Universidad de las Ciencias Informáticas

Introducción de la tecnología Streams del SGBDR Oracle 11g, para garantizar la explotación del sistema de Circulados Nacionales.

Clasificación: Investigación y Desarrollo.

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autores:

José Luis León López.

Fernando Mesa Rodríguez.

Tutor:

Alex David Saad Blanco.

Cuidad de la Habana
Junio 2009

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

José Luis León López	**Fernando Mesa Rodríguez**	**Alex David Saad Blanco**
_____	_____	_____
Firma del Autor	Firma del autor	Firma del Tutor

DATOS DE CONTACTO

Tutor:

Ingeniero en Sistemas desde 1997 hasta 2004 en el Órgano de Informática Comunicaciones y Cifras de la provincia Santiago de Cuba, especialidad que inicia siendo técnico de nivel medio en programación. Cursa diferentes materias de la especialidad, fundamentalmente en las áreas de sistemas operativos UNIX, sistema de gestión de base de datos Oracle y protocolo TCP/IP para los servicios de red. En el año 2003 culmina sus estudios superiores en la Universidad de Oriente diplomándose como Licenciado en Ciencias de la Computación.

A partir de Julio de 2004 es seleccionado por la Dirección de Informática Comunicaciones y Cifras de la institución para formar parte de proyectos a nivel nacional, participando en el desarrollo de software relacionado con técnicas de Minería de Texto y la asimilación e implementación de tecnologías de almacenamiento masivo (SAN) y alta disponibilidad (RAC, DATAGUARD). Durante los 2 últimos años se ha mantenido vinculado a tareas de dirección, investigación y aplicación de las nuevas posibilidades del Sistema Gestor de Bases de datos Oracle 11g, y asimilación e implementación de la tecnología Blade Server.

DEDIATORIA

A:

Mis padres y mi hermana.

José Luis León López.

A:

Mis abuelos y a mis padres.

Fernando Mesa Rodríguez

AGRADECIMIENTOS

A las personas más importantes de mi vida, **mis padres** por ser los mejores padres del mundo y estar ahí siempre que los necesité ,por darme su apoyo y sufrir conmigo cada derrota y sobre todo por darme su confianza. Sin ustedes nada de esto hubiese sido posible.

A **mi hermana** del alma, a la cual adoro por ser mi ejemplo de persona a seguir, por su gran ayuda en todo momento y por hacerme sentir que soy dichoso de tenerte como hermana. Tu eres parte de este logro POPA.

A **Alex David Saad Blanco** por su ayuda incondicional en todo momento a pesar de las dificultades y por estar ahí siempre que lo necesite en el desarrollo de la tesis como un miembro más.

A mi novia **Dayana** por estar a mi lado siempre y compartir conmigo los mejores y peores momentos sin condiciones, por soportar mis malos ratos y por hacerme feliz.

A mi mejor amigo y compañero de tesis **Fernan** por ser como un hermano en las buenas y las malas y por haber compartido estos 5 años “tú eres sangre de mi sangre socio”.

A mi abuela **Yolanda** que tanto quiero por su cariño y su preocupación por mí.

A mi familia en general por apoyarme tanto siempre.

A **Deivis Leyva** por su ayuda y apoyo durante toda la tesis.

A **Wendy** por su ayuda siempre que la necesité durante toda la carrera sin objeciones.

A mis compañeros y amigos de la universidad.

A todo el que de una forma u otra me apoyo durante toda la carrera .Gracias.

José Luis León López.

AGRADECIMIENTOS

A mi abuela Alicia que no se rindió nunca, procurando que mis estudios, de niño y adolecente, fueran lo mejor posible. Hoy me doy cuenta de lo importante que fue ese apoyo para mí.

A mi mamá por sus noches repasándome Historia o Matemática, buscando estrategias para que me aprendiera todo el contenido o para que entendiera un ejercicio.

A mi papá por aconsejarme y guiarme en los momentos difíciles.

A mi abuelo por sus consejos.

A Wendy, mi novia, por apoyarme y ayudarme durante toda la carrera, incluso en la realización de este trabajo.

A mis suegros que son los mejores suegros que se puede tener. Además, que pude contar siempre con ellos para lo que fuera.

A nuestro tutor, que nos ha ayudado enormemente en la realización de este trabajo. Sacrificando las horas de descanso.

A “El Jose” mi compañero de tesis que siempre hemos estado juntos en las buenas y las malas.

Fernando Mesa Rodríguez

RESUMEN

El presente documento contiene un estudio realizado de la tecnología Streams del Sistema Gestor de Bases de Datos Relacional Oracle en su versión 11g, y las ventajas que esta proporciona. Este estudio se hizo con el objetivo de asimilar dicha tecnología y utilizarla como mecanismo de replicación en entornos de bases de datos distribuidas. En este trabajo presentamos un análisis del estado actual de la replicación de datos y la implementación de Streams en las bases de datos del Sistema de Circulados Nacionales así como una propuesta de configuración que garantice la optimización de este proceso. También encontrará cómo se implementó el polígono de prueba donde se simuló la replica de datos en el Sistema de Circulados Nacionales y permitió configurar los Streams de forma óptima. De la misma manera se hace referencia a procedimientos importantes para el monitoreo y la administración de dicha tecnología.

INDICE

INTRODUCCION

Cada día el hombre actual es más dependiente de las computadoras, de tal manera que estas se han convertido en la herramienta principal para la organización, simplificación, agilización, automatización y desarrollo de la mayoría de los procesos del mundo de hoy, llegando a cambiar la forma de pensar del ser humano.

Los Sistemas Gestores de Bases de Datos se crearon debido a la necesidad de almacenar grandes volúmenes de información de forma organizada, accesible y segura. Una de las posibilidades que estos brindan, es la replicación de datos, que no es más que el proceso de copiar y mantener objetos en un entorno distribuido, con el objetivo de garantizar alta disponibilidad y desempeño**,** haciendo del acceso local a datos compartidos, una opción que reduce costos de conectividad y distribuye la carga de trabajo.

Oracle es un Sistema Gestor de Bases de Datos Relacional (SGBD-R) que implementa soluciones eficientes y efectivas para la manipulación de grandes volúmenes de información. En su última versión, Oracle 11g, que fue creada sobre la base de más de 30 años de experiencia, incluye una tecnología ya existente para la replicación de datos, llamada Streams pero siendo, esta vez, mucho más flexible en la configuración y administración de flujos de datos que en versiones anteriores.

En la batalla diaria por erradicar el delito y preservar el orden interior del país, el MININT desarrolla e implementa softwares y tecnologías que son fundamentales para el cumplimiento de esta tarea. Debido a la necesidad de tener control, a nivel nacional, sobre las personas circuladas, este ministerio desarrolló un software para esta importante tarea, el cual tiene como nombre Circulados Nacionales, entre sus funcionalidades se encuentra la replicación de información a nivel nacional, que por la importancia de los datos que se comparten, es necesario utilizar tecnologías que reduzcan la latencia en el proceso de propagación de datos y garanticen una alta disponibilidad y seguridad.

Actualmente, el sistema de Circulados Nacionales soporta sus mecanismos de replicación de datos sobre las posibilidades que ofrece la replicación tradicional de Oracle desde sus versiones iniciales. Si bien esta alternativa ha venido resolviendo la necesidad de compartir esta información, no es la forma más óptima de realizar este proceso. La replicación avanzada de Oracle, por estar implementada sobre mecanismos transaccionales que utilizan SQL y PL-SQL, le introduce a esta tecnología componentes que retardan el proceso de propagación y dificultan la implementación y administración de la misma.

Teniendo en cuenta la problemática descrita anteriormente, nos planteamos la solución de la misma y concebimos la formulación del **problema** de la siguiente manera:

¿Cómo optimizar la replicación de datos entre las bases de datos nacionales que soportan la explotación del sistema de Circulados Nacionales utilizando la tecnología Streams de Oracle 11g?

El **objeto de estudio** está constituido por:
El proceso de Replicación de Datos de las bases de datos del sistema de Circulados Nacionales.

La investigación persigue como **objetivo general**:
Obtener una propuesta de configuración e implementación para el proceso de Replicación de Datos, entre las bases de datos nacionales que soportan el sistema de Circulados Nacionales, usando la tecnología Streams de Oracle 11g

El **campo de acción** está centrado en:
Proceso de modelado de configuración de la tecnología Streams de Oracle 11g para el proceso de Replicación de Datos entre las bases de datos del sistema de Circulados Nacionales.

Se plantea la siguiente **Idea a defender:**
El uso de una implementación adecuada de la tecnología Streams de Oracle 11g, permitirá elevar el desempeño del proceso de Replicación de datos perteneciente al sistema de Circulados Nacionales, así como flexibilizar su configuración y administración.

Para dar cumplimiento a los objetivos de nuestra investigación se definieron como **tareas investigativas:**

1. Asimilar la tecnología Streams en Oracle 11g.
2. Asimilar el proceso actual de replicación de datos en el sistema de Circulados Nacionales.
3. Caracterizar la tecnología Streams y definir las ventajas y perspectiva futura, con respecto al proceso actual de replicación de datos en el sistema de Circulados Nacionales.
4. Describir métodos y procedimientos para el despliegue de la tecnología que se asimila.
5. Implementar la tecnología asimilada para optimizar el proceso actual de replicación de datos en un polígono de prueba que simule el sistema de Circulados Nacionales.

CAPITULO1: FUNDAMENTACIÓN TEÓRICA

Introducción

En este capítulo se hace mención a un grupo de conceptos que son importantes para el desarrollo de nuestra investigación y que serán referenciados en el resto del documento. Se realiza un breve enfoque a la replicación de datos entre Bases de Datos y al Sistema Gestor de Bases de Datos Oracle, definiendo con precisión los conceptos generales presentes en la Replicación de Datos, utilizando una nueva tecnología manejada por Oracle llamada Streams.

1.1 Bases de Datos

Una base de datos o banco de datos es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso. En la actualidad, debido al desarrollo tecnológico existente en la informática y la electrónica, la mayor parte de los datos se encuentran almacenados en formato digital (electrónico), lo cual ofrece un amplio rango de soluciones al problema de almacenar datos. Las Bases de Datos digitales permiten almacenar grandes volúmenes de datos de forma organizada.
Desde el punto de vista informático, una base de datos es un sistema formado por un conjunto de datos almacenados en discos que permite el acceso directo a ellos y un grupo de programas para su manipulación.[1]

Definición de bases de datos

Se define una Base de Datos como un conjunto de datos almacenados en serie y relacionados entre sí, los cuales son recolectados y explotados por los sistemas de información de una empresa o negocio en particular.

Características

Entre las principales características de una Base de Datos se encuentran:

1. **Independencia lógica y física de los datos:** Se refiere a la capacidad de modificar una definición de esquema en un nivel de la arquitectura, sin que esta modificación afecte al nivel inmediatamente superior. Para ello un registro externo en un esquema externo no tiene por qué ser igual a su registro correspondiente en el esquema conceptual.
2. **Redundancia mínima:** Se trata de usar la base de datos como repositorio común de datos para distintas aplicaciones.
3. **Acceso concurrente por parte de múltiples usuarios:** Control de concurrencia mediante técnicas de bloqueo o cerrado de datos accedidos.
4. **Distribución espacial de los datos:** La independencia lógica y física facilita la posibilidad de sistemas de bases de datos distribuidas. Los datos pueden encontrarse en otra habitación, otro edificio e incluso otro país. El usuario no tiene por qué preocuparse de la localización espacial de los datos a los que accede.
5. **Integridad de los datos:** Se refiere a las medidas de seguridad que impiden que se introduzcan datos erróneos. Esto puede suceder tanto por motivos físicos (defectos de hardware, actualización incompleta debido a causas externas), como de operación (introducción de datos incoherentes).
6. **Consultas complejas optimizadas:** La optimización de consultas permite la rápida ejecución de las mismas.
7. **Seguridad de acceso y auditoría:** Se refiere al derecho de acceso a los datos contenidos en la base de datos por parte de personas y organismos. El sistema de auditoría mantiene el control de acceso a la base de datos, con el objetivo de saber qué o quién realizó una determinada modificación de los datos y en que momento.
8. **Respaldo y recuperación:** Se refiere a la capacidad de un sistema de bases de datos de recuperar su estado en un momento previo a la pérdida de datos.
9. **Acceso a través de lenguajes de programación estándar:** Se refiere a la posibilidad ya mencionada de acceder a los datos de una base de datos, mediante lenguajes de programación ajenos al sistema de bases de datos propiamente dicho. [1]

Una base de datos típica conlleva la existencia de tres tipos de usuario con relación a su diseño, desarrollo y uso:

1. El administrador de bases de datos (DBA, por sus siglas en inglés): Diseña y mantiene la base de datos.
2. El desarrollador de aplicaciones (programador): Implementa las transacciones e interfaces.
3. Los usuarios finales: Consultan y editan los datos de la base de datos mediante un lenguaje de consulta de alto nivel.

En general, podemos decir que el propósito de una base de datos es doble:

1. Responder a consultas sobre los datos que contiene.
2. Ejecutar transacciones.[1]

1.2 Sistema Gestor de Bases de Datos

En 1964, se conciben los primeros Sistema Gestores de Bases de Datos (SGBD), con estos se pretendió dar un viraje a los sistemas de archivos, los cuales se limitaban a la estructuración del almacenamiento físico de los datos. Con los SGBD se logró por medio de actividades integradas verlos físicamente en un solo almacenamiento, pero lógicamente se manipulan a través de esquemas compuestos por estructuras donde se establecen vínculos de integridad, métodos de acceso y organización física sobre los datos, permitiendo así obtener valores agregados de utilización. En 1970 se propuso el Modelo de Datos Relacional, este modelo ha marcado la línea de investigación por muchos años y representa al mundo real mediante tablas relacionadas entre sí por columnas comunes. Viendo la necesidad de mejorar este estándar se desarrollaron los Sistemas Gestores de Bases de Datos Relacionales (SGBDR) cuyas características hacen al sistema mucho más eficiente que los sistemas de manejos de archivos. Un SGBDR es un conjunto de datos relacionados entre sí y un grupo de programas para tener acceso a esos datos, permitiendo concurrencia y recuperación. La persistencia de un SGBDR hace referencia a la conservación de los datos después de la finalización del proceso que los creó y la concurrencia se refiere a la capacidad del sistema para gestionar múltiples usuarios interactuando al mismo tiempo.

Un Sistema de Gestión de Bases de Datos (DBMS, por sus siglas en inglés) es un conjunto de programas que se encargan de manejar la creación y todos los accesos a las bases de datos y está compuesto por:

Un Lenguaje de Definición de Datos (DDL, por sus siglas en inglés)**:** Por medio de este el DBMS identifica las descripciones de los elementos contenidos en los esquemas y almacena la descripción del esquema en el catálogo del DBMS.

Lenguaje de Manipulación de Datos (DML, por sus siglas en inglés)**:** Permite la manipulación de las operaciones de Inserción, Eliminación y Modificación.

Tipos de DML's:

- De alto Nivel o No por procedimientos SQL.
- De bajo Nivel o por procedimientos SQL.

Un Lenguaje de Consulta Estructurado (SQL, por sus siglas en inglés):

Es un lenguaje declarativo de acceso a Base de Datos Relacionales que permite especificar diversos tipos de operaciones con las mismas. Una de sus características es el manejo del Álgebra Lineal y Cálculo Relacional, permitiendo lanzar consultas con el fin de recuperar de forma sencilla información de interés en una Base de Datos, así como realizar cambios sobre las mismas.

Objetivos de los SGBD

Existen distintos objetivos que deben cumplir los SGBD:

- **Abstracción de la información**. Los SGBD ahorran a los usuarios detalles acerca del almacenamiento físico de los datos. Da lo mismo si una base de datos ocupa uno o cientos de archivos, este hecho se hace transparente al usuario. Así se definen varios niveles de abstracción.
- **Independencia**. La independencia de los datos consiste en la capacidad de modificar el esquema (físico o lógico) de una base de datos sin tener que realizar cambios en las aplicaciones que se sirven de ella.

- **Consistencia**. En aquellos casos en los que no se ha logrado eliminar la redundancia, será necesario vigilar que aquella información que aparece repetida se actualice de forma coherente, es decir, que todos los datos repetidos se actualicen de forma simultánea. Por otra parte, la base de datos representa una realidad determinada que tiene algunas condiciones, por ejemplo que los menores de edad no pueden tener licencia de conducir; que el sistema no debería aceptar datos de un conductor menor de edad. En los SGBD existen herramientas que facilitan la programación de este tipo de condiciones.
- **Seguridad**. La información almacenada en una base de datos puede llegar a tener un gran valor. Los SGBD deben garantizar que esta información se encuentre segura frente a usuarios malintencionados, que intenten leer información privilegiada; frente a ataques que deseen manipular o destruir la información; o simplemente ante las torpezas de algún usuario autorizado, pero inexperto. Normalmente, los SGBD disponen de un complejo sistema de permisos a usuarios y grupos de usuarios, que permiten otorgar diversas categorías de permisos.
- **Integridad**. Se trata de adoptar las medidas necesarias para garantizar la validez de los datos almacenados. Es decir, se trata de proteger los datos ante fallos de hardware, datos introducidos por usuarios descuidados, o cualquier otra circunstancia capaz de corromper la información almacenada. Los SGBD proveen mecanismos para garantizar la recuperación de la base de datos hasta un estado consistente (ver Consistencia, más arriba) conocido en forma automática.
- **Respaldo**. Los SGBD deben proporcionar una forma eficiente de realizar copias de respaldo de la información almacenada en ellos, y de restaurar a partir de estas copias los datos que se hayan podido perder.
- **Control de la concurrencia**. En la mayoría de entornos (excepto quizás el doméstico), lo más habitual es que sean muchas las personas que acceden a una base de datos, ya sea para recuperar información, como para almacenarla, también es frecuente que dichos accesos se realicen de forma simultánea. Así pues, un SGBD debe controlar este acceso concurrente a la información, que podría derivar en inconsistencias.
- **Manejo de Transacciones**. Una Transacción es un programa que se ejecuta como una sola operación, esto quiere decir que el estado luego de una ejecución en la que se produce una falla, es el mismo que se obtendría si el programa no se hubiera ejecutado. Los SGBD proveen mecanismos para

programar las modificaciones de los datos de una forma mucho más simple que si no se dispusiera de ellos.

- **Tiempo de respuesta**. Lógicamente, es deseable minimizar el tiempo que el SGBD tarda en darnos la información solicitada y en almacenar los cambios realizados. [2]

Los SGBD más comunes son: Oracle, SqlServer, Informix, Sysbase, MySQL, PostgreSql, Magic, Firebird.

1.3 Flujo de Datos

Se entiende por flujo de datos, al transporte de información de un lugar a otro ya sea en el mismo ordenador o de una PC a otra. Este flujo de datos se denomina también Streams. El concepto de Streams o Flujo de Datos abarca desde leer o escribir datos en un ordenador hasta replicar datos de una PC en otra.

1.4 Replicación de Datos

La replicación de datos consiste en el transporte de datos entre dos o más servidores, permitiendo que ciertos datos de la base de datos estén almacenados en más de un sitio y así aumentar la disponibilidad de los datos y mejorar el rendimiento de las consultas globales. A los tipos básicos de replicación (de instantáneas, transaccional y de mezcla), se le incorporan opciones para ajustarse aún más a los requerimientos del usuario. [3]

1.4.1 Características de la Replicación de Datos

Componentes del modelo de replicación

Para representar los componentes y procesos de una topología de replicación, se utilizan metáforas de la industria de la publicación. El modelo se compone de los siguientes objetos: el publicador, el distribuidor, el suscriptor, la publicación, el artículo y la suscripción; así como de varios agentes, que son los procesos responsabilizados de copiar los datos entre el publicador y el suscriptor. Estos agentes son: agente de instantáneas, agente de distribución, agente del lector del registro, agente del lector de cola y agente de mezcla.

La replicación de datos es un asunto exclusivamente entre servidores de datos, en nuestro caso hablamos de servidores Oracle. Los servidores Oracle pueden desempeñar uno o varios de los siguientes roles: publicador, distribuidor o suscriptor.
El publicador es un servidor que pone los datos a disposición de otros servidores para poder replicarlos.
El distribuidor es un servidor que aloja la base de datos de distribución y almacena los datos históricos, transacciones y meta datos. **Los suscriptores** reciben los datos replicados.

Una publicación es un conjunto de artículos (este concepto: "artículo de una publicación", es diferente del concepto "artículo o registro de una base de datos", como explicaremos más adelante) de una base de datos. Esta agrupación de varios artículos facilita especificar un conjunto de datos relacionados lógicamente y los objetos de bases de datos que desea replicar conjuntamente. Un artículo de una publicación puede ser una tabla de datos, la cual puede contar con todas las filas o algunas (filtrado horizontal) y simultáneamente contar con todas las columnas o algunas (filtrado vertical), un procedimiento almacenado, una definición de vista, la ejecución de un procedimiento almacenado, una vista, una vista indicada o una función definida por el usuario.

Una suscripción es una petición de copia de datos o de objetos de base de datos para replicar. Una suscripción define qué publicación se recibirá, dónde y cuándo. Las suscripciones pueden ser de inserción o de extracción; y una publicación puede admitir una combinación de suscripciones de inserción y extracción. El publicador (en las suscripciones de inserción) o el suscriptor (en las suscripciones de extracción) solicitan la sincronización o distribución de datos de una suscripción.
El publicador puede disponer de una o más publicaciones, de las cuales los suscriptores se suscriben a las publicaciones que necesitan, nunca a artículos individuales de una publicación. El publicador, además, detecta qué datos han cambiado durante la replicación transaccional y mantiene información acerca de todas las publicaciones del sitio.
La función del distribuidor varía según la metodología de replicación implementada. En ocasiones, se configura como distribuidor el mismo publicador y se le denomina distribuidor local. En el resto de los casos, el distribuidor será remoto, pudiendo coincidir en algún caso con un suscriptor.
Los suscriptores, además de obtener sus suscripciones, en dependencia del tipo y opciones de replicación elegidas, pueden devolver datos modificados al publicador y además pueden tener sus propias publicaciones. [3]

Escenarios típicos de la replicación

En una solución de replicación pudiera ser necesario utilizar varias publicaciones en una combinación de metodologías y opciones. En la replicación, los datos o transacciones fluyen del publicador al suscriptor pasando por el distribuidor. Por lo tanto, en su configuración mínima, una topología de replicación se compone de al menos dos o tres servidores; Oracle desempeña los tres roles mencionados.

