables. You can also modify the DELETE command in the LCR to an UPDATE command, so that the delete statement is changed to an update statement to update a column (a delete indicator, for example) in the destination table.

With such flexibility, you can implement customized replicated databases to address unique business and statutory requirements.

Conflict Detection and Resolution

By design, the apply process detects data conflicts when applying the change to the database. A conflict occurs when the data in the destination row does not match the corresponding data in the LCR when the apply process is trying to identify the row for an UPDATE or DELETE operation. The LCR contains the old and new values of the column that changed in the source row, and it expects old values to be present in the row at the destination. If the old values do match, then there is a data conflict. In such cases, a conflict resolution procedure can be invoked, if needed. Oracle provides a number of prebuilt conflict handler procedures. You can use either the supplied conflict handler procedures as applicable, or write your own to resolve the conflict to satisfy your business requirements.

In case of unresolved conflicts or exceptions from the conflict handler procedure, the apply process puts the entire transaction into a persistent error queue. You can re-execute the error transaction after correcting the data conflict problem. If desired, you can also delete the error transaction if you resolved the conflict by some other means that does not require re-executing the error transaction.

Queues

Queues can be viewed as a storage location for messages. Applications can send messages to the queue and retrieve messages from the queue. When an application wants to communicate with some other application or process, it can leave a message in the message queue and then the other application will be able to retrieve the message from the queue.

Oracle Streams components use queues to exchange messages. Message queues offer a means of asynchronous communication in a failsafe manner for different processes or applications. The queues support operations to enqueue messages, to dequeue messages, and to propagate the messages to other queues or systems.

A message typically is made of two parts: the message control information and the content or payload. The content of the message can be of specific data type or can be raw data type. Oracle Streams has a very generic data type called ANYDATA. All LCRs must be staged in the ANYDATA queue.

The Oracle Streams Advanced Queuing feature supports messages of type ANYDATA or any abstract type. The main advantage of the ANYDATA queue is to allow applications to send messages of different types in the same queue. A queue can be persisted in the database using one or more tables.

Streams Tag

The redo information for all database changes contains a marker, or a tag. By default, the value of this tag field is NULL and it does not consume any space in the redo record. The data type of the tag field is RAW and its size limit is 2000 bytes. The tag field becomes part of the LCR when the capture process converts the redo information into an LCR.

Oracle Streams leverages this feature of the database to identify and track if a particular LCR was generated by a particular database. Such identification is mandatory when configuring bidirectional or multidirectional replication environments. Lack of such identification will cause the messages to recycle back to their original source database.

By default, the Streams rules for capture, propagate, and apply processes check the value of the tag field. LCRs with the tag field set to NULL are processed, while LCRs with a non-NULL value are discarded. The apply process on the destination database sets the tag field value to hexadecimal 00 (zero) when applying the change to the destination database. Thus, the redo information generated by the transactions

12 Oracle Streams 11g Data Replication

executed by the apply process will have a non-NULL tag field. So, if there were a capture process for bidirectional replication, it will ignore the LCR for such a change made by the local apply process, thereby avoiding the change recycling.

You can also leverage this functionality to temporarily suspend replication of certain actions. You can modify the tag value for your session at the source database so that the capture process will ignore all the LCRs from the redo records generated in your session. Then, you may have to perform the same actions at the destination database to keep data in sync with the source database. You can reset the tag value for your session back to its original value to resume normal replication, or simply exit the session.

Rules and Rule Sets

Oracle Streams uses rules to control the message capture, message propagation, and message consumption. A rule is a database object, such as table or index. It is specified as a condition when configuring Streams components. The rule condition is similar to the WHERE clause of a SQL statement.

The rule has the following components:

- **Rule condition** This is a combination of one or more expressions that returns a Boolean value (TRUE, FALSE, or NULL).
- **Evaluation context** This defines external data that can be referenced by the rule while evaluating the rule condition. The external data can be a variable, table data, or both.
- **Action context** This is optional information that is interpreted by the client of the rules engine while evaluating the rule condition. The capture, propagation, and apply processes are the clients of the rules engine.

Related rules are grouped together into a rule set, and a rule set is associated with a Streams component.

Oracle Streams supports two types of rule sets:

- **Positive rule set** If the rule in a positive rule set evaluates to TRUE, then Streams will include the LCR for processing.
- **Negative rule set** If the rule in a negative rule set evaluates to TRUE, then Streams will discard the LCR.

