Oracle Database 11g: Oracle Streams Replication

An Oracle White Paper
July 2007

ORACLE

EXECUTIVE OVERVIEW

Oracle Database 11g provides a unified solution for information sharing—Oracle Streams. Oracle Streams captures and distributes database updates, events, and application messages. It can automatically apply updates to destination databases, or pass events and messages to custom procedures and applications. Combining these capabilities provides an extremely flexible solution for replication, message queuing, and event notification solutions. Unlike traditional solutions, you can use Streams to create powerful information sharing solutions that incorporate some or all of the capabilities of classic information sharing solutions, all through the use of a single feature.

This paper discusses how the three basic elements of the Oracle Streams technology (capture, stage, consumption) are used to replicate information in the Oracle 11g database.

ORACLE STREAMS OVERVIEW

Oracle provides a number of technologies that support the goal of information consolidation, but there can be compelling business requirements that dictate sharing information. For example, some remote locations may not have good connectivity to the primary site or may prefer to maintain site autonomy. In other cases, the data may be consolidated, but the applications may need a method of communicating with one another. There are nearly as many different reasons for needing to share information between locations or applications, as there are businesses. Fortunately, Oracle Database 11g provides a single, unified solution for information sharing—Oracle Streams. It is the flexibility of Oracle Streams that allows it to solve each of these disparate information-sharing needs. Even if it's not possible to create a single, unified database, it is still possible for the data to appear unified to the end-user.

Oracle Streams provides a set of elements designed to facilitate the capture, staging, and consumption of events within the Oracle database. These events include both messages enqueued into a database queue by applications, as well as, modifications to database objects, such as tables or procedures, using DML or DDL.

Each element of Streams can be configured explicitly or implicitly. Explicit capture allows user applications to explicitly enqueue messages into a staging area on the source database. Using the implicit capture mechanism of Oracle Streams, changes

made to a database can be efficiently captured and replicated to one or more remote systems with little impact to the originating system. This capture mechanism extracts both data changes (DML) and structure changes (DDL) from the redo log and publishes these updates to the staging area.

Both explicitly and implicitly captured changes can be propagated to one or more remote staging areas where they can be applied at the destination site. Changes in a staging area are consumed by the apply engine, where the changes they represent are applied to a database, or they are consumed by an application. Oracle Streams includes a flexible apply engine, that allows use of a standard or custom apply function. This enables data to be transformed when necessary.

Existing applications can use Oracle Streams with minimal modification or special handling. Oracle Streams can specify rules at multiple levels of granularity: database, schema, and table. Each element of Oracle Streams can use these rules to easily configure, for example, the capture of changes for an entire database, or a set of schemas, or individual tables.

Using these basic elements, users control: how information is put into a stream, how the stream flows or is routed from node to node, what happens to events in the stream as they flow into each node, and how the stream terminates. By specifying the configuration of the elements acting on the stream, a user can address specific requirements. To simplify the deployment of Oracle Streams, Oracle provides applications specially configured for specific markets.

Although, as described above, Oracle Streams can be used to implement a variety of information sharing solutions, the focus of this paper is on using Oracle Streams in a replicated environment.

Unique Databases

Databases replicated using Oracle Streams technology need not be identical. Participating databases can maintain different data structures using Streams to transform the data into the appropriate format. Streams provides the ability to transform the stream at multiple points: during capture, while propagating to another destination, or during application at the destination site. In addition to supporting user-defined transformations, Oracle provides easy to use declarative transformations for the most common transformations, such as changing the name of a column or table.

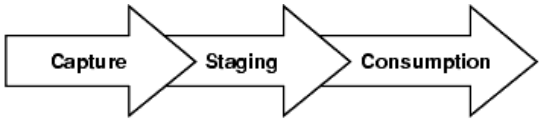
The data at each site can be subsetted based on content as well. For example, you could implement a rule that only events (or changes) related to the employees of a particular division, based on the department identifier column, be applied to a particular table. Oracle Streams automatically manages the changes to ensure that the data applied to the table matches the subset rule criteria.

Directed Networks

Although locally captured or enqueued events can be consumed locally, they can also be propagated to one or more remote locations. The directed network capability of streams allows changes to be directed through intermediate databases as a pass-through. Changes at any database can be published and propagated to or through other databases anywhere on the network. By using the rules-based publish and subscribe capabilities of the staging area queues, database administrators can choose which changes are propagated to each destination database, and can specify the route messages will traverse on their way to a destination. This directed network approach is also friendly to Wide Area Networks (WAN), enabling changes to subsequent destinations to traverse the network once to a single site for later fan-out to other destinations, rather than sending to each destination directly.

ORACLE STREAMS ARCHITECTURE

There are three basic elements of Oracle Streams: capture, staging, and consumption.