Variando la ubicación del servidor distribuidor podríamos contar con las siguientes variantes:

1. El rol de distribuidor desempeñado por el publicador.
2. El rol de distribuidor desempeñado por el suscriptor.
3. Un servidor de distribución, independiente del publicador y del suscriptor.
4. En la mayoría de las configuraciones, el peso fundamental de la replicación recae sobre el servidor de distribución. Por lo tanto éste puede ser un criterio para determinar su ubicación, teniendo en cuenta las configuraciones (posibilidades físicas) de los servidores, así como otras responsabilidades que pueden estar desempeñando (servidor de dominio, servidor de páginas web, entre otras).
5. Existe la posibilidad de contar con un servidor que se suscriba a una publicación y a la vez la publique para el resto de los suscriptores, esto puede ser muy útil cuando se cuente con una conexión muy costosa con el publicador principal. Por ejemplo, el publicador principal en Madrid y los suscriptores en Ciudad Habana, Varadero, Cayo Coco, Cayo Largo, etc. En casos como este, se puede elegir un suscriptor, digamos el servidor de Ciudad Habana, el cual se suscribe al publicador en Madrid y a la vez actúa como servidor de publicación para los servidores de Varadero, Cayo Coco, Cayo Largo y demás. Evidentemente en una configuración tal, pueden nuevamente combinarse la ubicación de los dos distribuidores y aumentar el número de variantes que pueden presentarse, pero las consideraciones para determinar la ubicación del servidor que fungirá como distribuidor son las ya mencionadas.

Los tipos básicos de replicación son:

- Replicación de Instantáneas.
- Replicación Transaccional.

- Replicación de Mezcla.

Para ajustarse aún más a los requerimientos de los usuarios, se incorporan opciones como son la actualización inmediata en el suscriptor, la actualización en cola y la transformación de datos replicados. [3]

1.4.2 Replicación de Instantáneas.

En la replicación de instantáneas los datos se copian tal y como aparecen en un momento determinado. Por consiguiente, no se requiere de un control continuo de los cambios. Las publicaciones de instantáneas, se suelen replicar con menos frecuencia que otros tipos de publicaciones. Puede llevar más tiempo propagar las modificaciones de datos a los suscriptores. Se recomienda utilizar: cuando la mayoría de los datos no cambian con frecuencia; se replican pequeñas cantidades de datos; los sitios con frecuencia están desconectados y es aceptable un período de latencia largo (la cantidad de tiempo que transcurre entre la actualización de los datos en un sitio y en otro). En ocasiones se hace necesario utilizarla cuando están involucrados algunos tipos de datos cuyas modificaciones no se registran en el registro de transacciones y por tanto no se pueden replicar utilizando la metodología de replicación transaccional.

Los servidores OLAP (Procesamiento Analítico en línea) son candidatos a la replicación de instantáneas. Las consultas ad-hoc que aplican los administradores de Sistemas de información son generalmente de sólo lectura y los datos con antigüedad de horas o días no afectan sus consultas. Por ejemplo, un departamento desea hacer una investigación sobre demografía de los artículos vendidos hace dos meses, la información de la semana pasada no afectará sus consultas; además, el departamento no está planeando hacer cambio en los datos, sólo necesita el almacén de datos. Hay que destacar además, que cuando están involucrados algunos tipos de datos cuyas modificaciones no se registran en el registro de transacciones y por lo tanto es necesario transportar estos datos, del publicador al suscriptor, para lo cual es necesario utilizar la replicación de instantáneas, al menos como una solución parcial.

Con la opción de actualización inmediata en el suscriptor se permite a los suscriptores actualizar datos solamente si el publicador los va a aceptar inmediatamente. Si el publicador los acepta, se propagan a otros suscriptores. El suscriptor debe estar conectado de forma estable y continua al publicador para poder realizar cambios en el suscriptor. Esta opción es útil en escenarios en los que tienen lugar unas cuantas modificaciones ocasionales en los servidores suscriptores. [3]

1.4.3 Replicación Transaccional.

En la Replicación Transaccional se propaga una instantánea inicial de datos a los suscriptores, y después, cuando se efectúan las modificaciones en el publicador, las transacciones individuales se propagan a los suscriptores. El SGBD almacena las transacciones que afectan a los objetos replicados y propaga esos cambios a los suscriptores de forma continua o a intervalos programados. Al finalizar la propagación de los cambios, todos los suscriptores tendrán los mismos valores que el publicador. Suele utilizarse cuando: se desea que las modificaciones de datos se propaguen a los suscriptores, normalmente pocos segundos después de producirse; se necesita que las transacciones sean atómicas, que se apliquen todas o ninguna al suscriptor; los suscriptores se conectan en su mayoría al publicador; su aplicación no puede permitir un período de latencia largo para los suscriptores que reciban cambios.

Es útil en escenarios en los que los suscriptores pueden tratar a sus datos como de sólo lectura, pero necesitan cambios a los datos con una cantidad mínima de latencia. Ejemplo: un sistema para el procesamiento y distribución de pedidos; en este tipo de escenario, podría tener varios publicadores recibiendo pedidos de mercancías. Estos pedidos se replican entonces a un almacén central donde se despachan los pedidos. El almacén puede tratar los datos como de sólo lectura y requiere nueva información en forma periódica.

Con el uso de la opción de actualización inmediata en el suscriptor se pierde aún más la autonomía de sitio, pero se reduce el tiempo en el cual los sitios actualizan sus copias de los datos. Para hacer modificaciones en la base de datos suscriptor, éstas se realizan o intentan realizarse también en la base de datos publicador, este paso se ejecuta en una confirmación de dos fases por lo que si su modificación se confirma, indica que es válida , luego en cuestión de minutos o según la planificación hecha, estos cambios son duplicados a las demás bases de datos suscriptoras. [3]

1.4.4 Replicación de Mezcla.

Permite que varios sitios funcionen en línea o desconectados de manera autónoma, y que posteriormente se mezclen las modificaciones de datos realizadas, en un resultado único y uniforme.

La instantánea inicial se aplica a los suscriptores; a continuación Oracle hace un seguimiento de los cambios realizados en los datos publicados en el publicador y en los suscriptores. Los datos se sincronizan entre los servidores a una hora programada o a petición. Las actualizaciones se realizan de

manera independiente, sin protocolo de confirmación en más de un servidor, así el publicador o más de un suscriptor pueden haber actualizado los mismos datos. Por lo tanto, pueden producirse conflictos al mezclar las modificaciones de datos. Cuando se produce un conflicto, el Agente de Mezcla invoca una resolución para determinar qué datos se aceptarán y se propagarán a otros sitios. Es útil cuando: varios suscriptores necesitan actualizar datos en diferentes ocasiones y propagar los cambios al publicador y a otros suscriptores; los suscriptores necesitan recibir datos, realizar cambios sin conexión y sincronizar más adelante los cambios con el publicador y otros suscriptores; el requisito de período de latencia de la aplicación es largo o corto; la autonomía del sitio es un factor crucial.

Es útil en ambientes en los que cada sitio realiza cambios solamente en sus datos pero que necesitan tener la información de los demás sitios. Por ejemplo podría crearse una base de datos que registre la historia delictiva de individuos. En cada municipio de una provincia, se puede tener una copia de la base de datos de toda la provincia y no se requiere estar conectado permanentemente a la base de datos de la instancia provincial. [3]

1.4.5 Factores para elegir un método de Replicación.

En la elección de un método adecuado para la distribución de los datos en una organización influyen varios factores, los cuales podemos agruparlos en dos grupos: factores relacionados con los requerimientos de la aplicación y factores relacionados con el entorno de red.
Dentro de los factores relacionados con los requerimientos de la aplicación, los fundamentales son:

- Autonomía
- Consistencia transaccional
- Latencia

La autonomía de un sitio da la medida de cuanto puede operar el sitio desconectado de la base de datos publicadora. La consistencia transaccional de un sitio viene dado por la necesidad de ejecutar o no inmediatamente todas las transacciones que se han ejecutado en el servidor, o si es suficiente con respetar el orden de las mismas y por último, la latencia de un sitio se refiere al momento en que se deben de sincronizar las copias de los datos. ¿Necesitan los datos estar el 100% en sincronía? O si es admisible determinada latencia ¿de qué tamaño es aceptable el intervalo?

Entre los factores relacionados con el entorno de red están, la velocidad de transmisión de datos de la red, deben considerarse preguntas como ¿Cómo luce la red? ¿Es rápida? Debe analizarse además la confiabilidad de la red y responder preguntas como ¿Cuán confiable es la red? Por otra parte en el caso que los servidores Oracle no permanezcan todo el día encendido, como pudiera suceder en algunas organizaciones, deben considerarse los horarios de disponibilidad de cada servidor.

La consideración de estos factores sirve de guía en la configuración del ambiente de replicación. Además debe considerar las siguientes preguntas: ¿Qué datos se van a publicar? ¿Reciben todos los suscriptores todos los datos o sólo subconjuntos de ellos? ¿Se deben particionar los datos por sitio? ¿Se debe permitir que los suscriptores envíen actualizaciones de los datos? Y en caso de permitirlas ¿Cómo deben implementarse? ¿Quiénes pueden tener acceso a los datos? ¿Se encuentran estos usuarios en línea? ¿Se encuentran conectados mediante enlaces caros? [3]

1.4.6 Fases generales para implementar y supervisar la replicación

A pesar de que existen varias formas de implementar y supervisar la replicación, y el proceso de replicación es diferente según el tipo y las opciones elegidas, en general, la replicación se compone de las siguientes fases:

- Configuración de la replicación.
- Generación y aplicación de la instantánea inicial.
- Modificación de los datos replicados.
- Sincronización y propagación de los datos.[1]

1.5 Características del SGBD Oracle.

El Sistema de Gestión de Bases de Datos Oracle, surgió a final de los años 70 y principio de los años 80. Oracle es básicamente una herramienta cliente-servidor para la gestión de bases de datos, es un producto vendido a nivel mundial, aunque la gran potencia que tiene y su elevado precio hacen que solo se vea en empresas muy grandes y multinacionales, por norma general. Es un sistema gestor de datos relacional de última generación que hace uso de los recursos de los sistemas informáticos en todas las arquitecturas de hardware, lo que permite garantizar su aprovechamiento en ambientes cargados de información, por su capacidad de almacenar y acudir a los datos de forma recurrente.

Esta herramienta está orientada al acceso remoto y redes (Internet). En la actualidad Oracle se puede implementar en diferentes plataformas: Familia de Microsoft, Unix, Linux, Vms, etc..., y es el mayor y más usado Sistema de Gestión de Bases de Datos Relacionales (*RDBMS, por sus siglas en inglés*) en el mundo, además es soportado en computadoras personales (*PC, por sus siglas en inglés*), microcomputadoras, computadoras con procesamiento paralelo masivo. Soporta unos 17 idiomas, funciona automáticamente en más de 80 arquitecturas de hardware y software distintos sin tener la necesidad de cambiar una sola línea de código. Esto es porque más del 80% de los códigos internos de Oracle son iguales a los establecidos en todas las plataformas de sistemas operativos. Oracle es un sistema comercial que aporta un SGBD que ofrece las particularidades básicas para trabajar en entornos multi-usuario.

Las arquitecturas en las que se asienta pueden ser: Intel, Alpha, Sparc, Risc, a nivel de procesadores. Oracle es perfectamente configurable en entornos "OLTP"(Procesamiento de Transacciones en línea), paralelos, Clúster, e incluso resulta una genial solución a nivel de Datawarehouse y CRM (Administración de Relación con los clientes). La plataforma para la que en realidad está desarrollada Oracle, y pensada desde un principio es Unix , ya que es un sistema que soporta la mayoría de la carga de los sistemas a nivel mundial, así como es un sistema abierto y configurable. Para la gestión, Oracle y Unix se complementan de una manera casi inalcanzable para otros sistemas, ya que para su administración, Oracle se amolda y apoya en la potencia de Unix, a la vez que aprovecha al 100% su versatilidad y configurabilidad. [4]

1.5.1 Características de Oracle 11g

La versión 11g de Oracle release 1, salió al mercado en febrero del 2008 con 482 nuevas características. Para así traer al mercado un producto de mayor calidad aún, que las anteriores versiones de Oracle. Este centra sus cambios en los siguientes grupos:

- Nuevas características en el SQL.
- Nuevas características en el soporte del lenguaje.
- Nuevas características en el PL/SQL.
- Nuevas características en el DBA.
- Nuevas características y mejoras en el Oracle Real Application Clusters
- Nuevas características en el Rendimiento.

- Nuevas características de la Seguridad.
- Nuevas características en el Enterprise Manager.

Nuevas características en el SQL.

A partir de 2007, el mercado comercial de la base de datos es muy adulto y las expectativas son altas. Todas las bases de datos principales hacen un buen trabajo en lo referente a almacenar y recuperar datos, y los clientes ahora exigen a las bases de datos, que se afinen solas, motores inteligentes que detectan y corrigen condiciones que no son las más óptimas.

Oracle 11g es el líder evidente en esta área. Este ha invertido fuertemente en incluir capacidades de auto afinación incluyendo almacenamiento automático, administración de memoria y asesores inteligentes de afinación. Ahora la versión 11g cierra el lazo y ofrece herramientas inteligentes de automatización para crear una base de datos autogenerada. Dentro las nuevas características de automatización más importantes de 11g hay que incluir el Asesor de Afinamiento SQL, que corrige automáticamente declaraciones SQL.

- **Afinador de Memoria Automático:** El afinador de memoria automático fue introducido en el Oracle 9i, así como el afinador del Área Global del Sistema (SGA) automático fue introducido con la versión 10g de Oracle. En 11g, toda la memoria puede ser afinada automáticamente con el ajuste de un solo parámetro.
- **Analizador de Rendimiento SQL** (Fully Automatic SQL Tuning): Usando Analizador de Rendimiento SQL, se puede decir que 11g aplica automáticamente los perfiles SQL para declaraciones donde los perfiles propuestos reportan mejores rendimientos, que los existentes por tres ocasiones. La comparación del rendimiento es llevada a cabo por una nueva tarea administrativa.
- **Almacenamiento automatizado, balance de la carga**: El Automatic Storage Management (ASM) ahora habilita una zona de almacenamiento compartida para múltiples bases de datos para así tener un balance óptimo de la carga de datos de cada una de la bases de datos. Los recursos compartidos de almacenamiento del disco alternativamente pueden ser asignados a bases de datos individuales y fácilmente pueden ser movidos de una base de datos a otra como tramitar cambio de requisitos.

Nuevas características en el DBA

- **ILM realizado:** Gestión de la Información del Ciclo de Vida (*ILM, por sus siglas en inglés*) ha sido abordado por décadas, pero Oracle ha dado un empujón para la codificación del acercamiento en 11g.
- **El control de niveles de tablas de estadísticas CBO refrescan el umbral:** Cuando Oracle permite la colección estadística, el umbral de deterioro por defecto del 10 %, ahora se puede modificar con el procedimiento *dbms_stats.set_table_prefs.*
- **Repositorio del grupo de archivos:** Oracle introdujo una nueva característica, llamada Repositorio del Grupo de Archivos (*FGR, por sus siglas en inglés*). El FGR deja al DBA definir un grupo lógicamente relacionado de archivos y construir una infraestructura de control de versión. FGR fue creado para soportar la tecnología Oracle Streams, imitando la funcionalidad del grupo de generación de datos del mainframe de IBM (*GDG, por sus siglas en inglés*).
- **Particionamiento de intervalos para tablas:** Este es un nuevo creador de particiones de esquemas de Oracle 11g, que automáticamente crea particiones basadas en el tiempo. Permite particionar por fechas, una partición por cada mes, por ejemplo, con la creación de particiones automáticas.
- **Nuevas utilidades para balanceado de cargas:** hay varias utilidades para el balanceado de carga que son nuevas en 11g (ya introducidas en 10gr2) como son:
  - Balanceado de carga del servidor Web.
  - Balanceado de carga de instancia RAC.
  - Balanceado de carga de Almacenamiento Automatizado.
  - Balanceado de carga de Data Guard.
  - Balanceado de carga del Listener.

- **Disparadores (triggers) DML más rápidos**: Los disparadores DML se terminan un 25% más rápido. Esto trae un especial impacto a nivel de fila en el momento de hacer actualizaciones a otras tablas.
- **Nuevos Asesores de Oracle11g:** Está el Oracle11g Streams Performance Advisor y el Partitioning Advisor.

- **Orden para disparar un triggers de una tabla:** Oracle 11g PL/SQL te permite especificar el orden para disparar un triggers.

Nuevas características en Oracle 11g de Alta disponibilidad y de RAC

Oracle continúa mejorando el Clúster de Aplicación Real en Oracle11g y se pueden apreciar algunas características nuevas excitantes en la manejabilidad RAC y en mejora del rendimiento:

- **ADDM para RAC** –Oracle incorporara a RAC dentro del monitor automático de diagnósticos de bases de datos.
- **Protocolos de las funciones de caché del RAC optimizadas**: Los Protocolos de las funciones de caché del RAC fueron mejoradas con respecto a la de la versión 10g.
- **Repositorio de Diagnósticos Automáticos (ADR, por sus siglas en inglés):** Cuando se detecta un error crítico, se crea automáticamente un "incidente". La información relacionada con este incidente es capturada automáticamente, se notifica al DBA y ciertos chequeos se corren automáticamente. Esta información puede ser empacada para ser hacia un soporte Oracle.

Nuevas características en el Enterprise Manager:

- **Interfaz de Aplicaciones:** Oracle expresa que las capacidades extendidas del Enterprise Manager son parte de una promesa de la firma de integrar muchas de sus adquisiciones incluyendo Siebel Systems y PeopleSoft Corp. en una sola plataforma.
- **Fácil Desinstalación:** Esto desinstalará lo mismo instalaciones exitosas, como malas instalaciones de Oracle.
- **Soporte de puesto de trabajo**: Una vez que el ADR (por sus siglas en inglés) ha detectado y ha reportado un problema crítico, el DBA puede consultar al ADR, puede escribir un reporte de la fuente del problema, y en algunos casos pude implementar reparaciones.

Nuevas características en el PL/SQL.

- **Estado desactivado para PL/SQL**: una nueva característica para 11g es el estado desactivado para el PL/SQL.

- **La compilación sencilla de PL/SQL:** La compilación nativa ya no requiere que un compilador de C compile su lenguaje PL/SQL. Su código va directamente a una biblioteca compartida.
- **Mecanismo de invalidación de procedimiento almacenado PL/SQL mejorado:** Permitiendo reducir el número de objetos que se invalidan como resultado de DDL.

Nuevas características de la Optimización del Afinamiento de Rendimiento:

- **Administrador de Recursos:** El Administrador de Recursos de 11g puede administrar E/S, no simplemente UPS. Se puede establecer la prioridad asociada con archivos específicos, tipos de archivo o grupos del disco ASM.

- **ADDM**: El ADDM en 11g es usado en el RAC entero, no sólo en el nivel de instancia (el nivel de la base de datos). Las directivas han sido añadidas a ADDM así es que puede ignorar asuntos sin importancia para el usuario. Por ejemplo, si se sabe que necesita más memoria y se lo informan incesantemente, se le puede pedir a ADDM que no reporte más esos mensajes.

Minería de datos en 11g.

- Con la liberación del primer libro de Minería de Datos en Oracle (*ODM, por sus siglas en inglés*), hemos visto el aumento del interés en la minería de datos dentro de 11g y se comenta que ODM será grandemente enriquecido, moviendo los objetos de minería de datos en el diccionario y mejorando la interfaz para análisis complicados. [5]

1.6 Replicación Avanzada

Tipos de ambientes de replicación

Replicación Multimaster

La replicación multimaster, también llamada replicación peer-to-peer o replicación *n-way*, habilita sitios múltiples, que se comportan como pares iguales, para administrar grupos de objetos replicados de la base de datos. Cada sitio en un ambiente de replicación multimaster es un sitio maestro que tiene comunicación con los demás sitios que pertenecen al mismo ambiente.

Las aplicaciones pueden actualizar cualquier tabla replicada en cualquier sitio en una configuración multimaster. El servidor de base de datos Oracle funcionando como el sitio master, en un ambiente multimaster, automáticamente trata de reunir los datos de todas las tablas replicadas y asegurar la integridad y convergencia de los mismos en las transacciones globales.
La replicación asincrónica es usada al implementar una replicación multimaster, aunque existen otras vías como la replicación sincrónica y la replicación de procedimiento. Cuando se usa una replicación asincrónica, la información acerca de los cambios DML realizados a las tablas es almacenada en la cola de transacciones diferidas del sitio master donde se realizan los cambios. Estas transacciones son llamadas transacciones diferidas. Las transacciones diferidas son propagadas a otro sitio master en intervalos de tiempo regulares.

Replicación de Vista Materializada

Una vista materializada contiene una copia completa o parcial del objetivo master, esta pudiera ser una tabla master de un sitio master o una vista materializada master de un sitio de vistas materializadas. Una vista materializada master, es una vista que se desempeña como master de otra vista materializada.

Las vistas materializadas brindan los siguientes beneficios:

- Permite acceso local, que mejora el tiempo de repuesta y la disponibilidad
- Evade las consultas a los sitios master y los sitios de vistas materializadas, ya que los usuarios pueden consultar las vistas materializadas locales.
- Incrementa la seguridad de los datos permitiendo replicar solamente el subconjunto seleccionado del set de datos del objetivo master.

Las vistas materializadas pueden ser de solo lectura, actualizables o escribibles.

Vistas materializadas de solo lectura

Las vistas materializadas de solo lectura brindan los siguientes beneficios:

- Elimina la posibilidad de conflicto, ya que no pueden ser actualizadas.
- Soportan vistas materializadas complejas. Un ejemplo puede ser una vista que contengan la cláusula CONNECT BY.

