Oracle Streams
Oracle FLEXCUBE Universal Banking
Release 12.2.0.0.0
[May] [2016]

Table of Contents

1. Introduction

1.1 Scope

This document provides an overview of Oracle streams and also outlines the various test scenarios that were used and the possible areas this solution could be used in the product.

1.2 Introduction to Oracle Streams

Oracle Streams is an information sharing/data replication methodology that captures database changes at a source database, stages and propagates the changes internally using Oracle Advanced Queues to one or more destination databases and then applies the changes at the destination database.

Oracle Streams Process supports capturing and propagation of DML and DDL changes for a specific table, set of tables, schema or database level.

Oracle Streams can be implemented in any of the following cases:-

- Source and Destination within the same Oracle database
- Between two Oracle databases (one source and one destination)
- Among multiple Oracle databases (one source and many destination)
- Among multiple Oracle databases (many source and many destination)
- Between an Oracle database and a non-Oracle database

First 3 cases have been attempted in this feasibility analysis with success. Last 2 cases have not yet been tested.

1.3 Advantages

Oracle streams can be used in the following cases:-

- Data Replication for reporting or Data Extracts. Refer the FLEXCUBE specific use cases detailed in the later sections
- Database Availability during Upgrade and Maintenance

2. Requirement /Problem Statement

The requirement is to display the capability of Oracle Streams as an effective data replication strategy.

3. Prerequisites

3.1 Software Required

Oracle Database server 11g (11.1.0.6 or above)

Download & install the Oracle Database server 11g from the below link

http://www.oracle.com/technology/software/products/database/index.html

3.2 Database Settings Required

Following settings are required to be done on the database servers:-

- Streams replication between two different Oracle databases require their global names to be different.
- Streams replication requires the source database to be in Archive log mode for uni-directional replication and source and destination database in case of bi-directional replication.
- Two way database link between the source and destination databases.
- A dedicated oracle schema/user needs to be created and assigned as a Streams administrator. DBA rights would be required during the creation and modification of the streams components.
- A dedicated tablespace is required for Streams administrator.
- A dedicated tablespace is required for the log miner data dictionary.
- Global_names parameter should be set to true on both source and destination databases.
- Job_queue_process parameter should be >=2
- Streams internally use Streams Pool in the SGA for Implicit capturing. It is recommended that the database parameter streams_pool_size is set according to the expected amount of data change or allow it to be managed by Automatic Memory Management to control the allocation.
- Streams do not support tables which have segment compression.
- Changes created on a table with 'nologging' option would not be captured. It is recommended to enable force logging for critical tablespaces.
- Capture background process is not started when the database is in restricted mode.
- If global names or DBID is changed in the source database then the capture process would have to be dropped and recreated.

3.3 Licensing Requirements

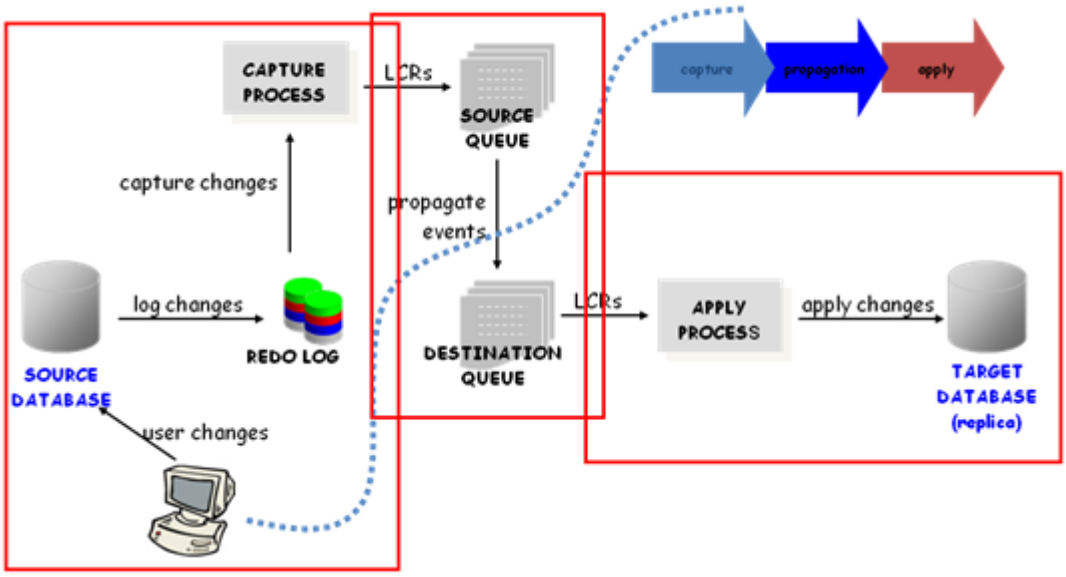
Oracle Streams capture process is included in the Oracle Enterprise Edition and as of now it is not licensed separately. Refer to the latest licensing guide for any changes.