Capture

Oracle Streams supports capture of events into the staging area. These events can be captured either explicitly or implicitly. Explicit capture allows applications to explicitly generate events and place them in the staging area. Implicit capture automatically captures changes to a table using one of the following techniques: log-based capture or synchronous capture With log-based capture, the server captures DML and DDL events for a source database by mining the source redo logs, log buffer, and archive logs locally. Alternatively, the server can mine the redo logs or archived logs of the source database at an alternate database, assuming the alternative database is on a similar platform type and operating system. With synchronous capture, DML changes are captured via an efficient internal mechanism during the user's transaction activity.

Logical Change Records

Oracle Streams uses logical change records, or LCRs, to describe changes made to a single row of a table modified with a single DML statement. A single DML statement that operates on multiple rows within a table will generate multiple LCRs, and a single transaction can consist of multiple DMLs. Each logical change record includes the name of the changed table, the old and new values for any changed columns, and the values for the key columns. Armed with this information, changes can be applied to the correct rows at the destination sites and conflicts can be detected.

It can be important to differentiate between changes that originate at different sites, so each LCR is tagged with identification information from the source database about the origin of the change. These tags are typically used during the consumption phase of Streams. In certain cases this information is used to filter out changes that originated at another site and were made by the apply process, to prevent their being placed in the staging area and cycling back to the originating site. This is most notably the case in an N-way replication configuration, where each replicated site communicates its changes directly to every other site in the configuration. In other cases, for example a hub and spoke configuration, it is important to capture all changes, including those that originate at other sites. In these cases, it may be desirable to have changes from an individual spoke reflected through the hub to other spokes in the configuration.

Log Based Capture

One of the key features of Oracle Streams replication is support for log-based capture. Log-based capture leverages the fact that changes made to tables are logged in the redo log to guarantee recoverability in the event of a crash or media failure. Capturing changes directly from the redo logs minimizes the overhead on the system. Oracle can read, analyze, and interpret redo information, which contains information about the history of activity on a database. Oracle Streams can mine the information and deliver change data to the capture process. Tables identified for replication with Oracle Streams must log supplemental information into the redo stream, such as primary key columns, to facilitate the delivery of this information.

The capture process consists of the following components: a reader server, one or more prepare servers, and one builder server. The reader server reads the redo log and divides the redo log into regions. The prepare servers scan the regions defined by the reader server in parallel and perform prefiltering of changes found in the redo log. The capture process can intelligently filter LCRs based upon defined rules. Thus, only changes to desired objects are captured.

The builder server merges redo records from the preparer servers and passes the merged redo records to the capture process. The capture process then formats the change into a logical change record. If the LCR is found to satisfy the defined rules it then enqueues the LCR into the staging area for further processing. Each reader server, preparer server, and builder server is a parallel execution server. The capture process is an Oracle background process.

Mining the Redo Log

Oracle Streams supports mining from the in-memory log buffer, hot mining of the active redo log, as well as mining archived log files. In the case of hot mining, the redo stream is mined for change data at the same time it is written to the active redo log, reducing the latency of capture. Streams seamlessly traverses the log buffer, redo log, or archived log as necessary without any additional configuration.

Only LCRs that satisfy the selection criteria for capture are placed in the staging area.

Streams provides the ability to capture changes for a source database at a different server. Redo transport services uses the log writer process (LGWR) at the source database to send redo data to the downstream database asynchronously. At the same time, the LGWR records redo data in the online redo log at the source database. The remote server, capturing changes into the staging area on this remote server, then mines these remote logs. Should the remote server be a subscriber to the changes, they can also be applied. This allows for complete offloading of Streams activities from the production database server--often a critical requirement for high volume OLTP databases. Secondly, since the log files are written remotely using the standard Oracle tools, you can configure a protection level that is appropriate to your environment. A single remote server can be used to capture changes for multiple source databases.

Oracle supports real-time mining of production redo logs at a downstream database via standby redo logs. Although Oracle supports only one real-time downstream capture process at a downstream database, this real-time process can co-exist with multiple downstream archived-log processes.

Synchronous Capture

In contrast to log-based capture, synchronous capture captures changes to a table as they happen. Synchronous capture is useful in situations where you need to replicate only a small subset of tables that change infrequently in a highly active database or in configurations where capture from the redo is not available.

Synchronous capture uses an efficient internal mechanism to capture DML changes to specified tables while the transaction is in progress. When a DML change is made to a table, it can result in changes to one or more rows in the table. Synchronous capture captures each row change and converts it to a Logical Change Record (LCR**)**. After capturing an LCR, synchronous capture writes a message containing the LCR to disk into a persistent staging area (queue).