Vistas materializadas actualizables

Las vistas materializadas actualizables poseen las siguientes propiedades:

- Siempre son basadas en una única tabla, aunque pueden ser referenciadas varias tablas en una subconsulta.
- Pueden ser refrescadas incrementalmente.
- Oracle propaga los cambios hechos a una vista materializada actualizable, a una tabla remota de vista materializada master o a una vista materializada master.

Las vistas materializadas actualizables poseen los siguientes beneficios:

- Permiten al usuario consultar y actualizar un set local de datos replicados incluso cuando se encuentra desconectada del sitio master o del sitio master de vistas materializadas.
- Requiere menos recursos que la replicación multimaster, incluso soportando la actualización de datos.

Vista materializada escribible

Se pueden crear vistas materializadas usando la cláusula FOR UPDATE durante la creación, pero entonces no se puede adicionar la vista materializada a un grupo de vistas materializadas. En este caso el usuario puede realizar cambios DML en la vista materializada, pero este cambio no podrá ponerse en el master y además el cambio se pierde cuando la vista materializada es refrescada. A este tipo de vista se les llama, vista materializada escribible.

Configuración híbrida de multimaster y vistas materializadas

La replicación multimaster y la replicación de vistas materializadas pueden combinarse en una configuración híbrida para obtener requerimientos diferentes. La configuración híbrida puede tener cualquier número de sitios master y varias vistas materializadas para cada master.

Diferencias entre las vistas materializadas y las tablas master replicadas:

Las tablas master replicadas, pueden contener datos de la tablas que se va a replicar completa, considerando que las vistas materializadas pueden replicar subconjuntos de los datos de las tablas master.

La replicación multimaster permite replicar cambios de cada transacción cuando el cambio tiene lugar. La vista materializada propaga los cambios de múltiples transacciones más eficientemente, pero en intervalos menos frecuentes.
Si el conflicto ocurre por hacer cambios a copias múltiples del mismo dato, entonces siempre ocurre una detección y una solución del conflicto en sitios master o en sitios de vistas materializadas. [6]

1.7 Características de la Tecnología Streams de Oracle

Oracle Streams, es una tecnología para compartir información, detectando que información es relevante y quienes la utilizan. Esta tecnología es utilizada por Oracle 11g para propagar los cambios en un ambiente replicado. Se basa en tres acciones aplicadas a la información, Captura, Propagación y Consumo.
Para la replicación la Captura se relaciona con un mecanismo que toma los cambios de los Redo Log. El Almacenamiento se vincula cuando los cambios capturados son enviados al área de almacenamiento y estos cambios son propagados a las áreas de almacenamiento de los equipos remotos.
El Consumo es la acción que se encarga de aplicar los cambios almacenados en la base de datos destino. Las tablas replicadas pueden ser diferentes, Oracle Streams se encarga de transformar la información para ajustarla a la base de datos de cada sitio replicado.
El Proceso de Captura configurado para recolectar cambios realizados, recupera los mismos desde los Redo Log, formatea la información como Registros Lógicos de Cambios (LCR logical change records) y coloca los LCRs en el área de almacenamiento de la base de datos local. Los LCRs son entonces propagados desde el área o cola de almacenamiento de la base de datos origen, hacia las áreas de almacenamiento de las bases de datos destino de la replicación. Los componentes de consumo de las bases de datos destino, son configurados para recibir los cambios y aplicarlos. (Ver figura 1.1). [7]

La figura 1.1 ilustra los procesos de captura, propagación y aplicación de una transacción utilizando Oracle Streams:

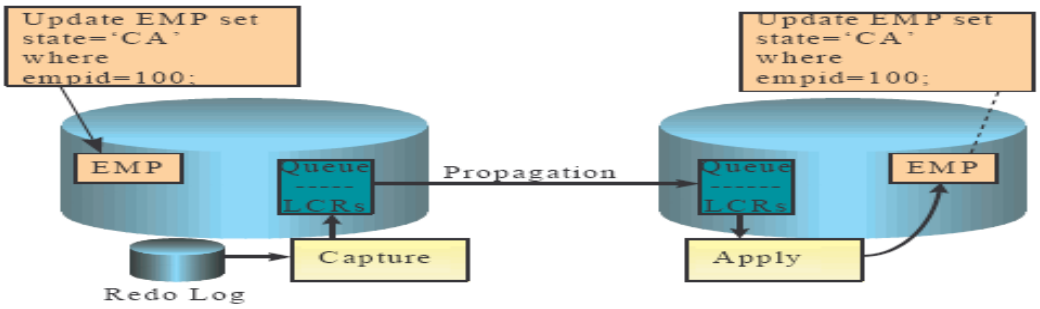

Figura 1.1.

Conclusiones del capítulo

En este capítulo se ha realizado un breve recorrido por conceptos y definiciones que se encuentran incluidos en nuestro trabajo, mediante los cuales nos apoyaremos como punto de estudio y reflexión para la posible toma de decisiones y serán utilizados para la asimilación y desempeño de nuestra investigación. Además se ha realizado un breve estudio acerca de la Replicación de Datos y algunas características de la Tecnología Streams del SGBD Oracle en su versión 11g, la cual constituye uno de los elementos principales de nuestra investigación, sirviendo como inicio al estudio profundo que requiere nuestro trabajo.

CAPITULO 2: ORACLE STREAM 11G, VENTAJAS Y COMPONENTES

Introducción.

En este capítulo se presenta una serie de aspectos importantes para la asimilación e implementación de la tecnología Stream del SGBD Oracle 11g, además se realiza una valoración del método tradicional de replicación que se ha venido utilizando para garantizar la arquitectura de información distribuida del sistema de Circulados Nacionales.

2.1 Tecnología Streams en Oracle 11g

La replicación de bases de datos es una tecnología utilizada por un espectro ancho de usuarios en la industria, el gobierno, academias y para el mantenimiento interno y afrontamiento de problemas de los clientes en aplicaciones que sean altamente disponibles y confiables. Oracle Streams provee a los usuarios una solución para realizar replicación de alto rendimiento, así como la flexibilidad y la funcionalidad requerida por configuraciones que comparten información arbitraria. La base de Datos Oracle 11g cuenta con varias mejoras de la tecnología Streams para optimizar las funciones de configuraciones comunes de los clientes.

El resultado es una característica que proporciona mayor funcionalidad y flexibilidad que las soluciones tradicionales para capturar y manejar mensajes, así como compartir estos con otras bases de datos y aplicaciones. Oracle Streams 11g provee las capacidades necesarias para construir y dirigir aplicaciones y proyectos distribuidos, almacenes de datos, y soluciones de alta disponibilidad. Pueden usarse todas las capacidades de esta tecnología al mismo tiempo. Si las necesidades cambian, puede implementarse una capacidad nueva de Streams sin sacrificar capacidades existentes.

Utilizando la tecnología Streams de Oracle 11g, se controla qué información es introducida en un flujo, cómo avanza el flujo o cómo se enruta de base de datos en base de datos, también controla lo que le ocurre a los mensajes en el flujo, como fluyen en cada base de datos, y cómo termina. Configurando capacidades específicas, se puede abordar requisitos específicos. Basado en sus especificaciones Streams, puede captar, representar y manejar mensajes en la base de datos automáticamente, incluyendo los cambios de lenguaje de manipulación de datos (DML) y los cambios de lenguaje de definición de datos (DDL). También se puede poner mensajes creados por el usuario en un flujo y Streams puede propagar

automáticamente la información para otras bases de datos u otras aplicaciones, cuando los mensajes alcanzan un destino, esta tecnología los puede consumir en dependencia de sus especificaciones. [7]

2.1.1 Nuevas características de Stream en Oracle 11g

La Topología Oracle Streams y el Asesor de Rendimiento de Oracle Streams

La topología de Oracle Streams identifica flujos individuales de mensajes y los componentes de Oracle Streams configurados en cada flujo. Un ambiente Oracle Streams normalmente abarca múltiples bases de datos, y la topología de esta tecnología provee una visión global del ambiente Oracle Streams entero.
El Oracle Streams Performance Advisor reporta medidas de rendimientos para una topología de Oracle Streams, incluyendo rendimiento específico y medidas de latencia. Oracle Streams Performance Advisor también identifica cuellos de botella en una topología Oracle Streams de manera que pueden corregirse. Además, el Oracle Streams Performance Advisor examina los componentes de la tecnología en una topología de dicha tecnología y recomienda formas para mejorar su rendimiento.

La conversión automática de tipos de datos durante la aplicación

Un **proceso** d**e aplicación** convierte automáticamente ciertos tipos de datos cuando hay incompatibilidad entre el tipo de dato de una columna en la fila registro lógico de Cambio (la fila LCR) y los tipo datos de la columna correspondiente en una tabla.

La Forma Simplificada para Restaurar Valores Predeterminados para Parámetros

Pueden establecerse parámetros del proceso de captura y aplicación para su valor predeterminado especificando el valor del parámetro a NULL en los procedimientos:

DBMS_CAPTURE_ADM.SET_PARAMETER y DBMS_APPLY_ADM.SET_PARAMETER.

El Soporte de Tablas de Oracle Streams en un Archivo Flashback

En anteriores ediciones de Oracle, Oracle Streams no soportaba la replicación de cambios a tablas en archivos flashback. En la edición 1 de Oracle 11g (11.1) y posteriores, Oracle Streams soporta tablas en archivos flashback **Soportes Columnas Virtuales de Oracle Streams;** lo mismo sucedía con la replicación de cambios en tablas con columnas virtuales, que no era soportada en anteriores ediciones de Oracle, por Oracle Streams, situación que fue solucionada a partir de la edición 1 del 11g de Oracle (11.1) y posteriores.

Parámetro Nuevo de Proceso de Captura: skip_autofiltered_table_ddl

Un nuevo parámetro de proceso de captura nombrado skip_autofiltered_table_ddl le permite capturar cambios DDL en objetos de la base de datos.

Nuevo Parámetro del Proceso de Aplicación: rtrim_on_implicit_conversion

Un nuevo parámetro del proceso de aplicación denominado rtrim_on_implicit_conversion determina si el proceso de aplicación recorta datos de caracteres durante la conversión automática de tipo de datos.

Captura Síncrona

La Captura Síncrona es un nuevo **cliente de Oracle Streams** que capta cambios de lenguaje de manipulación de datos (DML) hechos a tablas, inmediatamente después de que los cambios han sido confirmados.

Soporte de Oracle Streams para Columnas del XMLType

XMLType es de tipo Oracle-supplied que puede usarse para almacenar y consultar datos XML en la base de datos. Oracle Streams puede capturar, propagar y aplicar cambios a los datos XML Type. Los procesos de captura pueden captar cambios en columnas XMLType almacenadas como las columnas CLOB, pero no pueden captar cambios en las columnas XMLType almacenado como objeto relacional o como XML binario.

Soporte de Oracle Streams para la Encriptación Transparente de Datos

Oracle Streams soporta cambios de captura, de propagación, y de aplicación a columnas que han estado encriptados usando código transparente de datos. Oracle Streams soporta columnas que se encriptaron a nivel de columna o a través de la encriptación del tablespace. La encriptación del tablespace permite encriptar un tablespace entero. Todos los objetos creados dentro de un tablespace encriptado están automáticamente encriptados, incluyendo todas las columnas en los objetos de la base de datos dentro del tablespace. Una vez que una columna es encriptada, los componentes de Oracle Streams manipulan los datos de la columna de la misma manera, ya sea debido a la encriptación por columna o encriptación por tablespace.

El fraccionamiento y Fusión de un Destino Streams

Se puede dividir fácilmente la replicación no disponible, a partir de una reconfiguración de Streams; separar este flujo minimiza el tiempo que se necesita para ponerse al día, cuando la replicación vuelve a estar disponible y evita problemas de rendimiento en una cola global de captura. Cuando la replicación es puesta al día, puede fusionarse y regresar a la configuración original; para implementar esta posibilidad se usan los procedimientos del paquete DBMS_STREAMS_ADM: SPLIT_STREAMS, MERGE_STREAMS_JOB y MERGE_STREAMS.

Monitorear LCRs a través de Stream

El nuevo procedimiento SET_MESSAGE_TRACKING en el paquete DBMS_STREAMS_ADM le deja especificar una etiqueta rastreadora para registros lógicos de cambios (**LCRs**) generados por una sesión de la base de datos. Se puede consultar la vista V$STREAMS_MESSAGE_TRACKING para rastrear los LCRs a través del flujo y ver cómo fueron procesados por cada cliente de Oracle Streams.
El monitoreo LCR es útil si los LCRs no están siendo aplicados como se esperaba por uno o más procesos de aplicación. Cuando esto ocurre, se puede utilizar al monitoreo LCR para determinar donde los

LCRs están detenidos en el flujo y se ocupan del problema en esa posición. También el nuevo parámetro de proceso de captura message_tracking_frequency permite monitorear LCRs automáticamente.

Compare y Converja Objetos Compartidos de la Base de Datos

Un paquete nuevo Oracle-supplied llamado DBMS_COMPARISON le permite comparar las filas en un objeto compartido de la base de datos como una tabla, en dos bases de datos diferentes. Si las diferencias son encontradas en la base de datos objeto, entonces este paquete puede hacer converger los objetos de la base de datos a fin de que sean coherentes.

Alertas Automatizadas para Clientes Oracle Streams y Umbrales

Enterprise Manager alerta automáticamente cuando un cliente Oracle Streams se deshabilita o cuando cierto umbral definido se ha sobrepasado.

Los Trabajos Oracle Streams Usan Planificador Oracle

En ediciones pasadas, Oracle Streams usaba trabajos creados por el paquete DBMS_JOB para realizar trabajos como la propagación y notificación del evento, y el parámetro de inicialización JOB_QUEUE_PROCESS controlaba el número de procesos esclavos que eran creados. En la Edición del 11g de Oracle Database (11.1), Oracle Streams usa el Planificador Oracle para realizar estos trabajos.

2.2 Ventajas de Oracle Streams 11g

Oracle Streams utiliza una única solución para satisfacer todos los requerimientos que se necesitan para compartir información en un ambiente distribuido. Con la utilización de los paquetes PL/SQL, Streams, dispone de la flexibilidad de configuración y administración suficiente, como para poder asumir cualquier cambio en los requisitos planteados inicialmente. Con el uso de esta tecnología es posible la propagación de información dentro de una misma base de datos o entre diferentes bases de datos, logrando compartir información hacia una base de datos que se encuentre en otro sistema que no sea precisamente Oracle, operación que es transparente para Oracle Streams.

Cuando esta tecnología es desplegada, es capaz de compartir todos los tipos datos necesarios para su funcionamiento, sin que sea necesaria la asimilación de nuevos comandos, ni la instalación de un software adicional. Además es posible su utilización en una amplia variedad de configuraciones, como son: la replicación, el guardado de mensajes en cola, las cargas de Data Warehouse y las notificaciones de eventos. Es una tecnología incorporada a Oracle, la cual se apropia de las ventajas de confiabilidad y de seguridad que este proporciona.

En la replicación, Oracle Streams prácticamente no se ve limitada al número de bases de datos destinos, replicando asincrónicamente los datos hacia una gran cantidad de servidores. Brinda una mayor facilidad para la gestión al propagar los cambios DDL. Para ambientes heterogéneos, consta de Gateways y APIs para brindar una mayor seguridad a la base de datos, permitiendo también configurar ambientes bidireccionales, en los cuales el proceso de replicación se realiza entre dos bases de datos en ambos sentidos.

La replicación avanzada consta de dos fases de transacciones distribuidas del protocolo commit, que involucra una gran negociación de espera de acuses de recibo. Streams usa un protocolo distribuido de mensajería, que envía datos con la menor cantidad de acuses de recibo posibles, emitidos por la base de datos remota. Garantizando sólo una vez la entrega transaccional del mensaje, aumentando drásticamente el rendimiento y reduciendo el uso de los recursos de la red.

Streams también se utiliza en las notificaciones de eventos. Esta es una configuración mediante la cual puede avisar la ocurrencia de determinado evento importante o de interés, como pueden ser: rebajas de precio de determinado producto, una noticia en particular, retrasos de vuelos, etc. Estas notificaciones pueden ser enviadas hacia un bíper o teléfono celular. Streams pudiera detectar eventos DML y enviar una notificación hacia una aplicación, que luego esta pudiera mandarla a una cuenta de correo electrónico cualquiera, mejorando de esta forma las prestaciones a los usuarios, sin la necesidad de desarrollar e implementar nuevas aplicaciones. (Ver figura 2.1)

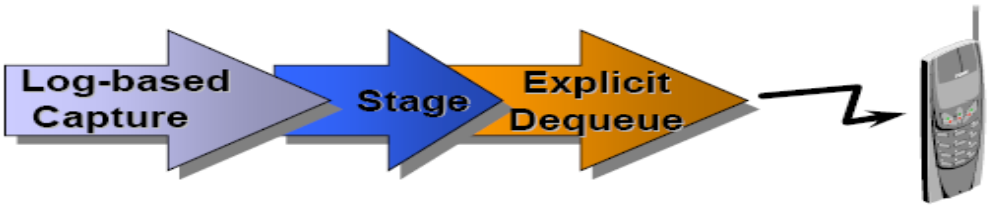

Figura 2.1

Al comparar la replicación de Oracle Streams con la Replicación Avanzada, debemos tener en cuenta un conjunto de características y posibilidades, que a continuación abordaremos.

Para acceder a los redo logs de una forma mas rápida y segura, Oracle implementa el Log Miner, característica que fue introducida por primera vez en la versión 8i de Oracle y se mejoró considerablemente en la versión 9i. En estas primeras versiones el Log Miner solo era capaz de leer

registros redo individuales y reconstruir el dato mediante una sentencia SQL, en posteriores versiones el Log Miner se convirtió en un componente de otros productos usados para la replicación de datos, como son: Oracle Data Guard (Logical Standby Database) y Oracle Streams.

El uso de los redo logs para detectar los cambios hechos a una base de datos es una característica de Stream que incrementa el valor de uso de esta estructura. Además evita los triggers internos para la captura, las copias adicionales en los logs de vistas materializadas y la capa SQL, trabajando a un nivel inferior, permitiendo mejores rendimientos.

Una de las características importantes de Oracle Streams, es la posibilidad de que la base de datos fuente y la base de datos destino, puedan tener diferentes nombres de columnas y los campos, distintos tipos de datos, aún así pueden ser replicados los datos aplicando determinadas transformaciones. Al ser Streams muy configurable y fácil de desplegar, el tiempo, el esfuerzo y el costo de la producción de un sistema que utilice esta tecnología, disminuyen, pues se ahorrarían implementaciones adicionales o la utilización de otros sistemas para lograr la realización de tareas más específicas.

Todas estas razones demuestran la superioridad de Oracle Streams sobre las anteriores tecnologías implementadas por Oracle en otras versiones, pues es una tecnología muy potente y de una gran flexibilidad, con la cual se logra mucho más que la replicación de datos entre dos o más bases de datos.

2.3 Arquitectura de Oracle Streams 11g

Oracle Stream para propagar los cambios en un ambiente de replicación, realiza tres procesos fundamentales: Captura, Propagación y Consumo. A continuación se mostraran las características y capacidades de cada uno de estos.

2.3.1 Captura de Información con Streams 11g.

Existen dos formas para captar información con Oracle Streams: La **captura implícita** y la **captura explícita**.

Con **la captura implícita**, el lenguaje de definición de datos (DDL) y los cambios de lenguaje de manipulación de datos (DML) son captados automáticamente ya sea por un **proceso de captura** o por **la captura sincrónica.** Un tipo específico de mensaje llamado **registros de cambio lógicos** describe estos cambios de la base de datos. El proceso de captura y la captura sincrónica pueden filtrar cambios de la

base de datos, usando **reglas** creadas por el usuario. Por lo tanto, sólo los cambios en los objetos especificados son captados.

Los Tipos de Información Capturada con Oracle Streams

Los Registros Lógicos de Cambios **(LCR** por sus siglas en inglés**)**

Un LCR es un **mensaje** con un formato específico que describe un cambio de la base de datos. Hay dos tipos de LCRs: Los **LCRs de fila** y **DDL LCRs**.

Se puede captar los siguientes tipos de LCRs con Oracle Streams:

- Un LCR capturado es un LCR que es captado implícitamente por un proceso de captura y puesto en la porción de la cola buffered, de una cola ANYDATA.
- Un LCR persistente, es un LCR almacenado en la porción persistente de una cola de ANYDATA. Un LCR persistente puede ser puesto en la cola en una de las siguientes formas:
  - Capturado implícitamente por una captura sincrónica y puesto en la cola.
  - Construido explícitamente por una aplicación y puesto en la cola.
  - Sacado de la cola por un proceso aplicación y puesto en la cola por el mismo proceso de aplicación usando el método SET_ENQUEUE_DESTINATION que pertenece al paquete DBMS_APPLY_ADM.

Los Mensajes de Usuario

Los mensajes que no contienen LCRs son llamados **mensajes de usuario.** Los mensajes de usuario pueden ser de cualquier tipo (exceptuando de tipo LCR). Estos mensajes pueden ser creados y consumidos por aplicaciones. Por ejemplo, una aplicación comercial podría crear un mensaje de usuario para cada orden, y estos mensajes podrían ser procesados por otra aplicación.

Usted puede captar los siguientes tipos de mensajes de usuario con Oracle Streams:

- Un mensaje persistente de usuario es un mensaje de tipo creado por el usuario que no contiene LCR, el mismo se coloca en una cola persistente. Un mensaje persistente de usuario puede ser puesto en la cola de las siguientes formas:

  - Creado explícitamente por una aplicación y puesto en la cola.
  - Sacado de la cola por un proceso aplicación y puesto en la cola por el mismo proceso utilizando el método SET_ENQUEUE_DESTINATION del paquete DBMS_APPLY_ADM.
  - Un mensaje persistente de usuario puede ser puesto en la porción persistente de una cola ANYDATA o una cola con un tipo definido.
- Un mensaje del usuario buffered es un mensaje de tipo creado por el usuario que no contiene LCR, es creado explícitamente por una aplicación y puesto en la cola en una cola buffered.
- Un mensaje de usuario buffered, puede ser puesto en la cola en la porción de la cola buffered de una cola ANYDATA o una cola con un tipo definido.