4. Oracle Streams Description

4.1 Oracle Streams Architecture

Oracle Streams comprises of the following three basic components:-

- Capture Process
- Staging and propagation
- Consumption or Apply process



4.1.1 Capture Process

The capture process is an Oracle background process that is designed to capture changes at table, schemas and database level. All changes that are recorded into redo log of a database, capture process capture those changes from redo log and format the captured changes into message called Logical change record (LCR).Oracle

Capture process can be configured to run on the production database or in a standby database.

Streams provide two ways to capture database changes:-

- Implicit Capture
- Explicit Capture

4.1.1.1 Implicit Capture

With Implicit capture mechanism of Oracle Streams, changes made to a database can be automatically captured and replicated to one or more databases.

Following are two types of implicit capture:-

- Capture Process
- Synchronous capture

Capture Process

The capture process is used to capture changes at table or schema or database level. All changes that are recorded into redo log of a database. A capture process retrieves change data from the redo log, either by mining the online redo log or if necessary, by mining archived log files.

A capture process captures changes from the redo log and formats each captured change into a message called a logical change record (LCR) and the capture process can intelligently filter LCRs based upon defined rules places it in a staging area for further processing. Rule is a configuration that enables capturing of changes to DDL or DML or both.

Synchronous Capture

Synchronous capture is an 11g feature in Oracle Streams that captures data manipulation language (DML) changes made to tables in the best optimized manner. Following are the properties of Synchronous capture:-

- This feature captures DML changes only and does not capture DDL changes.
- This feature is enabled only for table level capture
- Synchronous capture uses an optimized mechanism to capture DML changes to a set of specified tables immediately after the changes are committed.
- Synchronous capture is recommended when the number of tables and the amount of data change expected is minimal.

When a DML change it 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 into a specific message format called a row logical change record (row LCR). After capturing a row LCR, synchronous capture enqueues a message containing the row LCR into a queue.

Row LCRs created by synchronous capture always contain values for all the columns in a row, even if some of the columns where not modified by the change. This is not available in pre 11g releases. Feasibility testing has been conducted using Implicit & synchronous capture process.

Synchronous capture process does not support capturing of changes for the tables having following data types

- LONG

- LONG RAW
- CLOB
- NCLOB
- BLOB
- BFILE
- ROWID
- User-defined types (including object types, REFs, varrays, and nested tables
- Oracle-supplied types (including Any types, XML types, spatial types, and media types)

4.1.1.2 Explicit Capture

With Explicit Capture, applications can generate messages and enqueue them on their own. These messages can be formatted as LCRs, or they can be formatted into different types of messages for consumption by other applications. Messages can also be enqueued explicitly by an apply process or by an apply handler for an apply process.

This option has not been used for the feasibility study.

4.1.1.3 Capture Methods and Fitment

Capture Type	Capture Mechanism	Recommended Use Case
Implicit Capture	Mining of Redo Log	You want to capture changes to many tables You want to capture changes to schemas or an entire database You want to capture DDL changes You want to capture changes at a downstream database(Standby)

Implicit capture with Synchronous	Internal Mechanism	You want to capture DML changes to a small number of tables, typically maintenance & static data tables – not contract tables. You want to capture DML changes immediately after they occur.
Explicit capture by application	Manual Message Creation and Enqueue	You want to capture user messages that will be consumed by applications. You want to capture LCRs in a heterogeneous replication environment. You want to construct LCRs by using an application instead of by using a capture process or a synchronous capture

Different objects within the same DB can have different types of capture mechanisms.

4.1.1.4 Capture Process Rules

A capture process either captures or discards changes based on rules that you define. Each rule specifies the database objects and types of changes for which the rule evaluates to TRUE. You can place these rules in a positive rule set or negative rule set for the capture process.

If a rule evaluates to TRUE for a change, and the rule is in the positive rule set for a capture process, then the capture process captures the change. If a rule evaluates to TRUE for a change, and the rule is in the negative rule set for a capture process, then the capture process discards the change. If a capture process has both a positive and a negative rule set, then the negative rule set is always evaluated first.