Synchronous capture offers the same tagging capabilites as log-based capture. Each LCR is tagged with identification information from the source database about the origin of the change. These tags are typically used during the consumption phase of Streams. In certain cases this information is used to filter out changes that originated at another site and were made by the apply process, to prevent their being placed in the staging area and cycling back to the originating site. This is most notably the case in an N-way replication configuration, where each replicated site communicates its changes directly to every other site in the configuration. In other cases, for example a hub and spoke configuration, it is important to capture all changes, including those that originate at other sites. In these cases, it may be desirable to have changes from an individual spoke reflected through the hub to other spokes in the configuration

Staging

Once captured, events are placed in a staging area. The staging area is a queue designed to store and manage captured events. Database changes, already formatted as LCRs by the capture process, are stored in an in-memory queue until consumed by all subscribers. LCRs captured via synchronous capture are stored within a disk staging area. Events explicitly queued to the staging area by applications for future processing are also maintained in the staging area. The queue provides a holding area with security, as well as auditing and tracking of event data.

Subscribers examine the contents of the staging area and determine whether or not they have an interest in the message representing that event. A subscriber can either be a user application, another staging area (usually on another system), or the default apply process. The subscriber can optionally evaluate a set of rules to determine whether the message meets the criteria set forth in the subscription. If so, then the message will be consumed by the subscriber.

If the subscriber is a user application, the application will explicitly dequeue the message from the staging area in order to consume the message. If the subscriber is another staging area, the message will be propagated and enqueued to that staging area. If the subscriber is an apply process, the messages will be dequeued and consumed by the apply process.

Propagation

Events in the staging area can be propagated to staging areas in other databases. To simplify network routing and reduce WAN traffic, events need not be transmitted to all databases and applications. Rather, they can be directed through queues on one or more systems until they reach the subscribing system. For example, consider a network between three sites A, B, and C where sites A and B can communicate directly, as can B and C, but sites C and A do not have direct communication. Streams is configurable to allow the changes at site A to be reflected at site C via site B. Site B can pass the events through to site C without requiring site B to apply the change locally. Of course, if the change is also desired at this intermediate site as well, the change can be applied at site B without impacting the pass-through to site C.

Administrators have a great deal of flexibility to use in specifying the routing of the streams. By using the rules-based publish and subscribe capabilities of the staging area queues, they can choose which changes are propagated to each destination database, and can specify the route messages will traverse on their way to a destination. Throughout the system, Oracle Streams maintains the source database information for each event, as it can be important to differentiate between changes that originate at different sites. The administrator controls whether to select all changes, including those that originate at other sites, or only those changes made at the local site and not by the apply engine

Consumption

Messages in a staging area are consumed by the apply engine, where the changes they represent are applied to a database, or they are consumed by an application. The Oracle Streams apply engine applies DML changes, DDL changes, user-supplied LCRs, and user-enqueued messages. When the destination database is an Oracle database, the apply engine runs locally on the system hosting the Oracle database. For greater flexibility, the Oracle Streams apply engine supports both default and custom apply procedures. Support for explicit dequeue allows application developers to use Oracle Streams to notify applications of changes to data, while still leveraging the change capture and propagation features of Oracle Streams. With Oracle Streams replication, you can choose to use either the default apply procedure, or to create custom apply procedures as needed.

Default Apply

The default apply engine can apply automatically captured DML and DDL changes, as well as user-supplied LCRs. The default apply engine detects conflicts where the destination row has been changed and does not contain the expected original values. If a conflict is detected, a declarative or user-written conflict resolution routine may be invoked, if desired.

The apply engine is a background process consisting of a set of server processes: an apply coordinator, a reader server, and one or more apply servers. The reader server assembles transactions for capture-generated LCRs. The apply coordinator performs transaction dependency and DML level dependency scheduling to maximize concurrency. The apply servers actually apply the changes. Since the apply coordinator and apply servers are typically in the same Oracle instance, there is no network roundtrip involved in dependency scheduling. The apply engine can invoke declarative transformations or user-supplied functions to transform changes to the correct format or name for each LCR.

Changes from multiple databases should be sent to multiple destination staging areas. Multiple apply engines can then be used to apply the changes from each source database concurrently. Each apply engine only applies changes from one source site. Throughout the system, Oracle Streams maintains the source database information, as it can be important to differentiate between changes that originate at different sites. The administrator controls whether to capture all changes, including those that originate at other sites, or only those changes made at the local site and not by the apply engine.