Proceso de Captura

Cada base de datos Oracle tiene un set de dos o más archivos redo log. Estos archivos para una base de datos son conocidos como los **redo log de la base de datos**. La función primaria del redo log es registrar todo los cambios hechos en la base de datos. Estos archivos se usan para garantizar recuperabilidad en el caso de que se cometa error humano o el soporte informático no funcione debidamente.

Un **proceso de captura** es un cliente de Oracle Streams que se activa de forma opcional, el cual escanea los redo log de la base de datos para captar cambios de lenguaje de manipulación de datos (DML) y los cambios de lenguaje de definición de datos (DDL) hechos en objetos de la base de datos. El proceso de captura es configurado para captar cambios de un redo log, la base de datos donde los cambios fueron generados para el proceso de captura es llamado **base de datos de fuente**. Cuando un proceso de captura capta un cambio en la base de datos, lo convierte a un formato específico de mensaje, **(LCR).** Después de capturar a un LCR, un proceso de captura pone un mensaje conteniendo al LCR en una **cola**. Un proceso de captura solamente es asociado con una cola ANYDATA, cola que puede almacenar mensajes de diferentes tipos, y solo puede poner mensajes en esta cola. Para mejorar el rendimiento, **los LCRs capturados** siempre se guardan en una **cola buffered**, la cual está asociada con la memoria del Área Global del Sistema (SGA por sus siglas en inglés). Usted puede crear múltiples colas, pudiendo asociar procesos de captura diferentes a cada cola creada.

Los LCRs capturados pueden ser enviados a las colas en la misma base de datos u otras bases de datos por propagaciones y también pueden ser sacados de la cola por el proceso aplicación.

En algunas situaciones, una optimización le permite al proceso de captura enviar los LCRs al proceso de aplicación más eficazmente, esta optimización es llamada captura y aplicación combinada.
Un proceso de captura puede correr en su base de datos fuente o en una base de datos remota. Cuando un proceso de captura funciona con su base de datos fuente, el proceso de captura es un **proceso de captura local** y cuando un proceso de captura corre en una base de datos remota, es un **proceso de captura downstream**, la base de datos remota es llamada la **base de datos downstream**. (Ver figura 2.2)

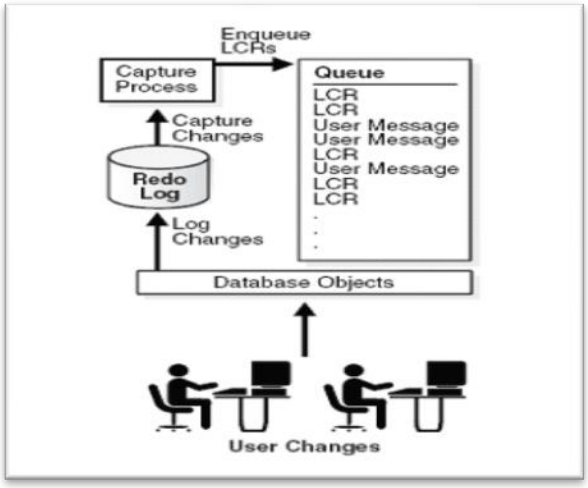

Figura 2.2

Nota: Un proceso de captura y una captura sincrónica no deberían captar cambios hechos para la misma tabla.

Reglas del Proceso de Captura

Un proceso de captura capta o descarta cambios basados en reglas definidas y cada regla especifica los objetos de la base de datos y tipos de cambios a los cuales la regla le da valor TRUE. Estas reglas pueden ser colocadas en un **set positivo de reglas** o en un **set negativo de reglas** para el proceso de captura. Si una regla le da valor TRUE a un cambio y la regla está en el set positivo de reglas para un proceso de captura, entonces dicho proceso capta el cambio. Si un proceso de captura tiene un set positivo y un set negativo de reglas, entonces el set negativo de reglas es siempre evaluado primero.
Se pueden especificar reglas del proceso de captura en los siguientes niveles:

- Una **regla de la tabla** capta o descarta en forma de cambios de fila, cada cambio DML o cambio DDL en una tabla en particular.
- Un **subconjunto de reglas** son las reglas de la tabla que incluyen al subconjunto de cambios de fila que pertenecen a una misma tabla.
- Una **regla de esquema** capta o descarta cada cambio DML o cambio DDL en forma de cambios de fila , en un esquema en particular.
- Una **regla global** capta o descarta todos los cambios DML o todos cambios DDL en la base de datos en forma de cambios de fila.

Nota: El proceso de captura no capta ciertos tipos de cambios ni los cambios en ciertos tipos de datos. Además, un proceso de captura nunca capta cambios en los esquemas SYS, SYSTEM o CTXSYS.

Los Tipos de Datos Captados por los Procesos de Captura

Al captar los cambios de fila que son el resultado de cambios DML hechos en las tablas, un **proceso de captura** puede captar cambios hechos en columnas que tienen los siguientes tipos de datos:

- VARCHAR2
- NVARCHAR2
- FLOAT
- NUMBER
- LONG
- DATE
- BINARY_FLOAT
- BINARY_DOUBLE
- TIMESTAMP
- TIMESTAMP WITH TIME ZONE
- TIMESTAMP WITH LOCAL TIME ZONE
- INTERVAL YEAR TO MONTH
- INTERVAL DAY TO SECOND
- RAW
- LONG RAW
- CHAR
- NCHAR
- UROWID
- CLOB with BASICFILE storage
- NCLOB with BASICFILE storage
- BLOB with BASICFILE storage
- XMLType stored as CLOB

Los tipos de Cambios DML Captados por los Procesos de Captura

Cuando se especifica que los cambios DML hechos para ciertas tablas deberían ser captados, un **proceso de captura** capta los siguientes tipos de cambios DML

- INSERT
- UPDATE
- DELETE
- MERGE

La Captura Local y la Captura Downstream

Se puede configurar un **proceso de captura** para correr localmente en una **base de datos fuente** o remotamente en una **base de datos downstream.** Una sola base de datos puede tener uno o más procesos captura, captando cambios locales y otros procesos de captura captando cambios de una base de datos remota; es decir, se puede configurar una sola base de datos para realizar la captura local y la captura downstream.

La Captura Local

La captura local se basa en un proceso de captura corriendo en la **base de datos de la fuente,** y esta captura realiza todas las acciones de captura de cambios.

Si se configura la captura local, entonces las siguientes acciones son realizadas en la base de datos fuente:

- El método DBMS_CAPTURE_ADM.BUILD es ejecutado para extraer (o construir) el diccionario de datos para el redo log.
- El Supplemental logging de la base de datos fuente coloca información adicional en el redo log.
- Cuando un proceso de captura se inicia por primera vez en una base de datos, Oracle usa la información del diccionario de datos extraída de los redo log para crear un **diccionario de datos LogMiner**. Los procesos de captura que se adicionen, pueden usar este diccionario de datos o pueden crear nuevos diccionarios.
- Un proceso de captura escanea el redo log para cambiar los LogMiner utilizados.
- El **motor de reglas** evalúa cambios basados en las **reglas** en uno o más **sets de reglas** del proceso de captura.
- El proceso de captura pone los cambios que satisfacen las reglas de sus sets de reglas, en una cola local ANYDATA.

- Si los cambios captados son compartidos con una o más bases de datos, entonces una o más propagaciones propagan estos cambios, desde la base de datos fuente, hacia las otras bases de datos.

La Captura Downstream

La captura Downstream está basada en un proceso de captura corriendo, en una base de datos fuera de la base de datos fuente.

Los siguientes tipos de configuraciones de captura downstream son posibles:

- La captura real-time downstream.
- La captura archived-log downstream.

El parámetro de control del proceso de captura downstream real time mine, funciona para cualquier proceso de captura downstream.

Uno o más procesos de captura de archived-log downstream y un proceso de captura real- time downstream pueden coexistir en una base de datos downstream**.**

La Captura Real- Time Downstream

Una configuración de captura **real- time downstream** trabaja de la manera siguiente:

- Los servicios de transporte Redo usan el proceso escritor de log (LGWR por sus siglas en inglés) en la base de datos fuente, para enviar los datos redo a la base de datos downstream ya sea sincrónicamente o asincrónicamente.
- Un proceso remoto (RFS por sus siglas en inglés) del servidor del archivo en la base de datos downstream, recibe los datos redo por la red y almacena los datos redo, en el redo log standby.
- Un cambio log en la base de datos fuente, causa un cambio log en la base de datos downstream.
- El proceso de captura real- time downstream capta cambios del redo log standby, siempre que sea posible y de los archived log standby, cada vez que sea necesario. Un proceso de captura puede captar cambios en los archived logs standby si se queda rezagado. Cuando este proceso se actualiza, comienza a captar cambios del redo log standby nuevamente. (Ver figura 2.3)

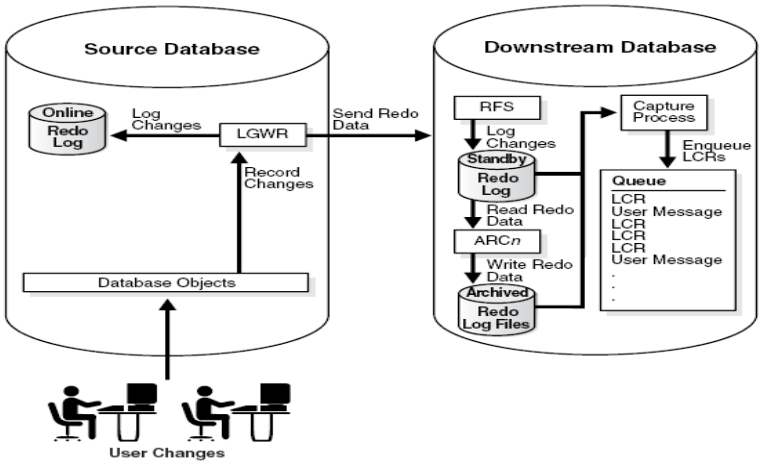

Figura 2.3.

La Captura de Archived-Log Downstream

Una configuración de captura **archived-log downstream** está centrada en que los archivos del redo log guardados en la base de datos fuente, se copian para la base de datos downstream y el proceso de captura capta cambios en estos archivos del redo log archivados. Se puede copiar el redo log files archivado para la base de datos downstream, usando servicios de transporte redo, con el paquete DBMS_FILE_TRANSFER, el protocolo de traslado de archivos (FTP por sus siglas en inglés), o algún otro mecanismo. (Ver figura 2.4)

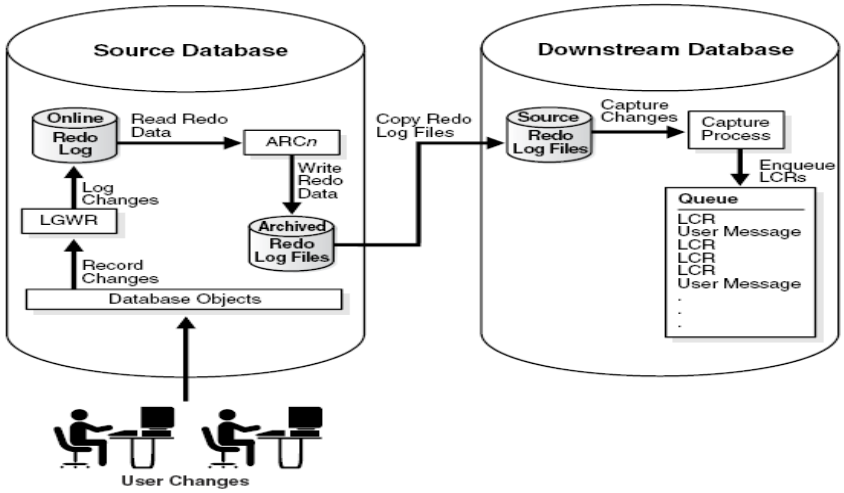

Figura 2.4.

Ventajas de captura real- time downstream sobre la captura de archived-log downstream

La ventaja de captura real- time downstream sobre la captura de archived-log downstream, está en que la captura la primera reduce la cantidad de tiempo requerido para captar cambios hechos en la base de datos fuente, el tiempo se acorta porque el proceso de captura real- time downstream no necesita esperar a que el archivo redo log este archivado para que pueda captar datos en él.

Nota: Sólo un proceso de captura real- time downstream puede existir en una base de datos downstream.

Ventajas de captura de archived-log downstream sobre la captura real- time downstream

Una captura archived-log downstream permite varios procesos de captura en una misma base de datos downstream. Se pueden copiar los archivos redo log de múltiples bases de datos fuente, hacia una única base de datos downstream y luego configurar múltiples procesos de captura archived-log downstream para que capturen los cambios de estos archivos.

Las Ventajas de la Captura Local

- La configuración y la administración del proceso de captura son más simples que cuando se usa la captura downstream.
- Un proceso captura local puede escanear los cambios en el redo log online, antes que la base de datos escriba estos cambios en el archived redo log.
- Cuando se utiliza un proceso captura de archived-log downstream, los archivos archived redo log se copian para la base de datos downstream, después de que la base de datos fuente ha terminado de escribir cambios para ellos, y algunas veces está obligada a copiar los archivos redo log para la base de datos downstream. Sin embargo, un proceso de captura real- time downstream puede captar cambios en el redo log en línea que envió la base de datos fuente.
- La cantidad de datos enviados a través de la red se reducen, porque los datos redo no se copian en la base de datos downstream. Aún cuando los LCRs capturados son propagados para otras bases de datos, los LCRs capturados pueden ser un subconjunto de los cambios totales hechos a

la base de datos, y sólo los LCRs que satisfacen las reglas del set de reglas para una **propagación,** son propagados.

- La seguridad podría ser mejorada porque sólo la base de datos fuente (local) puede acceder a los datos redo. Por ejemplo, si sólo se quiere captar cambios en el esquema hr, entonces cuándo se usa la captura local, sólo la base de datos fuente puede acceder a los datos redo para poner en la cola del proceso de captura, los cambios del esquema hr; sin embargo, cuando usted usa captura downstream, los datos redo se copian para la base de datos downstream, estos datos contienen todos los cambios hechos en la base de datos y no simplemente los cambios hechos para el esquema hr.
- Algunos tipos de **transformaciones basadas en reglas personalizadas** son más simples para configurar si el proceso de captura corre en la base de datos local fuente. Por ejemplo, si usted usa captura local, entonces una transformación basada en reglas personalizada puede usar información de reserva en una variable de sesión del PL/SQL que está poblada con información almacenada en la base de datos de la fuente.
- En un ambiente Oracle Streams dónde los mensajes son captados y aplicados en la misma base de datos, podría ser más simple y podría usar menos recursos para configurar consultas locales y los cálculos que requieren información acerca de los cambios captados y los datos locales.

Ventajas de la Captura Downstream

- Capta cambios usando menos recursos en la base de datos fuente, ya que es en la base de datos downstream es donde corre el proceso de captura de cambios, liberando a las bases de datos fuentes de esta tarea.
- Si usted piensa captar cambios que se originan en múltiples bases de datos fuente. Entonces la administración de proceso de captura puede ser simplificada haciendo correr múltiples procesos de captura de archived-log downstream con bases de datos fuente diferentes en una base de datos downstream. Es decir una base de datos downstream puede actuar como la posición central para la captura de cambio en múltiples fuentes. En tal configuración un proceso de captura real- time downstream puede correr a la base de datos downstream además de los procesos de captura de archived-log downstream.

- Copiar los datos redo para uno o más bases de datos downstream provee protección mejorada en contra de pérdida de datos. Por ejemplo, los archivos redo log en la base de datos downstream pueden servir para recuperación de la base de datos fuente en algunas situaciones.
- La habilidad para configurar múltiples procesos de captura en una o más bases de datos downstream los cuales captan cambios de una sola base de datos fuente provee más flexibilidad y puede mejorar escalabilidad.

Un proceso de captura es útil en las siguientes situaciones:

- Cuando usted quiere captar cambios para un número relativamente grande de tablas.
- Cuando usted quiere captar cambios para los esquemas o para una base de datos entera.
- Cuando usted quiere capturar los cambios DDL.
- Cuando usted quiere captar cambios en una base de datos fuera de la base de datos fuente.

Captura Sincrónica

La **captura sincrónica** es un cliente opcional Oracle Streams que capta cambios del lenguaje de manipulación de datos (DML por sus siglas en inglés) hechos en las tablas. La captura sincrónica usa un mecanismo interno para captar cambios DML en las tablas especificadas. Cuando la captura sincrónica es configurada para captar cambios en alguna tabla, la base de datos que contiene dicha tabla es llamada la **base de datos fuente.** Cuando un cambio DML se hace en una tabla, puede dar como resultado cambios para una o más filas en la tabla.

La captura sincrónica capta cada cambio de la fila y convierte la fila en un formato específico de mensaje llamado **fila de registro lógico de cambios** (fila **LCR).** Después que es capturado el la fila LCR, la captura sincrónica pone en la cola un mensaje que contiene dicha fila. Los LCRs de la fila creados por la captura sincrónica siempre contienen valores de todas las columnas en fila, aun si algunos de las columnas no fueron modificadas por el cambio. (Ver figura 2.5)

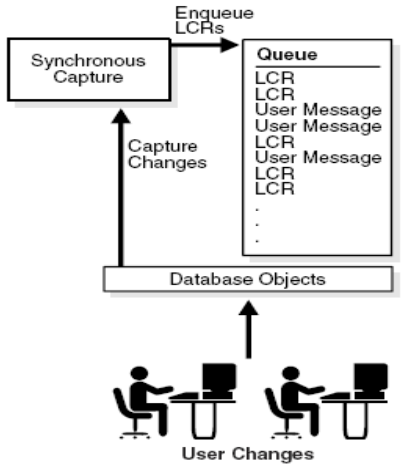

Figura 2.5.

Captura Sincrónica y las Colas

La captura sincrónica es siempre asociada a una sola cola ANYDATA y solo ella pone mensajes en la cola. La cola usada por la captura sincrónica debe ser una **cola Commit time.** Las colas Commit-time aseguran que los mensajes sean agrupados en transacciones, y que los grupos de transacciones estén en orden por **Commit** de número de cambio de sistema (CSCN por sus siglas en inglés).La captura sincrónica siempre pone la fila LCR en una cola persistente. Usted puede crear colas múltiples y puede asociar una captura sincrónica diferente con cada cola. Aunque la captura sincrónica debe poner los mensajes en una cola Commit- time, los mensajes captados por la misma pueden ser propagados para colas que no sean Commit- time. Por lo tanto, cualquier cola intermedia que almacene mensajes captados por la captura sincrónica no necesita ser cola Commit- time. También, los procesos de aplicación que aplican mensajes captados por la captura sincrónica pueden usar colas que no sean colas Commit- time.

Las Reglas de la Captura Sincrónicas

La captura sincrónica, captura los cambios o los descarta, basándose en reglas que usted define. Cada regla especifica los objetos de base de datos y los tipos de cambios, a los cuales la regla le da valor TRUE. Usted puede colocar estas Reglas en un **set positivo de reglas**. Si una regla le da valor TRUE a

un cambio, y dicha regla está en el set positivo de reglas de la captura sincrónica, entonces la captura sincrónica capta el cambio. La captura sincrónica no usa sets negativos de Reglas.
Usted puede especificar reglas de captura sincrónica a nivel de tabla. Una **regla de la tabla** capta o descarta cambios de una fila si son el resultado de cambios DML de una única tabla. . La captura sincrónica no usa Reglas de esquema o globales.

Los Tipos de Datos Capturados por la Captura Síncrona

Al captar los cambios de fila como resultando de cambios DML hechos a las tablas, la captura sincrónica puede captar cambios hechos a las columnas que presentan los siguientes tipos de datos:

- VARCHAR2
- NVARCHAR2
- NUMBER
- FLOAT
- DATE
- BINARY_FLOAT
- BINARY_DOUBLE
- TIMESTAMP
- TIMESTAMP WITH TIME ZONE
- TIMESTAMP WITH LOCAL TIME ZONE
- INTERVAL YEAR TO MONTH
- INTERVAL DAY TO SECOND
- RAW
- CHAR
- NCHAR
- UROWID

Los tipos de Cambios DML captados por la Captura Sincrónica

Cuando usted determina que los cambios DML hechos a tablas específicas deben ser captados, usando una captura sincrónica solo podrá captar los siguientes:

- INSERT
- UPDATE
- DELETE
- MERGE

La captura sincrónica convierte cada cambio MERGE en un cambio INSERT o un cambio UPDATE ya que este no es un comando valido para una fila LCR.

La captura sincrónica es útil en las siguientes situaciones:

- Para obtener un mejor rendimiento, si se quiere capturar los cambios DML para un número relativamente pequeño de tablas.
- Cuando usted quiere captar los cambios DML en una tabla inmediatamente después de estos cambios son realizados.

La Captura Explícita

Con la captura explícita, las aplicaciones generan mensajes y los ponen en la cola. Estos mensajes pueden formatearse como LCRs, o pueden formatearse en los tipos diferentes de mensajes para el consumo por otras aplicaciones. Los mensajes también pueden ser puestos en la cola explícitamente por el proceso de aplicación o por un manipulador aplicaciones de un proceso aplicación.

La captura explícita es útil en las siguientes situaciones:

- Cuando las aplicaciones generan mensajes que deben ser procesados por otras aplicaciones.
- Cuando se tiene un ambiente heterogéneo de la replicación en el cual un proceso de aplicación en una base de datos Oracle aplica cambios originados en una base de datos que no es Oracle.

2.3.2 Propagación de Información con Streams 11g.

Definiciones y conceptos del almacenamiento de mensajes y de la propagación de Información en Oracle Stream 11 g.

Oracle Streams utiliza **colas** para escenificar mensajes. Los mensajes escenificados pueden ser consumidos, propagados o ambos. Estos mensajes pueden ser consumidos por un **proceso de aplicación**, por un **cliente de envío de mensajes**, o una **aplicación del usuario**. Un proceso aplicación en ejecución saca de la cola mensajes de forma implícitamente mientras que los clientes de envío de mensajes y las aplicaciones del usuario sacan de la cola mensajes de forma explícita. Aun después de que un mensaje es consumido puede quedarse en la cola si usted ha configurado una **propagación** para enviar el mensaje a una o más colas, o si en la cola se especifica la retención del mensaje. La retención de mensaje se aplica a los mensajes captados por una captura sincrónica o los mensajes puestos en la cola explícitamente, pero no se aplica a los mensajes captados por un proceso de captura.