You can have both, positive and negative, rule sets defined for a Streams component. In such a case, the rule from the negative rule set is evaluated first. If it evaluates to TRUE, then the positive rule set is ignored, since the message will be discarded.

**NOTE**

The synchronous capture process can only have a positive rule set.

Oracle generates required rules and rule sets if you do not create them when configuring Streams replication. These are called system-generated rules and rule sets. For most simple replication environments, such system-generated rule and rule sets are sufficient. You can create your own rules and rule sets. You can modify system-generated rules to match your requirements.

Instantiation

When replicating table changes from the source database to the destination database, the destination database must contain the copy of the table. If the destination database does not contain the table, then it must be created, or *instantiated*, from the source database. There are a number of ways to instantiate the table. You can use Create Table As Select (CTAS), data pump, export/import, transportable tablespaces, split mirror copies, or recovery manager (RMAN), depending on your environment and needs.

First, the table has to be prepared for instantiation at the source database. During the replication configuration, Oracle automatically prepares the tables for instantiation. You can also prepare the table for instantiation using supplied procedures. At this point, Oracle records the database system change number (SCN) and populates an internal Streams data dictionary with the global name of the database, the table name and its object number, the column name and column number, and so forth. Next, if the table did not exist at the destination, it must be created with contents from the source database. And lastly, the destination table's instantiation SCN must be set to the SCN from the source database. The data pump and export/import utilities can set the instantiation SCN for the table at the destination database when importing data. You can also use a supplied procedure to set the instantiation SCN for the tables in the destination database.

The instantiation SCN controls which LCRs containing database changes are applied to the destination database by the apply process and which LCRs are ignored. If the commit SCN in the LCR for the source table is greater than the instantiation SCN for the table in the destination database, then the apply process will apply the change to the table. Otherwise, the LCR will be ignored. Oracle will not report any warning or error when ignoring such LCRs.

LogMiner Data Dictionary

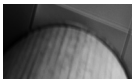
Oracle database processes use the database data dictionary to map object numbers, object version information, and internal column numbers to table names, column names, and column data types. The data dictionary is always kept in sync with the current database configuration.

14 Oracle Streams 11g Data Replication

The Streams capture process needs its own data dictionary because the current information in the database data dictionary might not apply to the redo information from the redo or archive log files that the capture process is reading. This information could have been generated first, and the database data dictionary could have changed before the capture process scanned the log file.

The data dictionary used by the capture process is called the LogMiner data dictionary. Oracle extracts the database data dictionary information in the redo logs when the very first capture process is created. The capture process, when started for the first time, reads this data dictionary information from the redo logs and creates the LogMiner data dictionary. The contents of this LogMiner data dictionary are maintained in internal LogMiner tables.

There can be multiple LogMiner data dictionaries for the source database. Multiple capture processes can share a common LogMiner data dictionary or each process can have its own LogMiner data dictionary.



NOTE

The synchronous capture process does not use the LogMiner data dictionary.

Streams Data Dictionary

Similar to the capture process, the propagation and apply processes need their own data dictionary to keep track of the object names and numbers from the source database. When objects are prepared for instantiation in the source database, information about the instantiation is written to the redo log along with object details. The capture process reads this information and populates what is called the Streams data dictionary in the database where it is running. For a local capture process, the Streams data dictionary will be in the source database, whereas for a downstream capture process, it will be in the downstream database. The Streams data dictionary is updated as and when objects are prepared for instantiation.

The propagation process requires the object mapping information in the Streams data dictionary from the source database when it evaluates rules to process the captured LCRs. Oracle automatically populates a local multi-version Streams data dictionary at each database that has the propagation process configured.

Similarly, the apply process requires object mapping information in the Streams data dictionary from the source database when it evaluates rules to process the captured LCRs. Oracle automatically populates a local multi-version Streams data dictionary at each destination database that has the apply process configured.

NOLOGGING and UNRECOVERABLE Operations

Oracle Streams captures database changes from the redo logs. When DML operations are performed with the NOLOGGING option, where applicable, the redo information is not generated. Using the UNRECOVERABLE option in the SQL*Loader direct path

load also suppresses the redo generation. In these situations, the DML changes are not captured due to their obvious absence in the redo log. To replicate these changes successfully, you should avoid the use of the `NOLOGGING` and `UNRECOVERABLE` operations.