Customized Apply

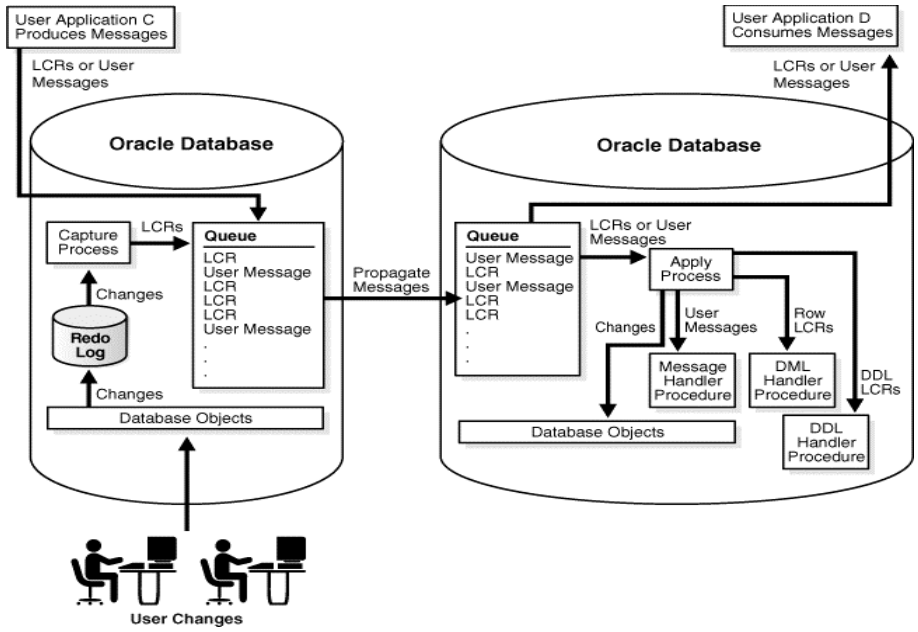
Streams provides the database administrator total control over the apply process. User-supplied PL/SQL procedures can be registered with Streams to customize the apply processing. With custom apply, the apply engine passes the LCR or user-enqueued message to the registered user-supplied procedure, or apply handler. This provides greater flexibility in processing the message. To further maximize

flexibility in apply handing, users can define multiple apply handlers:.: DDL handlers for managing replicated DDL; Message handlers for processing non-LCR messages: commit handlers to enable post-processing of the transaction as part of the transaction commit; as well as separate apply handlers for each type of DML operation (inserts, updates, or deletes) performed on a particular table.

For example, using this custom apply capability, a user could write a procedure to skip the apply of all deletes for the Employee table. Only the behavior of LCRs that perform deletes on the table Employee are affected in this scenario. Inserts and updates to the Employee table continue to be applied using the default apply engine. Thus no delete LCRs from other sites will be performed on the Streams site replica and all of the rows in that Employee table will still exist at the replica site. These custom DML handlers allow the user to exercise complete control over the behavior of the apply engine, enabling users to implement custom Streams sites to satisfy any business requirement.

ORACLE STREAMS REPLICATION

The following illustration demonstrates the streams architectural elements as used in replication. Oracle Streams replication captures changes from a source database, stages and propagates those changes to one or more remote databases, and then consumes or applies the changes at each target database.



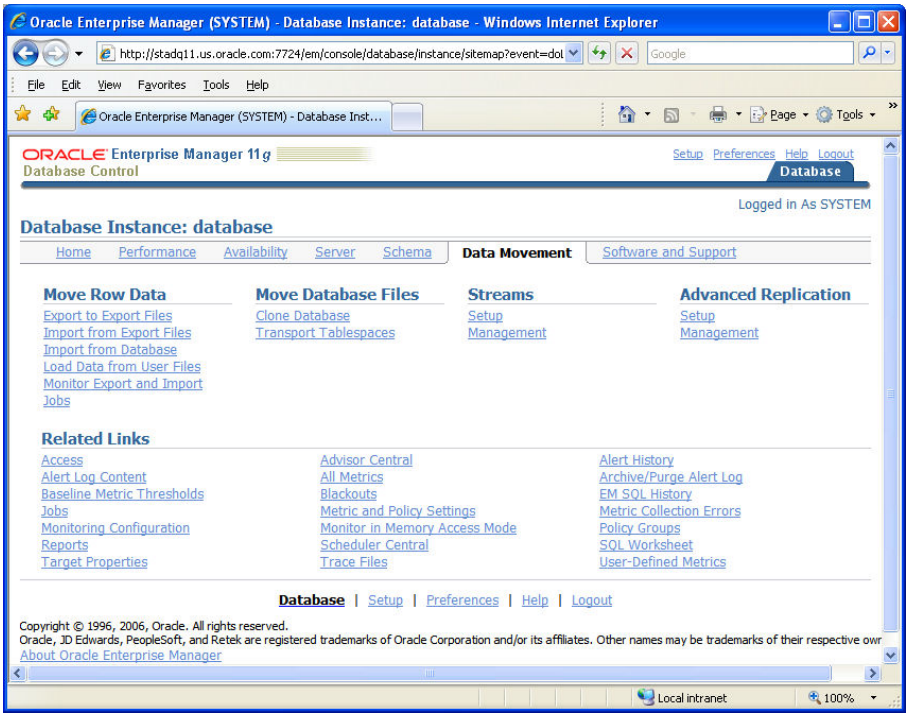
The lower left side of the figure shows user DML activity on one or more database tables , for example, an update statement that changes the state for a row in the EMP table representing an employee whose employee id (empid) is 100. As usual, the change is recorded in the redo log. The capture process, previously configured

to collect changes made to the EMP table, retrieves the changes from the redo log, formats the information into logical change records, and places the LCR into the staging area at the local database. Note that although in this example, the capture process is shown at the source database, for performance reasons, you may choose to run the capture process at another location. The captured LCR is then propagated from the staging area (or queue) at this source database and delivered to another database that has subscribed to the changes on the EMP table. In addition, the apply engine at this second database is configured to receive the changes on the EMP table. Once the change has been propagated to the new site, the local default apply process at this second database automatically applies the change to the database.