Colas Seguras

Las colas seguras son colas para las cuales los agentes de Oracle Streams, Advanced Queuing (AQ por sus siglas en inglés) deben ser asociados explícitamente con uno o más usuarios de la base de datos que pueden realizar operaciones en la cola, tales como poner y sacar de la cola. El dueño de una cola segura

puede realizar todas las operaciones necesarias en la cola, pero otros usuarios no pueden realizar estas operaciones en una cola segura, a menos que sean configuradas como **usuarios seguros de la cola.** En Oracle Streams las colas seguras pueden usarse para asegurar que sólo los usuarios apropiados y los clientes Oracle Streams puedan poner y sacar mensajes de la cola**.**

Colas Seguras y Clientes Oracle Streams.

Cuando usted crea un proceso de captura o un proceso aplicación, asociado con dicho proceso es configurado automáticamente un agente Oracle Streams AQ de la cola segura y el usuario que corre el proceso Oracle Streams es especificado automáticamente como un usuario seguro para esta cola . Por consiguiente, un proceso de captura es configurado para poner automáticamente mensajes en su cola segura y un proceso de aplicación es configurado para sacar automáticamente mensajes de su cola segura. En uno u otro caso el agente Oracle Streams AQ tiene el mismo nombre que el cliente de Oracle Streams.

Propagación de Mensaje entre Colas

Usted puede utilizar Oracle Streams para configurar **propagación de mensaje entre dos colas.** Estas colas pueden cambiar de tamaño en la misma base de datos o en bases de datos diferentes.
Una propagación es siempre entre una cola fuente y una cola destino. Aunque esta se realiza siempre entre dos colas, una sola cola puede participar de muchas propagaciones. Es decir una sola cola fuente puede propagar mensajes a múltiples colas destino, y una sola cola destino puede recibir mensajes de múltiples colas fuente. También una sola cola puede ser destino para algunas propagaciones y fuente para otras propagaciones. Sin embargo sólo una propagación es admitida entre una cola fuente y una cola destino. (Ver figura 2.6).

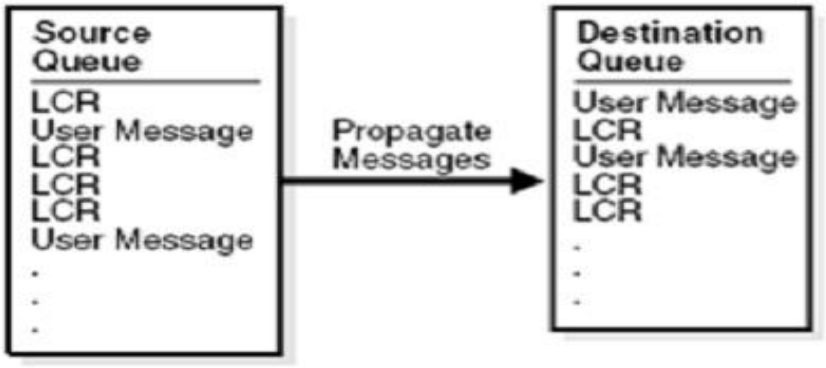

Figura 2.6 Propagación cola Fuente a cola Destino

Se puede crear, alterar, y desechar una propagación, además de poder definir **reglas de propagación** que controlan cuáles mensajes serán propagados o cuales serán descartados. El usuario que posee la cola fuente es el usuario que propaga mensajes, este debe tener los privilegios necesarios para propagar mensajes.

Estos privilegios incluyen lo siguiente:

- El privilegio EXECUTE en los sets de reglas usados por la propagación
- El privilegio EXECUTE en toda la transformación basada en Reglas personalizada funciones usadas en los sets de Regla.
- El privilegio poner en la cola destino si la cola destino está en la misma base de datos

Si la propagación envía mensajes hacia una cola destino en una base de datos remota, entonces el propietario de la cola fuente debe poder usar el enlace de la base de datos usado por la propagación y el usuario para el cual el enlace de la base de datos fue creado debe tener el privilegio de poner mensajes en la cola destino.

Una propagación puede propagar todo los mensajes de una cola fuente para una cola destino, o puede propagar sólo un subconjunto de los mensajes. Una misma propagación puede propagar mensajes a la porción **buffered queue** y la porción **persistent queue** de una cola. También una misma propagación puede propagar LCRs y mensajes del usuario.

Dependiendo de cómo se configure un ambiente Oracle Streams, los cambios podrían ser devueltos al sitio donde se originaron. Usted debe asegurar en la configuración de su ambiente no reciclar un cambio en un lazo interminable, utilizando una **etiqueta** de Oracle Streams para evitar tal lazo de cambio cíclico.

Reglas de Propagación

Una propagación propaga o descarta los mensajes basándose en reglas que son definidas. Para los LCRs, cada regla especifica los objetos de la base de datos y tipos de cambios a los cuales serán evaluados de TRUE. Para los mensajes del usuario se pueden crear reglas de tipos de mensajes específicos, para controlar el comportamiento de la propagación y puede colocar estas reglas en un set positivo de reglas o un set negativo de reglas el cual es usado por la propagación.

Si una regla le da valor TRUE a un mensaje y la misma está en el set positivo de reglas para una propagación, entonces se propaga el cambio, pero si una regla le da valor TRUE a un mensaje, y la regla está en el set negativo de reglas para una propagación, entonces la propagación descarta el cambio. Si una propagación tiene un set positivo y un set negativo de reglas entonces el set negativo de regla, siempre se evalúa primero.

Usted puede especificar reglas de propagación para LCRs en los diferentes niveles. Una **regla a nivel de tabla** propaga o descarta ya sea cambios de la fila como resultado de cambios DML o cambios DDL para una tabla particular. De igual forma sucede a nivel de esquema y a nivel global.

Entrega Asegurada de Mensaje

Un LCR capturado es propagado exitosamente hacia una cola destino cuando las siguientes acciones son completadas:

- El mensaje es procesado por todos los procesos de aplicación relevantes asociados con la cola destino.
- Cuando el mensaje es propagado satisfactoriamente entre dos colas, la cola destino notifica cuando recibe el mensaje. Si la cola fuente se configura para propagar mensajes a múltiples colas destino, entonces el mensaje permanece en la cola fuente hasta que cada cola destino notifique que el mensaje se propagó satisfactoriamente y los consumidores locales de la cola fuente hallan consumido el mensaje, solo así la cola fuente podrá eliminar el mensaje.

Esta confirmación del sistema asegura que el mensaje siempre se propague de la cola fuente a la cola destino, pero en algunas configuraciones la cola fuente puede crecer más que el tamaño óptimo. Cuando el tamaño de esta cola aumenta, aumenta el uso de la memoria (SGA) y consigo el espacio del disco duro. Las razones más comunes por las cuales el tamaño de la cola fuente puede aumentar son:

- Si el mensaje no puede ser propagado por determinada razón (como puede ser problemas con la red) hacia la cola destino especificada, entonces el mensaje permanece en la cola fuente hasta que la cola destino este disponible. Por lo tanto, es conveniente monitorear las colas regularmente, para así detectar problemas a tiempo.
- Suponga que la cola fuente esta propagando mensajes capturados por un proceso de captura o una captura asincrónica hacia múltiples colas destino y una o mas bases de datos destinos están notificando propagaciones satisfactorias de mensajes mucho mas lento que otras. En este caso, la cola fuente puede crecer por que las bases de datos destinos más lentas crean una acumulación de mensajes que ya han sido notificados satisfactoriamente por la bases de datos destino que son más rápidas.

En ambientes como estos es conveniente crear más de una proceso de captura o captura asincrónica para capturar los cambios en la cola fuente. Esto le permitirá usar una cola fuente para bases de datos destinos más lentas y otra para las más rápidas.

2.3.2 Consumo de Información con Streams 11g.

Consumir información con Oracle Streams significa extraer de la cola los mensajes que contienen información para procesarlos o descartarlos. La información consumida puede describir un cambio en la base de datos o puede ser cualquier tipo de información. Un mensaje puede ser desencolado en la misma base de datos donde fue originado o en otra base de datos.

Existen dos formas de consumir o aplicar información con Oracle Streams. Una de estas formas es el **consumo implícito,** con el cual un proceso de captura extrae de una cola automáticamente ya sea un LCR capturado, un LCR persistente, o un mensaje de usuario persistente. La cola debe ser una cola ANYDATA. Si un mensaje contiene un LCR, entonces el proceso de aplicación puede ya sea aplicarlo directamente o llamar un procedimiento especificado por el usuario para su procesamiento. Si el mensaje no contiene un LCR, entonces el proceso de aplicación puede invocar un procedimiento para procesarlo especificado por el usuario llamado manipulador de mensajes.

El **consumo explícito** constituye la otra forma de consumir mensajes, el cual puede realizarse de dos maneras , una con la **extracción manual**, que consiste en una aplicación que extrae explícitamente buffered LCRs, LCRs persistentes, mensajes de usuario buffered, o mensajes de usuario persistentes, a los que posteriormente procesa. La cola de donde son extraídos puede ser lo mismo una cola ANYDATA que una cola con un tipo definido. El **cliente de mensajería** es otra forma de consumo explícito que extrae de la porción persistente de la cola mensajes cuando lo invoca una aplicación o un usuario. Se usan reglas para especificar cual mensaje será extraído por el cliente de mensajería. Este mensaje puede ser un LCR persistente o un mensaje de usuario persistente.

Proceso de Aplicación

El proceso de aplicación es un proceso opcional de Oracle Streams que extrae mensajes de una cola específica y puede aplicar cada mensaje directamente, descartarlos, pasarlos por un manipulador de aplicación o los volverlos a poner en la cola. Este mensaje puede ser un LCR o un mensaje de usuario.

Tipos de datos que soporta el proceso de aplicación

Cuando se aplican filas LCR resultantes de cambios DML hechos a las tablas, el proceso de aplicación aplica los cambios a las columnas que tienen los tipos de datos siguientes:

- VARCHAR2
- NVARCHAR2
- NUMBER
- FLOAT
- LONG
- DATE
- BINARY_FLOAT
- BINARY_DOUBLE
- TIMESTAMP
- TIMESTAMP WITH TIME ZONE
- TIMESTAMP WITH LOCAL TIME ZONE
- INTERVAL YEAR TO MONTH
- INTERVAL DAY TO SECOND
- RAW
- LONG RAW
- CHAR
- NCHAR
- CLOB with BASICFILE storage
- NCLOB with BASICFILE storage
- BLOB with BASICFILE storage
- UROWID
- XMLType stored as CLOB, object relationally, or as binary XML

Componentes del Proceso de Aplicación

El proceso de aplicación consta en los siguientes componentes:

- En un **servidor de escritura** que extrae los mensajes de la cola. Este es el encargado de procesar las dependencias entre los LCRs y ensamblar los mensajes dentro de una transacción. Entonces este retorna la transacción ensamblada al proceso coordinador.
- En un **proceso coordinador**, que es un proceso de fondo que obtiene las transacciones del servidor de escritura y se las envía al servidor de aplicaciones adecuado.
- En uno o mas **servidores de aplicaciones** que aplican LCRs en los objetos de bases de datos, o pasa los LCRs a los manipuladores de aplicaciones apropiados. Para mensajes que no contengan LCRs, el servidor de aplicaciones pasa el mensaje a un manipulador de aplicaciones. El servidor de aplicaciones también puede poner mensajes en la porción persistente de una cola que se específica con el procedimiento DBMS_APPLY_ADM.SET_ENQUEUE_ DESTINATION.

Opciones de procesamiento de mensajes para un proceso de aplicación

Un proceso de aplicación puede aplicar ya sea mensajes directamente o mandar el mensaje a un manipulador de aplicación para que sea procesado. Las opciones para el procesamiento del mensaje dependen del mensaje recibido ya sea una fila LCR, un DDL LCR o un mensaje de usuario.(Ver Figura 2.8).

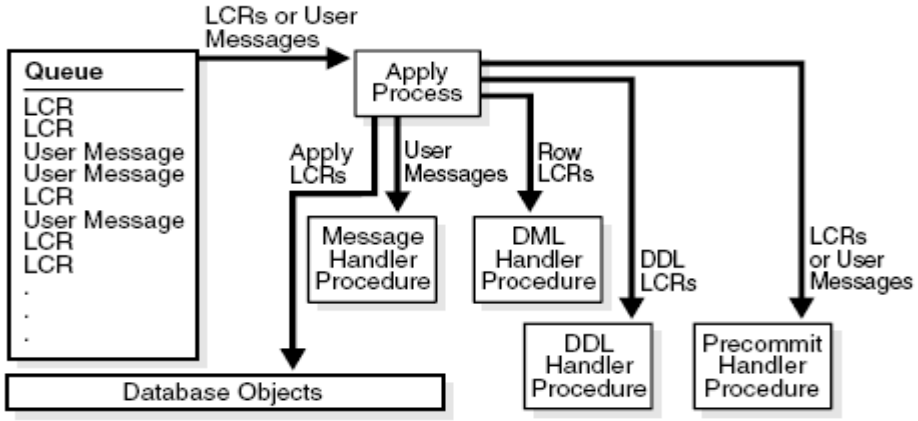

Figura 2.8 Opciones de procesamiento de mensajes para un proceso de aplicación

Aplicar los LCRs directamente

Al usar esta opción el proceso de aplicación aplica los LCRs sin correr ningún procedimiento creado por el usuario. El proceso de aplicación puede aplicar los cambios de los LCRs satisfactoriamente, o si es encontrado un conflicto o un error de aplicación este puede tratar de resolver el error con un manipulador de conflictos o con un procedimiento especificado por el usuario llamado manipulador de errores. Si el manipulador de conflictos puede resolver el problema, entonces este aplicara o descartara el LCR. De igual manera pasa con el manipulador de errores, pero este puede transformar el LCR antes de aplicarlo. Si algunos de estos manipuladores no pueden resolver el error, entonces el proceso de aplicación retiene la transacción y todos los LCRs asociados al este, dentro la cola de errores. El proceso de aplicación no puede aplicar directamente los mensajes que no contengan LCR. Cada mensaje de usuario extraído de la cola debe ser procesado por un manipulador de mensaje.

Procesar mensajes con los manipuladores de aplicación

Cuando usted usa un manipulador de aplicación, el proceso de aplicación le pasa el mensaje como parámetro a un procedimiento del usuario para que lo procese. El procedimiento puede procesar el mensaje en una forma personalizada.

Manipuladores DML (DML Handlers)

Es un procedimiento de usuario que procesa las filas LCRs que fueron captadas por algún proceso de captura, una captura sincrónica o una aplicación. Este procedimiento soporta los siguientes tipos de operaciones.

- INSERT
- UPDATE
- DELETE
- LOB_UPDATE

Manipuladores DDL (DDL Handlers).

Es un procedimiento creado por el usuario que procesa los DDL LCRs que fueron capturados por algún proceso de captura o una aplicación. Se puede usar de varias formas por ejemplo: si usted quiere guardar

un cambio DDL antes de ejecutarlo, entonces puedes crear un procedimiento de usuario que realice esa tarea.

Manipuladores de Mensajes (Message Handlers)

Es un procedimiento definido por usuario que procesa los **mensajes de usuario**. Este procedimiento puede poner mensajes en una cola en la base de datos local y Oracle Streams puede propagar cada mensaje a la cola apropiada de la base de datos destino. Si hay múltiples destinos, entonces Oracle Streams provee la infraestructura para propagar y procesar estos mensajes automáticamente.

Manipuladores Precommit (Precommit Handlers)

Es un procedimiento definido por usuario que puede recibir la información que se envía cuando se realiza un commit en la base de datos. Puede trabajar en conjunto con el DML handler o con message handler.

2.4 Reglas para la Replicación

Una **regla** es un objeto de la base de datos que le permite a un cliente realizar una acción cuando un acontecimiento ocurre y una condición es satisfecha. La misma está constituida por varios componentes.

2.4.1 Componentes de una Regla

Una **regla** consta de los siguientes componentes:

- Condición de Regla.
- Contexto de Evaluación de la Regla (Opcional)
- Contexto de Acción de la Regla (Opcional)

Cada regla es especificada como una condición que es similar a la condición en la cápsula WHERE contenida en la sentencia SELECT de una consulta SQL.
Las reglas que estén relacionadas se pueden agrupar en sets de reglas. Una sola regla puede estar en un solo set de reglas, puede estar en múltiples sets de reglas o no estar en ningún set. Los sets de reglas son evaluados por un motor de reglas, que es una parte incorporada de Oracle. Las aplicaciones creadas por usuario y características de Oracle, como Oracle Streams, pueden ser clientes de del motor de reglas.

Condición de la Regla

Una condición de regla combina una o más expresiones y condiciones, y retorna un valor booleano que puede tener valor: TRUE, FALSE, o NULL.
Una expresión es una combinación de uno o más valores y operadores, los cuales evalúan un valor determinado, el mismo puede ser: un dato en una tabla, un dato en una variable, o un dato retornado por una función SQL o una función PL/SQL. Por ejemplo:

La siguiente expresión incluye solo un valor:

```
salary;
```

La siguiente expresión incluye 2 valores (salary y .1) y un operador (*)

```
salary * .1;
```

La siguiente expresión consiste en 2 expresiones (salary y 3800) y una condición (=):

```
salary=3800;
```

Esta condición lógica evalúa de TRUE una fila determinada, cuando el valor de la columna `salary` es 3800.

Una condición de regla simple puede incluir más de una condición combinada con las condiciones lógicas AND, OR y NOT. Una condición lógica combina los resultados de dos condiciones componentes para producir un solo resultado basado en ellas, o en invertir el resultado de una sola condición. Por ejemplo consideremos la siguiente condición compuesta:

```
salary = 3800 OR job_title = 'Programmer;'
```

Esta condición de regla esta compuesta por 2 condiciones las cuales están unidas por la condición lógica OR. Si cualquiera de las 2 condiciones se cumple entonces la condición de reglas es evaluada de TRUE. Si la condición lógica fuera usando el AND en lugar del OR entonces ambas condiciones deberían cumplirse para que la condición de regla entera sea evaluada de TRUE.

Variables en las Condiciones de Reglas

Las condiciones de reglas pueden tener variables, cuando se usa variables en las condiciones de reglas, estas deben estar precedidas de dos puntos (:). Por ejemplo:

```
: x = 55
```

Las variables permiten referirse a los datos que no se guardan en una tabla. A una variable se le puede asignar una expresión comúnmente usada y así optimizar el funcionamiento de la regla.
Una condición de regla también puede contener una evaluación de una llamada a un subprograma. Tal condición es evaluada de la misma manera que las demás condiciones. Es decir, evalúa para un valor de FALSE, TRUE, o NULL (la incógnita). Por ejemplo:

```
is_manager --esta determina si un empleado es un gerente.
is_manager (employee_id) = 'Y'
```

Aquí el valor de `employee_id` es determinado por un dato en la tabla donde `employee_id` es una columna.
Se pueden usar variables que el tipo es definido por el usuario. Por consiguiente, las variables pueden tener atributos .Cuando una variable tiene atributos, cada atributo contiene datos parciales para esta. En una condición de regla, se especifican atributos usando notación de punto. Por ejemplo:

```
: y.z=9 Expresiones simples de Reglas.
```

Expresiones simples de Reglas

Una condición simple de regla tiene una de las siguientes formas:

- *Simple_rule_expression condition constant*
- *constant condition simple_rule_expression*
- *constant condition constant*

En una condición simple de regla, una expresión simple de regla (*simple_rule_expression)* es una de las siguientes:

- Columna de una Tablas

- Variable
- Atributo de una variable
- Resultado de un método, donde el método no toma argumentos y el resultado del método pueda ser retornado por la función variable del mismo, a fin de que la expresión sea uno de los tipos de datos sustentados para las reglas simples. Estos métodos incluyen subprogramas que cumplen estos requerimientos, tales como GET_TAG, GET_VALUE, GET_COMPATIBLE, GET_EXTRA_ATTRIBUTE, etc.

Los siguientes tipos de datos pueden ser usados en una condición simple de regla para evaluar las columnas de una tabla, las variables, los atributos de variables, y los resultados de métodos:

- VARCHAR2
- BINARY_DOUBLE
- NVARCHAR2
- TIMESTAMP
- NUMBER
- TIMESTAMP WITH TIME ZONE
- DATE
- TIMESTAMP WITH LOCAL TIME ZONE
- BINARY_FLOAT
- CHAR
- RAW

En una condición simple de regla un operador es uno de las siguientes:
(<=; < ; = ; > ; >= ; != ; IS NULL ; IS NOT NULL)

Constantes

Una constante es un valor fijo. O sea un número cualquiera como 12 o 54, un caracter como x o $ o un caracter de tipo string como "esto es un string".

Ejemplos de condiciones simples de Reglas:

Las siguientes son condiciones simples de Reglas, en las cuales se asume que el tipo de dato usado en las expresiones es soportado en las condiciones simples de reglas:

- `tab1.col = 5;`
- `:v1 > 'aaa';`
- `:v4 IS NOT NULL;`
- `tab2.col!= 5;`
- `:v2.a1 < 10.01;`
- `:my_var.my_to_upper ('abc') = 'ABC';`

Reglas con condiciones simples se llaman **reglas simples**. En una regla se pueden combinar 2 o más condiciones simples con las condiciones lógicas AND y OR, y la regla sigue siendo simple. Por ejemplo, reglas con las siguientes condiciones también son reglas simples:

- `tab1.col = 5 AND: v1 > 'aaa';`
- `tab1.col = 5 OR: v1 > 'aaa';`

Pero si se usa la condición lógica NOT entonces la regla ya no es simple.

Beneficios de las Reglas simples:

- Las reglas simples son indexadas internamente por el motor de reglas.
- Las reglas simples pueden ser evaluadas sin ejecutar el SQL
- Las reglas simples pueden ser evaluadas con datos parciales.