You can specify capture process rules at the following levels:-

- A table rule captures or discards either row changes resulting from DML changes or DDL changes to a particular table. Subset rules are table rules that include a subset of the row changes to a particular table.
- A schema rule captures or discards either row changes resulting from DML changes or DDL changes to the database objects in a particular schema.
- A global rule captures or discards either all row changes resulting from DML changes or all DDL changes in the database.

Note:

The capture process does not capture certain types of changes and changes to certain data types in table columns. Also, a capture process never captures changes in the SYS, SYSTEM, or CTXSYS schemas.

4.1.1.5 Unsupported Data types

The capture process does not support capturing of changes to following data types.

- SecureFile CLOB, NCLOB, and BLOB
- BFILE
- ROWID
- User-defined types (including object types, REFs, varrays, and nested tables)
- XMLType stored object relationally or as binary XML
- The following Oracle-supplied types: Any types, URI types, spatial types, and media types

4.1.1.6 SCN Instantiation

In Oracle Streams, the following general steps instantiate a database object:

1. Prepare the object for instantiation at the source database.
2. If a copy of the object does not exist at the destination database, then create an object physically at the destination database based on an object at the source database. You can use export/import, transportable tablespaces, or RMAN to copy database objects for instantiation. If the database objects already exist at the destination database, then this step is not necessary.
3. Set the instantiation SCN for the database object at the destination database. An instantiation system change number (SCN) instructs an apply process at the destination database to apply only changes that committed at the source database after the specified SCN.
4. Set the instantiation SCN for the database object at the destination database. An instantiation SCN instructs an apply process at the destination database to apply only changes that committed at the source database after the specified SCN.

Moving LogMiner tables from SYSTEM tablespace

By default, all LogMiner tables are created in the SYSTEM tablespace.

Create separate LogMiner tablespace for LogMiner tables.

CREATE TABLESPACE LOGMNRTS DATAFILE 'logmnrts.dbf' SIZE 100M AUTOEXTEND ON MAXSIZE UNLIMITED;

Begin

Dbms_logmnr_d.set_tablespace (‘LOGMNRTS’);

End;

/

4.1.1.7 Supplemental Logging

Oracle database redo log files contain redo information needed for instance and media recovery. Oracle streams need additional information to be logged into the redo log files. The process of logging this additional information into the redo files is called Supplemental Logging.

There are two levels of Supplemental Logging

- Database Level Supplemental Logging

  Alter database add supplemental log data;

- Table-level Supplemental Logging.

Creates individual log groups for each table.

ALTER TABLE t1 ADD SUPPLEMENTAL LOG DATA (primary key/unique key/all) COLUMNS;

4.1.2 Staging and Propagation

Staging

Once captured, events are placed in a staging area. The staging area is a queue (AQ) that provides a service to store and manage captured events.

Propagation

It is a process to propagate the message from one queue to one or more other queues. The queue from where messages are propagated can reside in the same database or in Different databases. The queue from where the messages are propagated is called the source queue, and the queue where the messages are received is called the destination queue.

If the source and destination schemas are hosted within the same database then the destination queue becomes an optional component.

4.1.3 Consumption

An apply engine dequeues the changes from the destination queue which needs to be applied. The database where the messages are consumed is called the destination database. The default apply engine applies DML changes and DDL changes represented by implicitly or explicitly captured LCRs. Apply process detects conflict when applying the LCR.

A conflict is a mismatch between old and the new values in an LCR and the expected data in the table. Oracle streams provides certain pre built conflict handlers to resolve the conflict and also provides an option to define our own conflict handler.

4.2 Conflicts in Oracle Streams

A conflict is a mismatch between the old values in an LCR and the expected data in a table. In streams environment, DML conflict happens when two transactions originating at different databases update the same row at nearly the same time

Types of conflict in stream environment

Update Conflict

Update conflicts occurs when two transactions originating from different databases update the same row at nearly the same time.

Uniqueness Conflict

A uniqueness conflict when two transactions originate from two different databases, each inserting a row into a table with the same primary key value.

Delete Conflict

A delete conflict occurs when two transactions originate at different databases, with one transaction deleting a row and another transaction updating or deleting the same row

Foreign Key Conflict

A foreign key conflict occurs when the apply process applies a row LCR containing a change to a row that violates a foreign key constraint.