Configuring a Replication Environment

Oracle Streams provides an easy way to instantiate replicated objects at the target sites, using Oracle export and import utilities or the RMAN duplicate database feature. The instantiation is performed at the target database without affecting any existing sites in the Streams configuration. All sites continue to process their own work while instantiation to new sites is in progress. After a new object is prepared at the source site to capture changes for replication, the Streams replication metadata for the object is marked with the appropriate SCN, to ensure no changes are lost. As copies of the objects are created at the target site using export and import, the system will note the source SCN of the copied data, and will begin applying any changes that committed after the object was copied.

To simplify configuration, administration and monitoring of Oracle Streams environments, Oracle provides a Streams tool within the Oracle Enterprise Manager Database Control. The Streams tool provides wizards that you can use to configure a Streams replication environment. You can also use this Streams tool to generate scripts, which you can then modify to meet your specific requirements.



A customized API, tailored for replication, provides another way to easily configure common replication scenarios. Using commands in the DBMS_STREAMS_ADM package, a database administrator can quickly implement a replication environment at the table, schema, or database level. This package performs multiple configuration steps for the administrator dependant on the type of Streams process specified including the automatic generation of rules for the process. For example, the following API call configures a Streams environment that replicates changes made to the HR schema between the local database (source_database=>null) and NYC database:

```
DBMS_STREAMS_ADM.MAINTAIN_SCHEMAS(
     schema_names=>'HR',
     source_directory_object=>null,
     destination_directory_object=>null,
     source_database=>null,
     destination_database=>'NYC',
     bi_directional=>TRUE,
     instantiation=>
DBMS_STREAMS_ADM.INSTANTIATION_SCHEMA_NETWORK
);
```

Issuing this simple command results in the creation of a bi-directional replication environment, which will replicate DML changes between two sites. The source and destination directory object parameters are set to null, because Oracle is using the network option of Data Pump to perform the initial instantiation of the replicated site.

The Data Pump export/import operation uses the Streams pool at the replication sites. The Streams pool is a portion of memory in the System Global Area that is

used by Streams replication to provide memory for the capture and apply processes. The size of this pool is automatically managed by Oracle's Automatic Shared Memory Management feature when the SGA_TARGET initialization parameter is set to a nonzero value.

As described in the following section, the specification of conflict resolution routines in a bi-directional replication environment is strongly recommended. The administrator can perform each of these steps individually, if desired, using other packages related to rules and Streams processing.

Conflict Resolution

When Oracle Streams is used to support replication, both the source and destination databases are fully available to end-users for reading and writing during any replication activity. Because users can update different copies of the same table anywhere, it is possible for changes made at different database sites to update the same data element at the same time, resulting in an update conflict. The default apply mechanism will detect these conflicts as incoming replication changes are applied. Oracle provides built-in conflict resolution routines, such as "latest timestamp" or "overwrite", to automatically resolve potential conflicts. Different resolution methods can be chosen for different tables. Users also have the option to create their own routines to employ resolution rules tailored to their particular business needs. Any unresolved conflicts are logged in the database for special handling or re-execution after the conflict is resolved manually.

Comparing Table Data

When sharing data in multiple locations, verifying data consistency between databases is essential. The Oracle database includes a package to compare table data between databases, DBMS_COMPARISON. Complete data, data ranges, or data subsets may be checked on a periodic or as needed basis, without interference to running applications. Data consistency can be confirmed at the table or row level and differences can be re-examined in case of transient differences due to in-flight transactions. If necessary, divergent data can be converged so that both compared databases reflect the same data.

Comparisons can be made between tables, views, or materialized views by creating a comparison definition. The definition identifies the two objects (tables, views, or materialized views) for comparison and their location, either within the same database or residing in different databases. After the definition is established, the data can be compared.

Streams Advisor

A Streams environment typically includes multiple databases. Streams includes dictionary views that provide a comprehensive view of the entire Streams topology. This topology identifies individual streams of messages and the Streams components (queue, capture, propagation, apply) configured in each stream. The

topology information is determined using the Streams Performance Advisor package.

The Streams Performance Advisor also reports performance measurements for a Streams topology, including throughput and latency measurements. The Streams Performance Advisor also identifies any potential bottlenecks in a Streams topology so that they can be managed effectively.

CUSTOMIZING ORACLE STREAMS REPLICATION

It is possible to control which events, or changes, are captured, propagated and applied using a combination of Oracle Streams rules and transformations.