Cuando un cliente usa el procedimiento `DBMS_RULES EVALUATE` para evaluar un evento, el cliente puede especificar que solo las reglas simples van a ser evaluadas especificando TRUE para el parámetro: `simple_rules_only`.

2.4.2 Contexto de Evaluación de Regla.

Un contexto de evaluación es un objeto de la base de datos que define datos externos que pueden ser referenciados en las condiciones de Regla. Los datos externos pueden existir, como variables, datos de una tabla, o ambos. Si la condición de regla estuviera en la cápsula WHERE en una consulta SQL, entonces los datos externos en el contexto de evaluación serían las tablas y variables referenciadas en la cláusula FROM de la consulta.

Es decir, las expresiones en la condición de regla deberían establecer referencias para las tablas, deberían crear alias, y las variables ponerlas en el contexto de evaluación para así hacer válida la cláusula WHERE.

Un **contexto de evaluación** de regla provee la información necesaria para interpretar y evaluar las condiciones de regla que establecen referencias para datos externos. Por ejemplo:

Si una Regla se refiere a una variable, entonces la información en el contexto de evaluación de regla debe contener el tipo variable, o si una regla se refiere a un alias de la tabla, entonces la información en el contexto de evaluación debe definir dicho alias. Los objetos a los que se les hace referencia en una regla son determinados por el contexto de evaluación de regla asociado a esta. El propietario de regla debe

tener los privilegios necesarios para acceder a estos objetos, así como el privilegio de realizar SELECT en tablas, EJECUTE, etcétera. La condición de regla se resuelve en el esquema que posee el contexto de evaluación. Por ejemplo consideremos un contexto de evaluación de reglas llamado `hr_evaluation_context` que contiene la siguiente información:

La tabla cuyo alias es `dep`, concuerda con la tabla `hr.departamentos`, las variables `lic_id1` y `loc_id2` son ambas de tipo `NUMBER`, el contexto de evaluación de regla (`hr_evaluction_context`) provee la información necesaria para evaluar la siguiente condición de Regla:

```
dep.location_id IN (:loc_id1,:loc_id2);
```

En este caso, la condición de regla evalúa de TRUE a una fila en la tabla `hr.departments`, si la misma tiene un valor en la columna `location_id` que corresponda con cualquiera de los valores pasados en las variables `bc_id1` o `bc_id2`. La regla no puede ser interpretada o evaluada correctamente sin la información en el `hr_evaluation_context`. También advertir que la notación de punto se usa para especificar el ID de la posición de la columna en el alias de la tabla del `dep`.

Contexto de evaluación y su asociación con los sets de reglas y las reglas.

Para ser evaluada, cada regla debe ser asociada con un contexto de evaluación o debe estar asociada a un set de reglas. Un contexto de evaluación simple puede estar asociado con múltiples reglas o sets de reglas.

La siguiente lista describe cual contexto de evaluación es usado cuando una regla es evaluada:

- Si un contexto de evaluación es asociado con una regla, este es usado para esa regla cada vez que la misma sea evaluada, y cualquier contexto de evaluación asociado con el set de regla será ignorado.
- Si una regla no tiene un contexto de evaluación, pero un contexto de evaluación fue especificado para la regla cuando esta fue adicionada a un set de reglas usando el método `ADD_RULE` en el paquete `DBMS_RULE_ADM`, entonces el contexto de evaluación especificado en el `ADD_RULE` es usado cuando el set de reglas sea evaluado.
- Si una Regla no tiene asociado ningún contexto de evaluación y no se ha especificado ninguno por el método `ADD_RULE`, entonces el contexto de evaluación del set se regla es usado para esa regla cuando se evalué el set.

2.4.3 Contexto de Acción de Regla

Un **contexto de acción** contiene información opcional asociada con una regla que es interpretada por el cliente del motor de reglas, cuando la misma es evaluada para un evento. El cliente del motor de reglas puede ser una aplicación creada por un usuario o una característica de Oracle, como Oracle Streams.

Cada regla solamente puede tener un contexto de acción. La información en un contexto de acción se encuentra en un arreglo de pares (nombre-valor) con el tipo `SYS.RE$NV_LIST`.

La información de un contexto de acción de regla proporciona un contexto para la acción ejecutada por un cliente del motor de reglas cuando una regla toma valor TRUE o MAYBE. El motor de reglas no interpreta el contexto de acción, en lugar de eso retorna este contexto al cliente del motor de regla, que es el encargado de interpretar la información del contexto de acción.

Por ejemplo, supongamos que un evento es definido como la adición de un nuevo empleado de una compañía. Si la información del empleado es guardada en la tabla `hr.employes`, entonces el evento ocurre cada vez que una fila es introducida en esta tabla. La compañía quiere especificar que un conjunto de acciones son realizadas cuando un nuevo empleado es adicionado, pero estas acciones dependen de a cual departamento es adicionado dicho empleado. Una de esas acciones puede ser registrar un empleado en un curso relacionado con el departamento. En este escenario la compañía puede crear una regla para cada departamento con el contexto de acción apropiado. Si la regla es evaluada de TRUE, el contexto de acción retorna específicamente el número del curso que el empleado debe tomar.

A continuación se muestran las condiciones de regla y los contextos de acción para tres departamentos:

Nombre de la regla	Parte de la condición	Contexto de Acción
`rule_dep_10`	`department_id = 10`	`course_number, 1057`
`rule_dep_20`	`department_id = 20`	`course_number, 1215`
`rule_dep_30`	`department_id = 30`	`NULL`

Estos contextos de acción retornan las siguientes instrucciones a la aplicación cliente:

- El contexto de acción para la regla `rule_dep_10` le da instrucciones a la aplicación cliente para matricular al empleado nuevo en el curso número 1057.
- El contexto de acción NULL para la regla `rule_dep_30` le da instrucciones a la aplicación cliente para no matricular al nuevo empleado en ningún curso.

Cada contexto de acción puede contener cero o más pares (nombre-valor). Si un contexto de acción contiene más de un par (nombre-valor), entonces cada nombre en la lista debe ser único.

En este ejemplo, la aplicación cliente para la cual el motor de reglas devuelve el contexto de acción, inscribe al empleado nuevo en el curso con el número devuelto. La aplicación cliente no registra al empleado en un curso si el contexto de acción retorna NULL o si el contexto de acción no contiene un número de curso.

2.4.4 Evaluación de Sets de Reglas

El **motor de reglas** evalúa los sets de reglas contra un evento que es definido por el cliente del motor de reglas. El cliente es el encargado de iniciar la evaluación de un evento, llamando al método `DBMS_RULE.EVALUATE`. Este método le permite al cliente enviar alguna información acerca del evento, al motor de reglas para la evaluación contra un set de reglas. Este evento pudiera tener más información que la que es enviada al motor de reglas.

La siguiente información es especificada por el cliente cuando este llama al método `DBMS_RULE.EVALUATE`:

- El nombre del set de reglas que contiene las reglas a usar para evaluar el evento.
- El contexto de evaluación a usar para la evaluación.

El proceso de evaluación de regla consta de los siguientes pasos:

1. Ocurre un evento definido por un cliente.
2. El cliente inicia evaluación de un set de reglas enviando información acerca de un evento al motor de reglas, usando el método `DBMS_RULE.EVALUATE`.
3. El motor de reglas evalúa la regla establecida para el evento utilizando el contexto de evaluación. El cliente especifica el set de reglas y el contexto de evaluación en la llamada al método

DBMS_RULE.EVALUATE. Solo son usados para la evaluación, las reglas que están especificadas en el set de reglas, utilizando el contexto de evaluación especificado.

4. El motor de reglas obtiene los resultados de la evaluación. Cada regla puede ser evaluada de TRUE, FALSE o NULL.
5. Las reglas evaluadas de TRUE, son retornadas al cliente del motor de reglas, en una lista completa o una por una, estas reglas regresan con su contexto de acción íntegro, el cual puede contener información o ser NULL.
6. El cliente realiza acciones basadas en los resultados devueltos por el motor de reglas. El motor de reglas no realiza acciones basadas en evaluaciones de reglas.

Evaluación Parcial

La evaluación parcial ocurre cuando el método DBMS_RULE.EVALUATE es ejecutado sin datos para todas las tablas y variables en el contexto de evaluación especificado.
Durante la evaluación parcial, algunas reglas pueden establecer referencias para columnas, o atributos que no están disponibles, mientras otras pueden establecer referencias sólo a datos disponibles. Por ejemplo, consideremos un escenario donde solo están disponibles los siguientes datos durante la evaluación:

Columna `tab1.col = 7`
Atributo `v1.a1 = 'ABC'`

Las siguientes reglas son usadas para evaluación:

La Regla R1 tiene la siguiente condición:

```
(tab1.col = 5)
```

La Regla R2 tiene la siguiente condición:

```
(:v1.a2 > 'aaa')
```

La Regla R3 tiene la siguiente condición:

```
(:v1.a1 = 'ABC') OR (:v2 = 5)
```

La Regla R4 tiene la siguiente condición:

```
(:v1.a1 = UPPER ('abc'))
```

Dado el escenario anterior R1 y R4 establecen referencias para datos disponibles, R2 establece referencias para datos no disponibles y R3 establece referencias para datos disponibles y datos no disponibles.
La evaluación parcial sólo evalúa condiciones simples dentro de una regla. Si la condición de regla tiene partes, por lo que no es simple, entonces la regla podría ser o no evaluada completamente, dependiendo de la extensión para la cual los datos están disponibles. Si la regla no es evaluada completamente, entonces esta puede retornar MAYBE.

Dadas las reglas en este escenario, R1 y la primera parte de R3 son evaluados, pero R2 y R4 no son evaluados .El siguiente resultado es retornado al cliente:

- R1 evalúa en FALSE, y no es retornado.
- R2 es retornado como MAYBE porque la información acerca del atributo v1.a2 no esta disponible.
- R3 es retornado como TRUE, porque es una regla simple y el valor de v1.a1 corresponde a la primera parte de la condición de regla.
- R4 es devuelto como MAYBE, porque la condición de regla no es simple. El cliente debe proveer el valor de la variable v1 para esta regla, para que sea evaluada de TRUE o FALSE.

2.4.5 Objetos de la base de datos y privilegios relacionados con las reglas.

Se pueden crear directamente los siguientes tipos de objetos en una base de datos utilizando el paquete `DBMS_RULE_ADM`:

- Contexto de Evaluación
- Reglas
- Sets de reglas

Nota: Usted puede crear reglas y sets de reglas indirectamente usando el paquete `DBMS_STREAM_ADM`.
El usuario controla los privilegios para estos objetos de la base de datos utilizando los siguientes métodos en el paquete `DBMS_RULE_ADM`:

- `GRANT_OBJECT_PRIVILEGE`
- `GRANT_SYSTEM_PRIVILEGE`
- `REVOKE_OBJECT_PRIVILEGE`
- `REVOKE_SYSTEM_PRIVILEGE`

Para permitir a un usuario crear sets de reglas, reglas, y contextos de evaluación en un esquema propio, se debe dar al usuario los siguientes privilegios en el sistema:

- `CREATE_RULE_SET_OBJ`
- `CREATE_RULE_OBJ`
- `CREATE_EVALUATION_CONTEXT_OBJ`

Estos privilegios pueden ser concedidos al usuario directamente, o a través un rol. Todos los privilegios necesarios para la creación de reglas, set de reglas, contextos de evaluación y otros objetos relacionados con reglas, se encuentran dentro del paquete de privilegios `DBMS_STREAMS_AUTH.GRANT_ADMIN_PRIVILEGE`, por lo tanto, si se ejecuta este paquete de privilegios, quedan asignados todos los privilegios necesarios para la administración de cada uno de los componentes y estructuras de datos de Streams.

2.5 Como usar Reglas en Oracle Stream 11g

En Oracle Streams, cada uno de los siguientes mecanismos son llamados "**Cliente Oracle Streams**", porque cada uno es cliente de un motor de reglas cuando son asociados a un set de reglas.

- Proceso de Captura
- Captura Sincrónica
- Propagación
- Proceso de Aplicación
- Mensajería de Cliente

Excepto la Captura Sincrónica, que esta asociada solo a un set positivo de reglas, todos estos clientes pueden ser asociados a dos set de reglas, uno positivo y otro negativo. (Ver Figura 2.9)

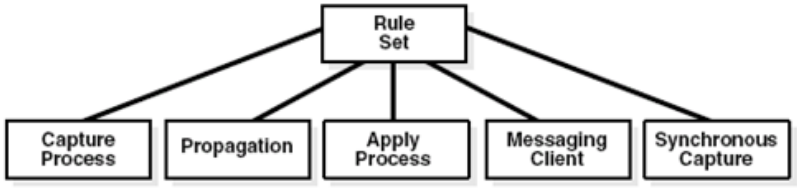

Figura 2.9: Varios clientes de Oracle Streams están asociados a un set de reglas.

2.5.1 Sets de reglas en Oracle Streams

Un cliente Oracle Streams realiza una tarea, si un mensaje satisface sus sets de reglas, siendo utilizados de las siguientes formas:

- Para especificar los cambios que un Proceso de Captura debe captar o descartar del redo log si se cumplen o no las reglas que están en los sets.
- Para especificar los cambios que se deben obtener en una Captura Sincrónica.
- Para especificar los mensajes que son enviados de una cola a otra o los que son descartados.
- Para especificar los mensajes que son descartados o extraídos de la cola, por el Proceso de Aplicación.
- Para especificar los LCRs persistentes o los mensajes persistentes de usuarios, que son descartados o extraídos de la cola por el cliente de mensajería.

Consideraciones para los sets de reglas

- Los clientes Oracle Streams pueden ser configurados sin sets de reglas, los cuales realizan las tareas para todos los mensajes encontrados, pero todo cliente de Oracle Streams, debe tener al menos una regla para los datos que no son soportados por la Base de Datos. La captura sincrónica no soporta esta configuración.
- Los clientes Oracle Streams pueden ser configurados solamente con un set positivo de reglas, el cual cumple las tareas para los mensajes evaluados de TRUE.
- Los clientes Oracle Streams pueden ser configurados solamente con un set negativo de reglas, el cual descarta el mensaje si la regla es evaluada de TRUE.
- Los clientes Oracle Streams pueden ser configurados con un set negativo de reglas y un set positivo de reglas, en este caso si un mensaje es evaluado de TRUE en el set negativo entonces el mensaje es descartado y no es evaluado por el set positivo, de lo contrario se evalúa por el set positivo.

Sets de reglas creados por el usuario

Un usuario de Oracle Streams puede crear sets de reglas personalizados usando la función `CREATE_RULE_SET` del paquete `DBMS_RULE_ADM`.

Ejemplo Sets de Reglas creados por el usuario.

Supongamos que se quiere crear un set de reglas llamado `strmadmin.complex_rules`, utilizando el contexto de evaluación usado por Oracle Streams llamado `SYS.STREAMS$_EVALUATION_CONTEXT`, el procedimiento seria el siguiente:

```
BEGIN
DBMS_RULE_ADM.CREATE_RULE_SET (
rule_set_name => 'strmadmin.complex_rules',
evaluation_context => 'SYS.STREAMS$_EVALUATION_CONTEXT');
END;
```

Nota: Cuando se crea un set de reglas no se especifica si el set es positivo o negativo ya que esto se define al crear el proceso al cual pertenece este set (Captura, Propagación o Consumo), mediante el parámetro `rule_set_name` el cual se usa para sets positivo de reglas y en caso de ser un set negativo de reglas el parámetro que se usaría seria `negative_rule_set_name`.

2.5.2 Reglas en Oracle Streams

Una regla especifica las condiciones que un cambio debe cumplir para que un cliente Oracle Streams realice una tarea específica. Las reglas pudieran estar asociadas a uno o varios sets de reglas o pudieran no estarlo. Una regla puede estar asociada a uno o varios sets positivos de reglas, o a uno o varios sets negativos de reglas pero no puede estar asociada a ambos. Oracle Streams permite realizar reglas globales, reglas para esquemas y reglas para tablas.

Reglas Globales creadas por el sistema

Una **regla global** es especificada cuando se quiere capturar, propagar o aplicar cualquier cambio DML o DDL realizado en la base de datos fuente. Para crear **reglas globales** se usa el paquete `DBMS_STREAMS_ADM` el cual descarta cambios no soportados por Oracle Streams.

Ejemplo de Reglas Globales creadas por el sistema

```
BEGIN
DBMS_STREAMS_ADM.ADD_GLOBAL_RULES (
streams_type => 'capture',
streams_name => 'capture_name',
queue_name => 'streams_queue',
include_dml => TRUE,
include_ddl => TRUE,
include_tagged_lcr => FALSE,
source_database => NULL,
inclusion_rule => TRUE);
END;
```

A continuación se mencionan las especificaciones de cada uno de los parámetros del método `ADD_GLOBAL_RULES` del paquete `DBMS_STREAMS_ADM`.

`streams_Type`: Tipo de proceso para el cual se a creado la regla (Captura, Propagación o Consumo).
`streams_name`: Nombre del proceso para el cual se creo la regla.
`queue_name`: Nombre de la cola en la que se guardan los LCRs para el proceso.
`include_dml`: Con valor TRUE establece que se deben capturar cambios DML.
`include_ddl`: Con valor TRUE que se deben capturar cambios DDL.
`include_tagged_lcr`: Establece si se incluyen las etiquetas de los LCRs para el posterior monitoreo de estado de los LCRs.
`source_database`: Nombre de la base de datos fuente.
`inclusion_rule`: Define si se incluye o no la regla en el set de regla positivo. Tomando valor TRUE para el set positivo de reglas y FALSE para el set negativo de reglas.

Nota: Cuando se usa una regla creada por el sistema, esta crea automáticamente un set de reglas al cual se le asocia esta regla, el mismo puede ser positivo o negativo en dependencia del valor especificado para el parámetro `inclusion_rule`.

Reglas para Esquemas, creadas por el sistema

Una **regla para esquema** es especificada cuando se quiere capturar, propagar o aplicar cualquier cambio DML o DDL realizado en un esquema específico de la base de datos fuente.
Para crear **reglas a esquemas** se usa el paquete `DBMS_STREAMS_ADM` el cual descarta cambios no soportados por Oracle Streams.

Ejemplo de Reglas para Esquemas creadas por el sistema

Supongamos que se quiere crear una regla de propagación para un esquema usando el método `ADD_SCHEMA_PROPAGATION_RULES` del paquete `DBMS_STREAMS_ADM`, el procedimiento seria el siguiente:

```
BEGIN
DBMS_STREAMS_ADM.ADD_SCHEMA_PROPAGATION_RULES (
schema_name => 'hr',
streams_name => 'dbs1_to_dbs2',
source_queue_name => 'streams_queue',
destination_queue_name => 'streams_queue@dbs2.example.com',
include_dml => TRUE,
include_ddl => TRUE,
include_tagged_lcr => FALSE,
source_database => 'dbs1.example.com',
inclusion_rule => TRUE);
END;
```

A continuación se mencionan las especificaciones de cada uno de los parámetros del método ADD_SCHEMA_PROPAGATION_RULES del paquete DBMS_STREAMS_ADM:

schema_name: Nombre del esquema en el cual se crea la regla de propagación.
streams_name: El nombre del proceso para el cual se creo la regla.
queue_name: Nombre de la cola fuente en la que se guardan los LCRs para el proceso.
include_dml: Con valor TRUE establece que se deben capturar los cambios DML.
include_ddl: Con valor TRUE establece que se deben capturar los cambios DDL.
include_tagged_lcr: Establece si se incluyen las etiquetas de los LCRs para el posterior monitoreo de estado de los LCRs.
source_database: Nombre de la base de datos fuente.
inclusion_rule: Define si se incluye la regla en un set de regla positivo o en un set de reglas negativo. Utilizando el valor TRUE para el set positivo y el valor FALSE para el set negativo.

Ejemplo de reglas para tablas creadas por el sistema

Una **regla para tabla** es especificada cuando se quiere capturar, propagar o aplicar cualquier cambio DML o DDL realizado a una tabla específica de la base de datos fuente. Para crear reglas a tablas se usa el paquete DBMS_RULE_ADM, el cual descarta cambios no soportados por Oracle Streams.
Supongamos que se quiere crear una regla de propagación para una tabla usando el método ADD_TABLE_RULES del paquete DBMS_STREAMS_ADM, el procedimiento seria el siguiente:

```
BEGIN
DBMS_STREAMS_ADM.ADD_TABLE_RULES (
table_name => 'hr.locations',
streams_type => 'apply',
streams_name => 'apply',
queue_name => 'streams queue',
include_dml => TRUE,
include_ddl => FALSE,
include_tagged_lcr => FALSE,
source_database => 'dbs1.example.com',
```

```
inclusion_rule => TRUE);
END;
```

A continuación se mencionan las especificaciones de cada uno de los parámetros del método `ADD_TABLE_RULES` del paquete `DBMS_STREAMS_ADM`:

`table_name`: Nombre de la tabla para la cual la regla va a evaluar cambios.
`streams_type`: Este parámetro se refiere al tipo de proceso para el cual se ha creado la regla (Captura, Propagación o Consumo).
`streams_name`: El nombre del proceso para el cual se creo la regla.
`queue_name`: Nombre de la cola en la que se guardan los LCRs para el proceso.
`include_dml`: Con valor TRUE establece que se deben capturar cambios DML.
`include_ddl`: Con valor TRUE establece que se deben capturar cambios DDL.
`include_tagged_lcr`: Este parámetro establece si se incluyen las etiquetas de los LCRs para el posterior monitoreo de estado de los LCRs.
`source_database`: Nombre de la base de datos fuente.

Reglas de mensajes

Cuando se emplea una regla para especificar una tarea en Oracle Streams, que es aplicable solo para un **mensaje de usuario** de un tipo específico de mensaje que no contiene LCRs, se esta especificando una **regla de mensaje.** Se pueden especificar **reglas de mensajes** para propagaciones, procesos de aplicación, y cliente de mensajería.
Una **regla simple de mensaje** en el set positivo de reglas para una propagación, propaga los mensajes de usuarios que cumplan la condición de regla. Una **regla simple de mensaje** en el set negativo de reglas para una propagación descarta los mensajes de usuarios que sean de tipo mensaje en la cola fuente que cumplan la condición de regla.