4.2.1 Conflict Resolution

There are two ways to handle the conflict resolutions:

- Prebuilt Update Conflict Handler
- Custom Conflict handler

4.2.1.1 Prebuilt Update Conflict Handler

It is a conflict handler provided by oracle stream to resolve update conflict only. SET_UPDATE_CONFLICT_HANDLER procedures in the DBMS_APPLY_ADM package need to be define for particular table to use prebuilt conflict handler.

Types of prebuilt conflict handlers

Overwrite

The Overwrite handler replaces the current value at the destination database with the new value in the LCR from the source database. if there is conflict

Discard

The Discard handler ignores the values in the LCR from the source database and retains the value at the destination database. If there is conflict

Maximum

When a conflict occurs, the Maximum conflict handler compares the new value in the LCR from the source database with the current value in the destination database by using the resolution column. If the new value of the resolution column in the LCR in source database is greater than the current value of the column at the destination database, then the apply process applies the source LCR to resolves the conflict.

Minimum

When a conflict occurs, the Minimum conflict handler compares the new value in the LCR from the source database with the current value in the destination database for a designated resolution column. If the new value of the resolution column in the LCR in source database is less than the current value of the column at the destination database, then the apply process applies the source LCR to resolves the conflict.

4.2.1.2 Custom Conflict handler

The conflict handler is defined by user to handle the conflict resolution by creating the pl/sql procedure. The SET_DML_HANDLER procedure in the DBMS_APPLY_ADM package is used to define one or more custom conflict handlers for a particular table.

5. Oracle Streams – Features with FLEXCUBE Use Cases

In this section, various features of Oracle Streams which can be a potential use case for FLEXCUBE are discussed in brief.

5.1 FLEXCUBE Use Case for Uni-directional table level replication

Unidirectional Data Replication for a specific table or a set of tables to a different instance can act as a tool for:-

Data Availability

This can be used when a copy of a specific piece of data is required outside the main database – for e.g., a copy of Customer account balances for any eventual offline access.

Reporting

This can be used to have a copy of specific tables for catering to long running reports and extracts.

Data Synchronization

This would typically define the Head Office to country data propagation used case. In such scenarios, we currently use DB links which create performance and availability issues on the HO instance. Both limitations can be mitigated using Oracle Streams.

5.2 FLEXCUBE Use Case for Bi-Directional table level replication

If a use case for this scenario is envisaged, complicated conflict resolution handling would be required using custom conflict handlers. This scenario should thus be avoided.

5.3 FLEXCUBE Use Case for Uni-Directional Schema Level replication

Unidirectional Schema Level replication can act as a tool for:-

Reporting and Data Extraction

The entire FLEXCUBE database can be replicated using streams to a Standard Edition database hosted on a commodity platform such as low cost blades / Linux server. This copy can be used to run reports or data extraction routines. Current solution approach to this use case is based on either Oracle Dataguard or Storage Level Replication which mandates Oracle Enterprise Edition or same class and storage as the production. Significant cost savings can be attained with the new approach.

Procedural Replication

For better source control, program units can be compiled in one schema and propagated to others. This approach can help in maintaining consistency in centralized branch deployments where there are multiple branch schemas and one host schema on the same database (FCUBS) or across multiple LOB's (FCIS). Similar approach can also be adopted for multi-country deployments in single data center where the source base is common across countries.

5.4 FLEXCUBE Use Case for Bi-Directional Schema level replication

If a use case for this scenario is envisaged, complicated conflict resolution handling would be required using custom conflict handlers. This scenario should thus be avoided.

5.5 FLEXCUBE Use Case for Uni-directional DB level replication

Uni-directional DB level replication can act as a tool for:-

Database/System Upgrades

During O/S upgrades and Database upgrades, this configuration can be used to serve as an alternate transaction processing system for the duration of the upgrade. On completion of the upgrade process, the streams capture data is propagated to the upgraded production system.

5.6 FLEXCUBE Use Case for Bi-Directional DB level replication

If a use case for this scenario is envisaged, complicated conflict resolution handling would be required using custom conflict handlers. This scenario should thus be avoided.

Extending any of the above use cases a new FLEXCUBE screen can be envisaged to be designed which can provide a user interface for selecting subset of tables and generating a customizable script for the streams setup.

Oracle Streams
[May] [2016]
Version 12.2.0.0.0

Oracle Financial Services Software Limited
Oracle Park
Off Western Express Highway
Goregaon (East)
Mumbai, Maharashtra 400 063
India

Worldwide Inquiries:
Phone: +91 22 6718 3000
Fax:+91 22 6718 3001
www.oracle.com/financialservices/