Rules

Each element of Oracle Streams uses a set of rules to distinguish the events to be managed. These user-specified rules are enforced by capture to selectively enqueue change events into the staging area. Similarly, the apply engine will selectively apply changes based on the rules defined for apply. Rules are also used to selectively propagate events between staging areas. Rules can specify the selection of the entire database, or schemas, as well as individual tables. Rules are SQL-like expressions that evaluate to either TRUE or FALSE. Rules can also be used to limit events based on the content of the event itself. An example of a rule limited to any untagged DML changes to the HR schema is:

```
:dml.get_object_owner() = 'HR' AND
     :dml.is_null_tag() = 'Y'
```

Note the use of the tag in this rule. In most cases, changes local to a database have no tag (null tag,) while changes applied from other databases are tagged in the redo log with a non-null value.

For simplicity, rules are organized into named collections referred to as rule sets. Rules can easily be added to or subtracted from the rule set. Multiple rules can belong to a rule set and multiple rule sets can contain the same individual rules. The DBMS_RULES_ADM package is provided to facilitate the management of these rules.

Each Streams process uses two rule sets, a positive rule set and a negative rule set, and rule set evaluation is optimized to efficiently identify the events that meet the specified criteria for each process. The positive rule set identifies the rules for objects to be included for Streams processing. The negative rule set lists the rules for objects to be discarded from Streams processing. Each Streams process evaluates the negative rule set first, discarding any message that satisfies a rule in this negative rule set. The process then evaluates rules using the positive rule set. Only messages that evaluate to TRUE as a result of a rule in the positive rule set are passed on to the process.

Horizontal and Vertical Subsetting

Each of the Oracle Streams elements supports subsetting of objects, so a destination need only subscribe to a subset of the data at the source. Consider a typical distributed scenario involving headquarter databases and branch office databases. The branch office database only needs the data relevant for that branch. A branch office would therefore only subscribe to events affecting that branch. Other branches would receive different subsets.

Oracle Streams supports migration of data from one subset to another. Subscriptions are content-based, and should that content change, the subscriptions may change. This may require data to be deleted from one subset database, and inserted into another. Streams automatically manages the changes and row migration issues to ensure that the data applied to the replica reflects the subset criteria. By configuring subset rules for propagation to a particular destination, Insert and Delete LCRs that do not match the subset criteria are not sent to the replica site. Update LCRs are examined to determine if the change affects the replica and, if so, converts the update into the appropriate command (insert or delete) to apply the change correctly. The following example shows how to configure the rules for an apply site to maintain data only for the customers in California (CA). The table Customers is in the HR schema and this subset of data is replicated to the MYREP.WORLD database into the Streams queue:

```
DBMS_STREAMS_ADM.ADD_SUBSET_PROPAGATION_RULES(
   table_name              => 'HR.CUSTOMER',
   dml_condition           => 'STATE=''CA''',
   streams_type            => 'PROPAGATION',
   queue_name              =>'strmadmin.streams_queue',
   destination_queue_name =>
                'strmadmin.streams_queue@MYREP.WORLD'
   source_database         => 'MYDB.WORLD'
 );
```

In this example, the CUSTOMER table contains a column STATE, and each replica site is limited to customers for a particular state. In this instance, state = CA. If an LCR with an insert or delete for a customer has the state column equal to CA, the LCR is sent to the target site. If the insert or delete is for any other state, the LCR is not sent. However, suppose an update to the customer table occurs showing that customer Joe has moved from California to New York. Streams will automatically remove the customer record for Joe, so that Joe will no longer appear in the replica as a California customer. If the update to the customer table shows that customer Jim moved into California from Pennsylvania, then the customer record for Jim will be automatically inserted into the database.

Vertical subsetting of data can be implemented using transformations, as described in the next section.

Transformations

A transformation is a change in the form of an object being captured, propagated or applied, or a change in the data it holds. Oracle Streams supports a variety of declarative transformations, such as renaming a table, schema or column, or adding a column to or removing a column from a table at a particular site. Oracle Streams also supports user-supplied transformations. These custom transformations are represented by PL/SQL functions that take the source data type as input and return an object of the target data type.

When either a declarative or custom transformation is used, every event for the specified object is manipulated using the transformation. Custom transformations can be performed as events are placed into or removed from the staging area. Declarative transformations are performed only during capture. A transformation during capture modifies the message before inserting it into the staging area. For example, a company may want to replicate a table that contains a column with confidential information that should not exist at any other site. Using a transformation during capture, the column can be eliminated from the event or LCR. As a result, no other site will see this restricted information. Custom transformations can also be configured for propagation, which may be useful for formatting the message before it is sent to subsequent remote sites. Finally, a custom transformation can be specified as the event is consumed with the apply process, which can be useful for formatting a message in a manner appropriate for a specific destination.