Ejemplo de Regla de Mensaje

La regla creada en este ejemplo soporta mensajes de los siguientes tipos:

```
CREATE TYPE strmadmin.region_pri_msg AS OBJECT (
```

```
region VARCHAR2 (100),
priority NUMBER,
message VARCHAR2 (3000))
```

El siguiente ejemplo saca de la cola mensajes si la región es EUROPA y la prioridad es 1. Usando el método ADD_MESSAGE_RULE para crear una regla para mensajes de tipo region_pri_msg.

```
BEGIN
DBMS_STREAMS_ADM.ADD_MESSAGE_RULE (
message_type => 'strmadmin.region_pri_msg',
rule_condition => ':msg.region = ''EUROPE'' AND ' ||
':msg.priority = ''1'' ',
streams_type => 'dequeue',
streams_name => 'msg_client',
queue_name => 'streams_queue',
inclusion_rule => TRUE);
END;
```

El método ADD_MESSAGE_RULE crea una regla con una condición de regla similar a la siguiente:
`:"VAR$_52".region = 'EUROPE' AND:"VAR$_52".priority = '1'`

Nota: Las Reglas pueden ser también creadas por el usuario o creadas por el sistema con condiciones definidas por el usuario.

Reglas creadas por el sistema con condiciones definidas por el usuario

Algunos de los métodos para crear reglas en el paquete DBMS_STREAMS_ADM, incluyen un parámetro and_condicion, este parámetro brinda la posibilidad de adicionar condiciones a las reglas creadas por el sistema.
La variable en la condición especificada debe ser: lcr convirtiéndose a: dml o: ddl en dependencia de la regla que se genere. Por ejemplo para especificar que una regla de tabla generada por el ADD_TABLE_RULES evalúa de TRUE solo si la tabla es hr.departaments, la base de datos fuente es

dbs1.example.com y la etiqueta Oracle Streams es el hexadecimal equivalente de ' 02 ' el procedimiento seria el siguiente:

```
BEGIN
DBMS_STREAMS_ADM.ADD_TABLE_RULES (
table_name => 'hr.departments',
streams_type => 'apply',
streams_name => 'apply_02',
queue_name => 'streams_queue',
include_dml => TRUE,
include_ddl => TRUE,
include_tagged_lcr => TRUE,
source_database => 'dbs1.example.com',
inclusion_rule => TRUE,
and_condition => ':lcr.get_tag () = HEXTORAW (''02'')');
END;
```

Reglas creadas por el usuario

Un usuario de Oracle Streams puede crear de forma personalizada reglas simples y compuestas, usando el paquete DBMS_RULE_ADM.

Ejemplo de Reglas creadas por el usuario

Supongamos que se quiere crear una regla llamada strmadmin.hr_not_regions_dml la cual será satisfecha por los cambios realizados en el esquema hr y que no estén comprendidos en la tabla REGIONS, utilizando el contexto de evaluación usado por Oracle Streams llamado SYS.STREAMS$_EVALUATION_CONTEXT, esta regla pertenece al set de reglas strmadmin.complex_rules.

El procedimiento para este ejemplo seria el siguiente:

```
DBMS_RULE_ADM.CREATE_RULE (
rule_name => 'strmadmin.hr_not_regions_dml',
```

```
condition => ' (:dml.get_object_owner () = ''HR'' AND NOT ' ||
': dml.get_object_name () = ''REGIONS'') AND ' ||
': dml.is_null_tag () = ''Y'' ');
```

Una vez creada, la regla debe ser adicionada al set de reglas.

El procedimiento para este ejemplo seria el siguiente:

```
DBMS_RULE_ADM.ADD_RULE (rule_name => 'strmadmin.hr_not_regions_dml',
rule_set_name => 'strmadmin.complex_rules');
```

Para eliminar una regla o un set de reglas, se utiliza el método `DROP_RULE` para las reglas y el método `DROP_RULE_SET` para los sets, ambos del paquete `DBMS_RULE_ADM`. Por ejemplo:

```
---Eliminando Regla
EXEC DBMS_RULE_ADM.DROP_RULE (rule_name => 'strmadmin.hr_not_regions_dml');
---Eliminando Set de Regla
EXEC DBMS_RULE_ADM.DROP_RULE_SET (rule_set_name => 'strmadmin.complex_rules',
delete_rules => TRUE);
```

Nota: El parámetro `delete_rules` habilita la eliminación de las reglas comprendidas en el set de reglas.

2.5.3 **Sub Set de Reglas**

Un Sub-Set de reglas es un tipo especial de regla de tabla para cambios DML que se realizan solo en un Sub-Set de filas de la tabla. Se pueden crear Sub-Set de Reglas para procesos de captura, capturas sincrónicas, procesos de aplicación y cliente de mensajería, usando el método `ADD_SUBSET_RULES`. Para crear **Sub-Sets de reglas para la propagación** se usa el método: `ADD_SUBSET_PROPAGATION_RULES`.

Ejemplo de Sub-Set de Reglas

Este ejemplo muestra un proceso de aplicación de Oracle Streams que utiliza un Sub-Set de reglas para aplicar un sub-set de filas LCRs, utilizando el método `ADD_SUBSET_RULES`. El cual crea automáticamente tres reglas, una para operaciones INSERT, otra para operaciones UPDATE y otra para DELETE.

```
BEGIN
DBMS_STREAMS_ADM.ADD_SUBSET_RULES (
table_name => 'hr.regions',
dml_condition => 'region_id=2',
streams_type => 'apply',
streams_name => 'apply',
queue_name => 'streams_queue',
include_tagged_lcr => FALSE,
source_database => 'dbs1.example.com');
END;
```

Sub-Set de Reglas y Supplemental Logging

El Supplemental Logging es requerido cuando se especifica los siguientes tipos de Sub-Set de reglas:

- Sub-Set de reglas para un **proceso de captura**.
- Sub-Set de reglas para una **propagación** que valla a propagar LCRs capturados por un proceso de captura.
- Sub-Set de reglas para un **proceso de aplicación** que valla a aplicar LCRs capturados por un proceso de captura.

En cualquiera de estos casos un incondicional **Supplemental Logging** debe ser especificado en la base de datos fuente, para todas las columnas en la condición del Sub-Set y para todas las columnas en la tabla de la Base de Datos destino en la que se aplicaran los cambios.
En algunos casos donde un Sub-Set de reglas es especificado, un UPDATE puede ser convertido a un INSERT, en este caso la información suplementaria podría ser necesaria para todas o una cierta cantidad de columnas.

Uso de Sub-Set de Reglas para la captura, cuando todos los destinos necesitan solo un Sub-Set de cambios.

Un Sub-Set de Reglas debería ser utilizado con un proceso de captura o una captura síncrona cuando todas las Bases de Datos destino, para los cambios capturados, necesiten solo cambios de filas que

satisfagan una serie de condiciones para la tabla. En este caso, un **proceso de captura** o una **captura sincrónica,** captura un Sub-Set de cambios DML en la tabla y una o más propagaciones envían estos cambios, que se encuentran en las filas LCRs, a una o más Bases de Datos destino.
En cada Base de Datos destino, un proceso de consumo o aplicación, aplica estas filas LCRs a un Sub-Set de la tabla, los cuales deben satisfacer, el conjunto de condiciones del Sub Set de Reglas para un proceso de captura.
En cada Base de Datos destino, un proceso de aplicación o consumo, aplica estas filas LCRs a un Sub-Set de tabla, en el cual todas las filas cumplen el conjunto de condiciones en el Sub Set de Reglas, para el proceso de captura.

Uso de Sub-Set de Reglas para una Propagación o para un proceso de Aplicación cuando algunos destinos necesitan Sub-Set de cambios.

Un Sub-Set de Reglas debería ser usado en una propagación o en un proceso de aplicación, cuando son sólo algunas de las bases de datos destinos del ambiente, las que necesitan un Sub-Set de cambios DML capturados. Por ejemplo:

- Cuando la mayor parte de de las bases de datos destino para cambios DML capturados en una tabla, necesitan un Sub-Set diferente de estos cambios.
- Cuando la mayor parte de las bases de datos destino necesitan todos los cambios DML capturados en una tabla, pero algunas bases de datos destino necesitan solo un Sub-Set de estos cambios.

Sintaxis

```
DBMS_STREAMS_ADM.ADD_SUBSET_PROPAGATION_RULES (
table_name IN VARCHAR2,
dml_condition IN VARCHAR2,
streams_name IN VARCHAR2 DEFAULT NULL,
source_queue_name IN VARCHAR2,
destination_queue_name IN VARCHAR2,
include_tagged_lcr IN BOOLEAN DEFAULT FALSE,
source_database IN VARCHAR2 DEFAULT NULL,
insert_rule_name OUT VARCHAR2,--parámetro opcional
```

```
update_rule_name OUT VARCHAR2, --parámetro opcional
delete_rule_name OUT VARCHAR2, --parámetro opcional
queue_to_queue IN BOOLEAN DEFAULT NULL);
```

A continuación se mencionan las especificaciones de cada uno de los parámetros del método `ADD_SUBSET_PROPAGATION_RULES` del paquete `DBMS_STREAMS_ADM`.

`table_name`: Nombre de la tabla para la cual se crea el Sub-Set conformado de la siguiente forma: nombre del esquema. [Nombre de la tabla].
`dml_condition`: Condición para el Sub-Set, conformado con sintaxis similar al de la cláusula WHERE de la sentencia SELECT.
`streams_name`: Nombre de la propagación (si no existe, se crea automáticamente).
`source_queue_name`: Nombre de la cola fuente.
`destination_queue_name`: Nombre de la cola destino.
`include_tagged_lcr`: Con valor TRUE considera siempre un LCR para la propagación a pesar de tener etiqueta no NULL(es recomendado para cuando se quiere propagar una base de datos entera). Con valor FALSE considera un LCR para la propagación, solo cuando tiene etiqueta NULL(Es recomendado para configuraciones que se actualicen en cualquier momento, para evitar que se envíen cambios de vuelta a la base de datos origen).
`insert_rule`: Especifica contener la regla generada por el sistema llamada INSERT, la cual maneja los inserts así como también los updates que pueden ser convertidos en inserts.
`update_rule`: Especifica contener la regla generada por el sistema llamada UPDATE, la cual maneja los updates.
`delete_rule`: Especifica contener la regla generada por el sistema llamada DELETE, la cual maneja los deletes, así como también los updates que pueden ser convertidos en deletes.
`queue_to_queue`: Especifica si es propagación Cola a Cola.

2.5.4 Contexto de Evaluación

Un Contexto de Evaluación es utilizado por Oracle Streams para evaluar Reglas Globales, Reglas de Esquema, Reglas de Tablas o Sub-Set de Reglas, Oracle Streams utiliza un contexto de evaluación llamado `STREAMS$_EVALUATION_CONTEXT. PUBLIC` que se crea durante la instalación de Oracle.

Un usuario de Oracle Streams puede utilizar un **contexto de evaluación personalizado** en un ambiente Oracle Streams, el cual se crea usando el método, `CREATE_EVALUATION_CONTEXT` contenido en el paquete `DBMS_RULE_ADM`.

Cuando se crea el contexto de evaluación usando el método `CREATE_EVALUATION_CONTEXT`, entonces la función `SYS.DBMS_STREAMS_INTERNAL.EVALUATION_CONTEXT_FUNCTION` debe ser definida para el parámetro `evaluation_function`. Se puede modificar contextos de evaluación ya creados con el método `ALTER_EVALUATION_CONTEXT`, del paquete `DBMS_RULE_ADM`.

Ejemplo de creación de un contexto de evaluación personalizado:

El siguiente ejemplo utiliza un alias creado para la tabla VEH_NAC perteneciente a BCIRCUL.

```
--Creando alias.

DECLARE
  ta  SYS.RE$TABLE_ALIAS_LIST;
BEGIN
  ta: = SYS.RE$TABLE_ALIAS_LIST (SYS.RE$TABLE_ALIAS ('BVEH',
'BCIRCUL.VEH_NAC')
  );
--Creando context

  DBMS_RULE_ADM.CREATE_EVALUATION_CONTEXT (
    evaluation_context_name    => 'STRM_EVACTX,
    table_aliases              => ta,
    evaluation_context_comment => 'ALIAS PARA LA TABLA');
END;
```

2.5.5 Reglas Basadas en Trasformaciones

Una Regla Basada en Transformación es una modificación a un mensaje cuando una Regla en un Set Positivo de Reglas, es evaluada de TRUE.
Oracle Streams permite 2 tipos de Reglas Basadas en Transformaciones:

- Reglas Basadas en Transformaciones Declarativas.
- Reglas Basadas en Transformaciones Personalizadas.

Reglas Basadas en Transformaciones Declarativas

Las Reglas Basadas en Transformaciones Declarativas cubren un grupo de transformaciones comunes para escenarios de filas LCRs. Se especifica o declara, este tipo de transformación usando uno de los siguientes métodos en el paquete DBMS_STREAMS_ADM:
ADD_COLUMN: Transformación declarativa que adiciona columnas a una fila LCR.
DELETE_COLUMN: Transformación declarativa que elimina columnas a una fila LCR.
RENAME_COLUMN: Transformación declarativa que renombra columnas de una fila LCR.
RENAME_SCHEMA: Transformación declarativa que renombra el esquema de una fila LCR.
RENAME_TABLE: Transformación declarativa que renombra la tabla donde se encuentra una fila LCR.
Cuando se especifica una transformación basada en reglas declarativas, se debe especificar también la regla que estará asociada a esta. Cuando la regla especificada es evaluada de TRUE para una fila LCR, Oracle Streams realiza la transformación declarativa sin invocar al PL/SQL.

Ventajas

- Es funcionalmente mejor y menos compleja que las transformaciones basadas en reglas personalizadas ya que la transformación es realizada internamente sin usar PL/SQL.
- Se reduce la complejidad, ya que no son requeridas las funciones PL/SQL personalizadas.

Reglas Basadas en Transformaciones Personalizadas

Requieren de una función PL/SQL definida por el usuario para realizar transformaciones. Esta función tiene como entrada un objeto de cualquier tipo que contiene un mensaje y retorna este objeto transformado, esta puede devolver un objeto transformado (función de transformación uno a uno) o varios

objetos transformados, almacenados en un arreglo (función de transformación uno a varios). La función de transformación uno a uno es soportada por cualquier cliente de Oracle Streams, sin embargo la función de transformación uno a varios es soportada solamente por clientes de procesos de captura y captura sincrónica.

Para especificar una Regla Basada en Transformaciones Personalizada, se utiliza el método DBMS_STREAMS_ADM.SET_RULE_TRANSFORM_FUNCTION. Este tipo de transformaciones son usadas para modificar LCRs capturados, LCRs persistentes y mensajes de usuarios persistentes.

Por ejemplo, si se tienen 2 objetos de cualquier tipo de datos en 2 bases de datos y estos son de tipos diferentes, la función toma como entrada un objeto de un tipo de dato y da como salida un objeto con el mismo valor, pero con el tipo de datos que se encuentra en la fila LCR.

Ventajas

- La flexibilidad es mejorada ya que puede usar PL/SQL
- Un rango mayor de transformaciones pueden ser realizadas incluyendo LCRs que contienen DDL y mensajes de usuarios, así como filas LCRs.

Reglas basadas en transformaciones en el proceso de captura

Para que una transformación sea realizada durante la captura por un proceso de captura, una Regla que es asociada con una Regla Basada en Transformación en el Set Positivo de Reglas para dicho proceso, debe ser evaluada de TRUE para un cambio en particular encontrado en el redo log.

Si esta transformación es una Transformación Declarativa, entonces Oracle transforma el LCR capturado internamente, cuando la regla en un Set Positivo de Reglas es evaluada de TRUE para un mensaje.

Si la transformación es una Transformación Personalizada**,** entonces un contexto de acción que contiene un par (nombre-valor) con el nombre STREAMS$_TRANSFORM_FUNCTION o STREAMS$_ARRAY_ TRANS_FUNCTION es retornado por el proceso de captura, cuando la regla en un set positivo de reglas es evaluada de TRUE para el LCR capturado.

El proceso de captura completa los siguientes pasos para realizar una Regla Basada en Transformaciones:

1. Formatea el cambio en el redo log en un LCR.

2. Convierte el LCR en un objeto ANYDATA.
3. Transforma el LCR. Si la transformación es Basada en Regla Declarativa, entonces Oracle transforma el objeto ANYDATA internamente basándose en las especificaciones de la transformación declarativa. Si la transformación es basada en regla personalizada, entonces el usuario de captura para el proceso de captura ejecuta la función PL/SQL en el par de valor de nombre para transformar el objeto ANYDATA.
4. Pone en la cola asociada al proceso de captura, uno o mas objetos ANYDATA transformados, o descarta el LCR si un arreglo retornado por la función de transformación contiene 0 elementos.

Todas estas acciones son realizadas por el usuario de la captura.

La figura 2.10 muestra una transformación durante un proceso de captura.

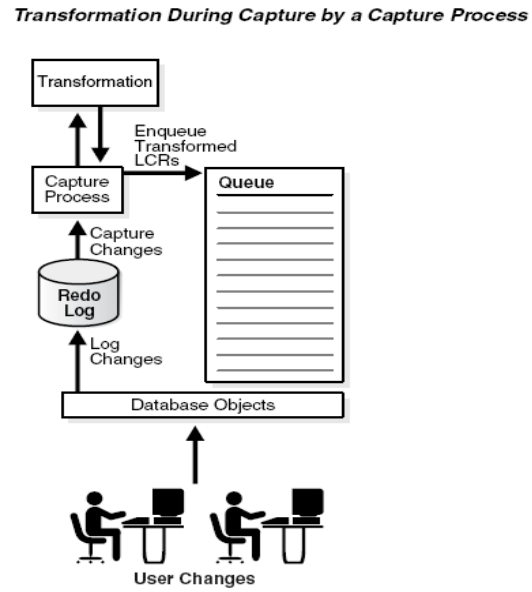

Figura 2.10

Reglas basadas en transformaciones en el proceso de propagación

Para que una transformación sea realizada durante la propagación, una regla que es asociada con una regla basada en transformación en el set positivo de reglas para la propagación, debe ser evaluada de TRUE para un mensaje en la cola fuente .Este mensaje puede ser un LCR capturado, LCR guardado, LCR persistente, mensaje de usuario guardado, o mensaje de usuario persistente.

Si esta transformación es una Transformación Declarativa, entonces Oracle transforma el LCR capturado internamente, cuando la regla en un set positivo de regla es evaluada de TRUE para un mensaje.
Si la transformación es una Transformación Personalizada, entonces un contexto de acción que contiene un par (nombre-valor) con el nombre STREAMS$_TRANSFORM_FUNCTION es retornado por la propagación cuando la regla en el set positivo de regla es evaluada de TRUE para un mensaje.

El proceso de captura completa los siguientes pasos para realizar una regla basada en transformaciones:

1. Comienza sacando el mensaje de la cola fuente.
2. Transforma el mensaje .Si la transformación es basada en regla declarativa, entonces Oracle transforma el mensaje internamente basándose en las especificaciones de la transformación declarativa. Si la transformación es basada en regla personalizada, entonces el propietario de la cola fuente ejecuta la función PL/SQL en el par de valores de nombres para transformar el mensaje.
3. Completa la extracción de mensajes transformados de la cola.
4. Propaga los mensajes transformados a la cola destino.

La figura 2.11 muestra una transformación durante una propagación.

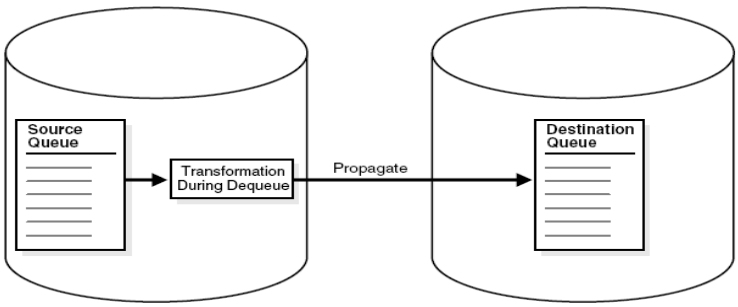

Figura 2.11

Reglas basadas en transformaciones en el proceso de aplicación o consumo

Para que una transformación sea realizada durante el proceso de aplicación, una regla que es asociada con una Regla Basada en Transformaciones en el Set Positivo de Reglas para un proceso de aplicación debe evaluar de TRUE para un mensaje en la cola para el proceso de aplicación. Este mensaje puede ser un LCR capturado, LCR guardado LCR persistente, mensaje de usuario guardado, o mensaje de usuario persistente. Si esta transformación es una Transformación Declarativa, entonces Oracle transforma el LCR capturado internamente, cuando la regla en un set positivo de regla es evaluada de TRUE a un mensaje. Si la transformación es una Transformación Personalizada**,** entonces un contexto de acción que contiene un par (nombre-valor) con el nombre STREAMS$_TRANSFORM_FUNCTION es retornado por la propagación cuando la regla en el set positivo de regla es evaluada de TRUE para un mensaje.

El proceso de aplicación completa los siguientes pasos para realizar una regla basada en transformaciones:

1. Comienza a sacar los mensajes de la cola.
2. Transforma el mensaje .Si la transformación es basada en regla declarativa, entonces Oracle transforma el mensaje internamente basándose en las especificaciones de la transformación declarativa. Si la transformación es basada en regla personalizada, entonces el propietario de la cola fuente ejecuta la función PL/SQL en el par de valores de nombres para transformar el mensaje.
3. Completa la extracción de mensajes transformados de la cola.
4. Aplica los mensajes transformados, lo cual puede conllevar a cambiar objetos de la base de datos destino o enviar el mensaje transformado a un apply handler.

La figura 2.12 muestra una transformación durante un proceso de aplicación.