To ensure proper logging of changes made to tables, you may want to set `FORCE LOGGING` at the tablespace level or at the database level. Once this is set, Oracle will silently generate redo log information for all `NOLOGGING` and `UNRECOVERABLE` operations.

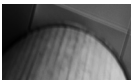
If you must use the `NOLOGGING` and `UNRECOVERABLE` operations for performance reasons on the source database, you will have to perform the same operations at the destination database to preserve data synchronization. Otherwise, the data mismatch at the destination database can result in apply process errors in subsequent DML operations.

Supplemental Logging

The database changes are recorded in the redo log with required information to perform database recovery in the event of an instance or media failure. Oracle Streams uses the same redo log information to create messages (LCRs) to apply to the destination database. The redo information contains the values for the columns that changed at the source database. This information may not always be sufficient to identify correctly the row in the destination table to apply the same change.

Supplemental logging is a process of recording additional column data into the redo log. The capture process inserts this additional information in the LCRs. The apply process uses this additional information to correctly identify the row to which to apply the change.

In situations where the application maintains data integrity outside the Oracle database, and the tables do not have primary key or unique constraints, it becomes necessary to configure adequate supplemental logging. Sometimes it requires supplemental logging of all columns in the table.



NOTE

Supplemental logging is always configured at the source database irrespective of the capture process location—local or downstream.

Supplemental logging can be configured at the database level or at the table level. At the database level, it can be configured to record additional information in the redo log to identify the rows in the redo log, or it can be configured to record the before and after values of specified types of columns, such as primary key columns, unique index columns, foreign key columns, or all columns of the table.

If a table does not have a primary key constraint or unique index, then it becomes necessary to identify a set of columns for supplemental logging.

At the table level, supplemental logging creates separate log groups containing the column names for each table. Such logging can be conditional or unconditional. *Conditional supplemental logging* records in the redo log the before image of all the specified columns only when one of those columns is updated. *Unconditional supplemental logging* records in the redo log the before image of all the specified columns regardless of whether or not any of the columns changed. Sometimes this is called “always logging.” Unconditional logging is necessary for all the columns used for row identification.



NOTE

The synchronous capture process does not need supplemental logging information.

Logical Change Records

As mentioned earlier, the capture process reformats the information for the database change it captured from the log file. The reformatted message is called a Logical Change Record and represents the database change.

The following two types of LCR can be created by the capture process:

- **Row LCR** Each row LCR (sometimes called DML LCR) represents the change made to a single row. In addition, there could be multiple LCRs for a change made to a single column with a data type of LONG, LONG RAW, LOB, or XMLType stored as CLOB. Also, a single DML statement can affect multiple rows, causing creation of multiple LCRs.
- **DDL LCR** This LCR represents the change made by a DDL command.

These LCRs contain enough information to apply the change to the destination database. In addition, you can include extra information in the LCR for auditing and tracking purposes.

Although the LCR has an internal data format used by Oracle Streams, there are procedures to access and modify the information contained in the LCR.

Table Data Comparison

Oracle Database 11g includes procedures to compare and synchronize (merge) data in shared tables in a distributed and replicated environment. The procedures in the DBMS_COMPARISON package can be used to compare table data without interfering with other applications. These procedures allow you to compare data for the entire table or for data subsets or data ranges. The comparison can be done on a periodic basis or whenever needed. Data consistency can be checked at the row level or table level. The identified differences can be viewed, if there are any. These data

differences could arise due to incomplete transactions, unrecoverable errors, and so forth. When data differences are found, the supplied procedures can be used to merge the differences and confirm that the table data mismatch is resolved.

Summary

Oracle Streams is an information sharing solution. It offers a robust and flexible infrastructure to manage information flow between Oracle and non-Oracle databases. Using the redo information, Oracle Streams can capture and easily replicate database changes seamlessly within your network. The Streams rules for capture, propagation, and apply processes offer customization to control the selection, routing, and consumption of messages to address your business needs. With the availability of synchronous capture, you can replicate data from databases using Oracle Database 11g Standard Edition. The downstream capture configuration can offload the logging process to another database to minimize load on the production system. Oracle Streams is an integrated feature of the Oracle Database software.