Heterogeneous Environments

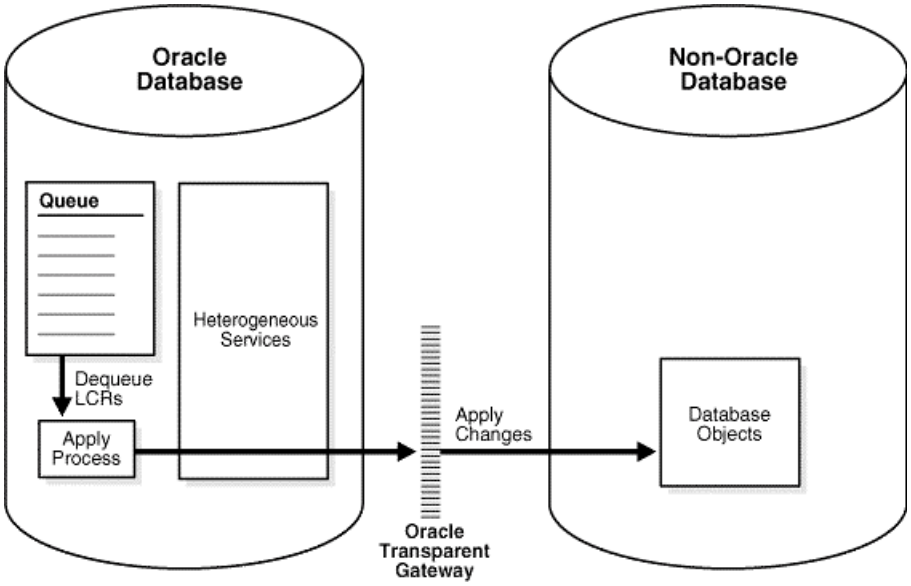
Oracle Streams is an open information sharing solution, supporting heterogeneous replication between Oracle and non-Oracle systems. Using a transparent gateway, changes initiated at Oracle databases can be applied to non-Oracle platforms. Only DML changes can be replicated to non-Oracle platforms.

In order to meet the needs of integrating with legacy applications, the messaging gateway is provided to automatically propagate messages between Oracle queues and Websphere MQ and Tibco queues.

Oracle to Non-Oracle Capture/Apply

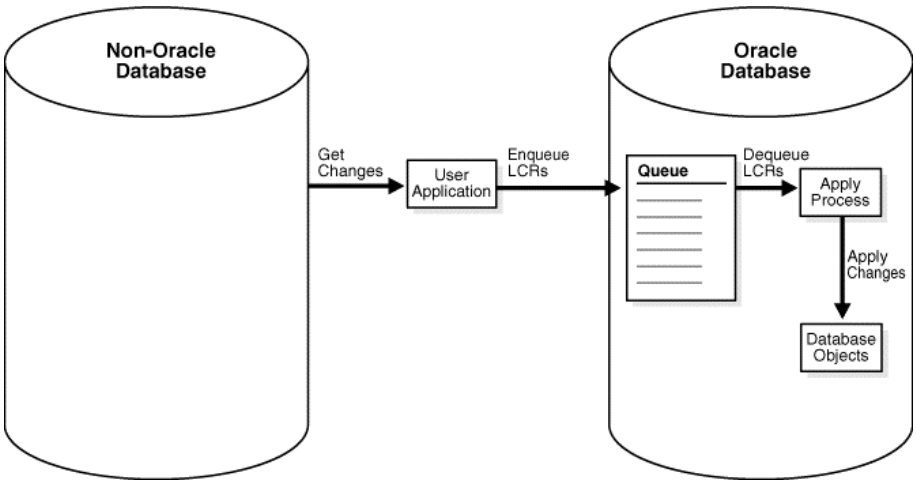
To implement capture and apply of DML changes from an Oracle source to a non-Oracle destination, an Oracle system functions as a proxy and executes the apply engine that would normally be done at an Oracle destination site. The Oracle system then communicates with the non-Oracle system via a transparent gateway. The changes are dequeued in an Oracle database itself and a local Oracle apply process applies the changes to a non-Oracle system across a network connection through a transparent gateway specific to the non-Oracle database.

The following figure illustrates how heterogeneous propagation of DML statements work



Non-Oracle to Oracle Capture/Apply

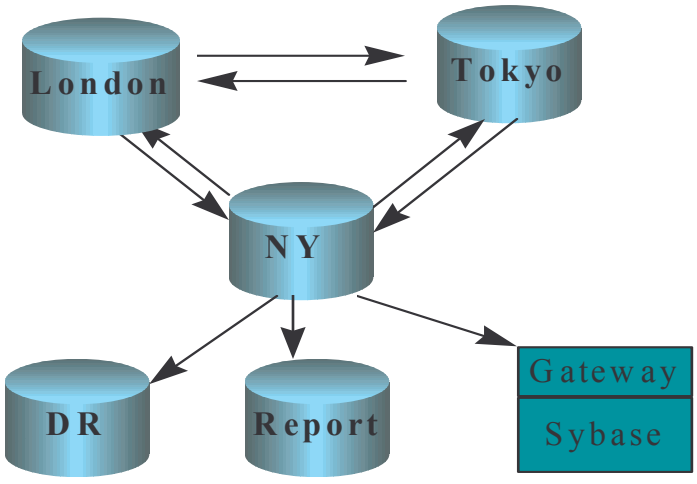
Users who want to propagate changes from a non-Oracle database to an Oracle database must write an application to capture the changes made to the non-Oracle database. The application can capture the changes by reading from transaction logs or by using triggers. The application is then responsible for assembling and ordering these changes into transactions, converting them into an LCR format and publishing them into the target Oracle database staging area.



EXAMPLE STREAMS CONFIGURATION

Oracle Streams is designed for maximum flexibility to satisfy customer information sharing requirements in a variety of markets. For example, customers can use Oracle Streams replication to create n-way master or directed network configurations, to implement data provisioning solutions, or to provide high availability during a platform or application migration. Of course, all customers can utilize the full power of Oracle Streams, and create configurations that seemingly span multiple markets, enabling new classes of applications. In addition, all deployments and their associated meta-data are compatible. For example, a replication installation can easily be extended to load a data warehouse or enable bi-directional replication—a complete reconfiguration is not required.

The flexibility of Oracle Streams is demonstrated in the following example. A corporate IT organization is charged with maintaining data availability around the clock for a critical global application. Their strategy is to maintain three geographically distinct databases with all the requisite data for this high profile application. Additional requirements include a reporting database containing the most current information for the analysts in the company headquarters office in New York to perform ad-hoc querying as well as a disaster recovery database separately maintained from their New York office. A final requirement is to share data with existing applications that are hosted on a Sybase database.



Oracle Streams Usage Example

Oracle Streams is used to replicate data in an N-way configuration consisting of three regional sites: New York, London, and Tokyo. At each of these sites, Streams log-based capture will capture any changes that occur for subscribed tables in each local region, and will stage them locally in the queue. All changes captured in each region will be then forwarded to each of the other region's databases with the goal

that all changes made at each site will be reflected at every other site, providing complete data for the subscribed objects throughout the world.

Since the updates will be automatically applied when received at each regional database, the Oracle Streams default apply engine is used to apply the changes. As updates are applied, Oracle Streams checks for conflicts, and resolves any conflicts that are detected. Streams can also be used to exchange data for particular tables with non-Oracle databases. Utilizing the Oracle Database Gateway for Sybase, the Streams apply engine will apply the changes to a Sybase database using the same mechanisms as it does for Oracle databases.

The databases for reporting and disaster recovery are hosted from the New York database site. The reporting database is a fully functional Oracle database that has a read-only copy of the relevant application tables. The reporting site will not be configured to capture changes on these application tables. Streams will impose no restrictions on the configuration or usage of this reporting database.

Application/Platform Migration Example

Because the three sites in this example are participating in an N-way replication environment, it is possible to perform an application or platform migration at any or all of these sites without requiring any downtime. As each site is upgraded, users can be re-routed to the two remaining sites. When the migration at a site is complete, it can be made available again for updates. Changes made at the other locations will automatically be propagated to the site once the migration is complete.

This feature is useful even for locations not currently participating in a Streams replication environment. You can easily perform an application or platform upgrade at a site without incurring any downtime by first creating a temporary replica of the site. Oracle Streams one-step API's, described earlier in this document, greatly simplify this process. The following general steps outline the operations that must be completed in order to perform a database maintenance operation while remaining online:

1. Create a new, empty database.
2. Use Oracle Streams one-step procedures, pre_instantiation and post_instantiation, to configure a replication environment, where this new database is the destination database, and the original database is the source database. After instantiating (populating) the destination database (with Export/Import or RMAN), Oracle Streams will automatically ensure that DML and DDL changes occurring at the source database are ultimately applied at the destination database.
3. Perform the desired maintenance operations on the destination database. During this time, the original database remains available online with Streams configured, but not started.

4. After completing the maintenance operations, use Oracle Streams to apply changes made at the source database to the destination database.
5. Once these changes have been successfully applied at the destination, take the source database offline and make the destination database available for applications and users.

CONCLUSION

Oracle Streams simplifies sharing data between databases and database clusters, with its revolutionary concept of unifying message queuing and data replication capabilities. With Oracle Streams replication, the basic elements of capture, staging, and consumption in combination with user-defined transformations and heterogeneous interoperation produce more sophisticated configurations than ever before possible. This resilient technology provides the configuration flexibility required to handle the real world situations encountered in modern global corporations. Oracle Streams replication technology is the premier feature for sharing data in complex distributed database environments.

Oracle Database 11g: Oracle Streams Replication
July 2007
Contributing Authors: Patricia McElroy, Maria Pratt

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com