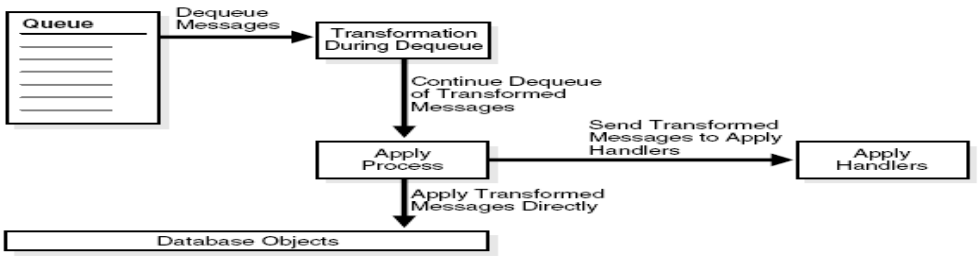

Figura 2.12

Un usuario de Oracle Streams puede realizar transformaciones a columnas, esquemas y tablas usando reglas basadas en transformaciones.

Reglas basadas en transformaciones para renombrar columnas

Si se desea renombrar una columna en una fila LCR se usa el método RENAME_COLUMN del paquete DBMS_STREAMS_ADM. Este método adiciona o remueve una regla basada en transformación declarativa la cual renombra una columna en una fila LCR que satisface una regla específica. Para que la transformación sea realizada cuando la regla específica es evaluada de TRUE, la regla debe estar en el Set Positivo de reglas de un cliente Oracle Streams.

Sintaxis

```
DBMS_STREAMS_ADM.RENAME_COLUMN (
rule_name IN VARCHAR2,
table_name IN VARCHAR2,
from_column_name IN VARCHAR2,
to_column_name IN VARCHAR2,
value_type IN VARCHAR2 DEFAULT '*',
step_number IN NUMBER DEFAULT 0,
operation IN VARCHAR2 DEFAULT 'ADD');
```

A continuación se mencionan las especificaciones de cada uno de los parámetros del método `RENAME_COLUMN` del paquete `DBMS_STREAMS_ADM`.

`rule_name`: Nombre de la regla a la que se le asigna la regla basada en transformación (nombre del esquema.[nombre de la regla]).
`table_name`: Nombre de la tabla en la fila LCR en la cual se va a renombrar la columna (nombre del esquema. [nombre del objeto]).
`from_column_name`: Nombre de la columna que se va a renombrar en cada fila LCR que cumpla la regla especificada en el parámetro `rule_name`.
`to_column_name`: Nuevo nombre para la columna en la fila LCR que se va a renombrar.

value_type: Se especifica en 'NEW 'si se quiere modificar la columna en el nuevo valor para la fila LCR, o especifica en 'OLD' si se quiere modificar la columna en el valor viejo para la fila LCR. Se especifica '*' para modificar ambos.
step_number: Orden de ejecución para la transformación.
operation: Especificar ADD para adicionarle la transformación a la regla o REMOVE para eliminar la transformación de la regla.

Reglas basadas en transformaciones para renombrar esquemas

Si se desea renombrar un esquema en una fila LCR se usa el método RENAME_SCHEMA del paquete DBMS_STREAMS_ADM de forma similar a la función RENAME_COLUMN vista anteriormente. Este método adiciona o remueve una regla basada en transformación declarativa la cual renombra un esquema en una fila LCR que satisface una regla específica. Para que la transformación sea realizada cuando la regla específica es evaluada de TRUE, la regla debe estar en el Set Positivo de Reglas de un cliente Oracle Streams.

Sintaxis

```
DBMS_STREAMS_ADM.RENAME_SCHEMA (
rule_name IN VARCHAR2,
from_schema_name IN VARCHAR2,
to_schema_name IN VARCHAR2,
step_number IN NUMBER DEFAULT 0,
operation IN VARCHAR2 DEFAULT 'ADD');
```

Reglas basadas en transformaciones para renombrar tablas

Si se desea renombrar una tabla en una fila LCR se usa el método RENAME_TABLE del paquete DBMS_STREAMS_ADM de forma similar a la función RENAME_SCHEMA vista anteriormente. Este método adiciona o remueve una regla basada en transformación declarativa la cual renombra una tabla en una fila LCR que satisface una regla específica.

Para que la transformación sea realizada cuando la regla específica es evaluada de TRUE, la regla debe estar en el Set Positivo de Reglas de un cliente Oracle Streams.

Sintaxis

```
DBMS_STREAMS_ADM.RENAME_TABLE (
rule_name IN VARCHAR2,
from_table_name IN VARCHAR2,
to_table_name IN VARCHAR2,
step_number IN NUMBER DEFAULT 0,
operation IN VARCHAR2 DEFAULT 'ADD');
```

2.6 Configuración de un Ambiente de Replica con Oracle Streams 11g

Preparación del ambiente Oracle Streams para Configuración

A continuación se muestra como se prepara una base de datos para configurar un ambiente Oracle Streams.

Configurar un Administrador de Oracle Streams

Para administrar un ambiente Oracle Streams, no se debe usar los usuarios SYS o SYSTEM y el administrador de Oracle Streams no debe usar el SYSTEM tablespace como su tablespace por defecto.

Pasos para configurar un administrador de Oracle Streams.

En SQL*Plus, conéctese con un usuario administrador el cual pueda crear usuarios, otorgar privilegios y crear tablespace. Manténgase conectado por este usuario para realizar los siguientes pasos:

1. Cree un tablespace el administrador de Oracle Streams o use uno que ya exista. La siguiente sentencia crea un nuevo tablespace para el administrador de Oracle Streams:

```
CREATE TABLESPACE streams_tbs DATAFILE '/usr/oracle/dbs/streams_tbs.dbf'
SIZE 25M REUSE AUTOEXTEND ON MAXSIZE UNLIMITED;
```

2. Cree un nuevo usuario que será el administrador de Oracle Streams o use uno ya existente. La siguiente sentencia crea un usuario strmadmin y especifica que usara el tablespace `streams_tb`:

```
CREATE USER strmadmin IDENTIFIED BY password
DEFAULT TABLESPACE streams_tbs
QUOTA UNLIMITED ON streams_tb
```

3. Asigne al administrador de Oracle Streams el rol DBA:

```
GRANT DBA TO strmadmin;
```

El rol DBA es requerido por el usuario para crear o alterar el proceso de captura, la captura sincrónica y el proceso de aplicación. Cuando el usuario no necesite realizar estas tareas entonces el rol DBA puede ser quitado de los permisos del usuario.

4. Corra el procedimiento `GRANT_ADMIN_PRIVILEGE` en paquete `DBMS_STREAMS_AUTH`, para conceder tales privilegios para el administrador Oracle Streams, o se puede conceder directamente.

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

Parámetros relevantes de inicialización en Oracle Streams:

ALTER SYSTEM SET UNDO_RETENTION
ALTER SYSTEM SET GLOBAL_NAMES
ALTER SYSTEM SET AQ_TM_PROCESSES
ALTER SYSTEM SET JOB_QUEUE_PROCESSES

Configurar Oracle Streams Pool

El Oracle Streams pool es un porción de memoria en el Área del Sistema Global (SGA) que es usada por Oracle Streams. Oracle Streams pool almacena en memoria la cola buffered de mensajes y brinda memoria al proceso de captura y al proceso de aplicación. El pool de Oracle Streams siempre almacena los LCRs capturados por el proceso de captura y mensajes puestos en la cola buffered por una aplicación. El tamaño del pool de Oracle Streams es determinada en una de las siguientes formas:

- Usando el Automatic Memory Management
- Usando el Automatic Shared Memory Management
- Ajustando el tamaño del Oracle Streams Pool manualmente.
- Usando el ajuste por defecto del tamaño del Oracle Streams Pool.

Ejemplo:

```
ALTER SYSTEM SET  STREAMS_POOL_SIZE = 200M SCOPE = SPFILE;
```

Configurar la Conectividad de Red y los Enlaces de Bases de Datos

Si se tiene planeado usar Oracle Streams para compartir información entre bases de datos, entonces configure la conectividad de la red y los enlaces de base de datos entre estas bases de datos.
Si su objetivo es propagar mensajes de una cola fuente de una base de datos hacia una cola destino de otra, entonces cree un enlace de base de datos privado entre las dos bases de datos. Cada enlace de base de datos usa la clausula CONNECT TO para propagar mensajes de una base de datos a otra.
Por ejemplo, para crear un enlace de base de datos a una base de datos llamada `dbs2.example.com` conéctese como el administrador de Oracle Streams llamado strmadmin y corra la siguiente sentencia:

```
CREATE DATABASE LINK dbs2.example.com CONNECT TO strmadmin
IDENTIFIED BY password
USING 'dbs2.example.com';
```

Preparando para Configurar Un Proceso de Captura

Las siguientes tareas deben ser completadas antes de que se configure un proceso de captura:

- Configure en modo ARCHIVELOG, cualquier **base de datos fuente** que generara datos redo que serán capturados por un proceso de captura. Para **un procesos de captura downstream**, la **base de datos downstream** también deben correr en el modo ARCHIVELOG. Si se piensa configurar un **proceso captura real- time downstream,** la base de datos downstream no necesita correr en el modo ARCHIVELOG si se quiere correr sólo **el proceso de captura archived- log downstream** en ella.
- Asegúrese de que los parámetros de inicialización son puestos correctamente en cualquier base de datos que correrá un proceso de captura.
- Cree a un administrador Oracle Streams en cada base de datos involucrada en la configuración Oracle Streams. Los ejemplos en este capítulo suponen que el administrador de Oracle Streams es strmadmin.

- Para crear un proceso de captura, al administrador Oracle a Streams le debe ser concedido el rol DBA, pero este rol puede ser cancelado después que el proceso de captura es creado si es necesario.
- Cree una cola ANYDATA en caso que no exista ninguna ,para asociarla con el proceso de captura. Los ejemplos en este capítulo suponen que la **cola** usada por el proceso de captura es la cola strmadmin.streams.
- Cree la cola en la misma base de datos que correrá el proceso de captura.
- Especifique el supplemental logging para algunas columnas en una **base de datos fuente** para que los cambios a las columnas sean aplicados exitosamente en la **base de datos destino.** Típicamente, **supplemental logging** es requerido en ambientes **de la replicación** Oracle Streams, pero podría ser requerido en cualquier ambiente que procesa **LCRs capturados** con un **proceso aplicación**.
- El procedimiento ADD_TABLE_RULE, automáticamente configura supplemental logging a cualquier columna de llave primaria, de llave única, de índice bitmap y de llave foráneas en las tablas replicadas. Además, se puede usar la función ALTER DATABASE para especificar un supplemental logging para todas las tablas en una base de datos o para una tabla particular.

Configuración del Proceso de Captura

Una vez que se hayan realizados los pasos anteriores, entonces se puede comenzar con la **configuración del proceso de captura**, pudiendo crear procesos de capturas locales que captan cambios en la base de datos fuente o procesos de capturas downstream que captan cambios remotos con los siguientes procedimientos:

```
DBMS_STREAMS_ADM.ADD_TABLE_RULES
DBMS_STREAMS_ADM.ADD_SUBSET_RULES
DBMS_STREAMS_ADM.ADD_SCHEMA_RULES
DBMS_STREAMS_ADM.ADD_GLOBAL_RULES
DBMS_CAPTURE_ADM.CREATE_CAPTURE
```

Cada procedimiento en el paquete DBMS_STREAMS_ADM crea un proceso de captura con un nombre especificado, si este no existe. Puede crear un set positivo de reglas o uno negativo, y pude adicionar a este, reglas de tablas, reglas de esquemas o reglas globales.
El procedimiento CREATE_CAPTURE crea un proceso de captura, pero no crea la regla para dicho proceso. De todas formas permite asociarle un set de reglas que ya exista, sea negativo o positivo, permite también establecer un first SCN y un start SCN. Un proceso de captura downstream solo se pude crear usando este paquete.

Ejemplo de Configuración de un Proceso de Captura Local usando DBMS_STREAMS_ADM.

Para configurar un proceso de captura local usando el paquete DBMS_STREAMS_ADM, complete los siguientes cambios:

1. Complete las tares de preparación para la configuración del Proceso de captura.
2. Corra el procedimiento ADD_TABLE_RULES que esta en el paquete DBMS_STREAMS_ADM para crear un proceso de captura local. Por ejemplo:

```
BEGIN
DBMS_STREAMS_ADM.ADD_TABLE_RULES (
table_name => 'hr.employees',
streams_type => 'capture',
streams_name => 'strm01_capture',
queue_name => 'strmadmin.streams_queue',
include_dml => TRUE,
include_ddl => TRUE,
include_tagged_lcr => FALSE,
source_database => NULL,
inclusion_rule => TRUE);
END;
```

Al correr este método se ejecutan las siguientes acciones:

- Se crea solo si no existe un proceso de captura llamado strm01_capture. Si se crea un nuevo proceso de captura, entonces este procedimiento también pone al start SCN hacia el punto en tiempo de creación.
- Asocia el proceso de captura con una cola existente llamada streams_queue.

- Crea un set positivo de reglas y lo asocia con el proceso de captura si el proceso de captura no tiene un set positivo de reglas, ya que el parámetro inclusion_rule esta establecido en true. El nombre del Set de Reglas es generado por el sistema.
- Crea dos reglas. Una reglas evalúa en true para los cambios DML en la tablas hr.employees t, y la otra regla da valor true para los cambios DDL de la misma tabla. Los nombres de las reglas son generados por el sistema.
- Adiciona las dos reglas al Set Positivo de Reglas asociado con el proceso de captura. La regla es adicionada al Set Positivo de Reglas porque el parámetro inclusion_rule tiene valor true.
- Especifica que el proceso de captura, capturará los cambios en el redo log solo si un cambio tiene una etiqueta nula, ya que el parámetro include_tagged_lcr tiene valor false.
- Habilita el supplemental logging para cualquier columna de la tabla ya sea: llave primaria, llave única, índice bitmap y llave foránea.

Ejemplo de Configuración de un Proceso de Captura Local usando DBMS_CAPTURE_ADM.

Para configurar un proceso de captura local usando el paquete DBMS_CAPTURE_ADM, complete los siguientes cambios:

1. Complete las tares de preparación para la configuración del Proceso de captura.
2. Corra el procedimiento CREATE_CAPTURE que esta en el paquete DBMS_CAPTURE_ADM para crear un proceso de captura local. Por ejemplo:

```
BEGIN
DBMS_CAPTURE_ADM.CREATE_CAPTURE (
queue_name => 'strmadmin.streams_queue',
capture_name => 'strm02_capture',
rule_set_name => 'strmadmin.strm01_rule_set',
start_scn => NULL,
source_database => NULL,
first_scn => NULL);
END;
```

Al correr este método se ejecutan las siguientes acciones:

- Se crea un proceso de captura llamado strm02_capture.
- Asocia el proceso de captura con una cola llamada `streams_queue` que ya existe.
- Asocia el proceso de captura con un Set Positivo de Regla llamado strm01_rule_set, ya existente.

- Crea un proceso de captura que captura cambios locales ya que el parámetro `source_database` tiene valor NULL.
- Especifica que la propia Base de Datos determinara el start SCN y el first SCN para el proceso de captura.

Después de Configurado el Proceso de Captura.

Si se planea configurar procesos de propagación y aplicación que procesen registro de cambios lógico capturados por el nuevo proceso de captura, entonces realice la configuración en el siguiente orden:

1. Cree las colas que serán requeridas por los proceso de aplicación y propagación.
2. Cree todas las propagaciones que propagaran los LCRs capturados por el nuevo proceso de captura.
3. Cree todos los procesos de aplicación que quitarán de la cola los LCRs capturados por el nuevo proceso de captura.
4. Configure cada proceso de aplicación para que aplique LCRs capturados.
5. Cree instancias de las tablas, de las cuales el nuevo proceso de captura captará los cambios.
6. Inicie los procesos de aplicación que procesarán LCRs capturados por el nuevo proceso de captura.
7. Inicie el nuevo proceso de captura.

Configurar colas y propagación.

Crear una cola ANYDATA

Una cola ANYDATA puede almacenar mensajes de cualquier tipo. Cada proceso de captura y cada proceso de aplicación esta asociado con un a cola ANYDATA, así como cada proceso de propagación está asociado con una cola fuente ANYDATA y una cola destino ANYDATA.
La manera fácil de crear una cola ANYDATA es usar el procedimiento SET_UP_QUEUE del paquete DBMS_STREAMS_ADM. Este procedimiento permite configurar los siguientes parámetros:

- La tabla de la cola
- El nombre de la cola
- El usuario de la cola
- El comentario de la cola

Si la tabla especificada para cola no existe, entonces es creada. Si no se especifica ninguna cola entonces se le asigna una la por defecto streams_queue_table.Corra el procedimiento SET_UP_QUEUE que esta en el paquete DBMS_STREAMS_ADM para crear una cola ANYDATA.

```
BEGIN
DBMS_STREAMS_ADM.SET_UP_QUEUE (
queue_table => 'strmadmin.streams_queue_table',
queue_name => 'strmadmin.streams_queue',
queue_user => 'hr');
END;
```

Crear la propagación entre colas ANYDATA

La propagación envía el mensaje desde la cola fuente hacia la cola destino, se puede ayudar del Oracle Streams Advanced Queuing (AQ) para administrar esta propagación. Para crear una propagación entre dos colas ANYDATA se puede usar cualquiera de los siguientes procedimientos.

```
DBMS_STREAMS_ADM.ADD_SUBSET_PROPAGATION_RULES
DBMS_STREAMS_ADM.ADD_TABLE_PROPAGATION_RULES
DBMS_STREAMS_ADM.ADD_SCHEMA_PROPAGATION_RULES
DBMS_STREAMS_ADM.ADD_GLOBAL_PROPAGATION_RULES
DBMS_PROPAGATION_ADM.CREATE_PROPAGATION
```

Cada uno de estos procedimientos que pertenecen al paquete DBMS_STREAMS_ADM, crea una propagación con un nombre especificado, si este no existe ya, crea también un set de reglas positivo o negativo, si la propagación no presenta ningún set asociado, y adiciona reglas de tablas, reglas de esquema, reglas globales al set de reglas. El procedimiento crea una propagación pero no crea set de reglas, ni reglas. Aunque permite asociarle a la propagación un set ya existente. Es importante saber que todas las propagaciones se inician automáticamente después que son creadas.

Se deben cumplir las siguientes tareas antes de crear una propagación:

- Crear una cola fuente y una cola destino para la propagación. Para completar la tarea ambas colas deben ser de tipo ANYDATA.
- Crear un enlace de base de datos entre la base de datos que contiene la cola fuente y la base de datos que contiene la cola destino.

Ejemplo de Configuración de una Propagación usando DBMS_STREAMS_ADM.

El siguiente ejemplo corre el procedimiento ADD_TABLE_PROPAGATION_RULES que esta en el paquete DBMS_ STREAMS _ADM para crear una propagación:

```
BEGIN
DBMS_STREAMS_ADM.ADD_TABLE_PROPAGATION_RULES (
table_name => 'hr.departments',
streams_name => 'strm01_propagation',
source_queue_name => 'strmadmin.strm_a_queue',
destination_queue_name => 'strmadmin.strm_b_queue@dbs2.example.com',
include_dml => TRUE,
include_ddl => TRUE,
include_tagged_lcr => FALSE,
source_database => 'dbs1.example.com',
inclusion_rule => TRUE,
queue_to_queue => TRUE);
END;
```

Al correr este procedimiento se llevan a cabo las siguientes acciones:

- Se crea una propagación llamada strm01_propagation, solo si no existe ya.
- Especifica que los LCRs se propagarán desde strm_a_queue en la base de datos local hacia strm_b_queue en la base de datos dbs2.example.com.
- Especifica que la propagación usa el enlace de base de datos dbs2.example.com, por que el parámetro destination_queue_name contiene @dbs2.example.com.
- Como el parámetro inclusion_rule tiene valor, se crea set positivo de regla asociado a la propagación. El nombre del set de reglas es generad por el sistema y usa el contexto de evaluación SYS.STREAMS$_EVALUATION_CONTEXT.
- Crea dos reglas una que evalúa en true todos los cambios DML hechos a la tabla hr.departments y otra regla que evalúa en true todos los cambios DDL hechos a la misma tabla. Los nombres de las dos reglas son generados por el sistema.
- Crea un job de propagación para la propagación cola-cola.

Ejemplo de Configuración de una Propagación usando DBMS_PROPAGATION_ADM.

El siguiente ejemplo corre el procedimiento CREATE_PROPAGATION que esta en el paquete DBMS_ PROPAGATION_ADM para crear una propagación:

```
BEGIN
DBMS_PROPAGATION_ADM.CREATE_PROPAGATION (
```

```
propagation_name => 'strm02_propagation',
source_queue => 'strmadmin.strm03_queue',
destination_queue => 'strmadmin.strm04_queue',
destination_dblink => 'dbs2.example.com',
rule_set_name => 'strmadmin.strm01_rule_set',
queue_to_queue => TRUE);
END;
```

Al correr este procedimiento se llevan a caso las siguientes acciones:

- Se crea una propagación llamada strm02_propagation, solo si no existe ya.
- Especifica que los LCRs se propagarán desde strm03_queue en la base de datos local hacia strm04_queue en la base de datos dbs2.example.com.
- Especifica que la propagación usa el enlace de base de datos dbs2.example.com para propagar los mensajes.
- Asocia la propagación con un set de reglas llamado strm01_rule_set que ya existe.
- Crea un job de propagación para la propagación cola-cola.

Creación de un Proceso Aplicación

Se puede crear un proceso de aplicación con cualquiera de los siguientes procedimientos:

```
DBMS_STREAMS_ADM.ADD_TABLE_RULES
DBMS_STREAMS_ADM.ADD_SUBSET_RULES
DBMS_STREAMS_ADM.ADD_SCHEMA_RULES
DBMS_STREAMS_ADM.ADD_GLOBAL_RULES
DBMS_STREAMS_ADM.ADD_MESSAGE_RULE
DBMS_APPLY_ADM.CREATE_APPLY
```

Cada procedimiento del paquete crea un proceso de aplicación con un nombre especificado, si este no existe ya, crea también un set de reglas positivo o negativo, si la aplicación no presenta ningún set asociado y adiciona reglas de tablas, reglas de esquema, reglas globales al set de reglas.

El procedimiento del paquete crea un proceso de aplicación pero no crea set de reglas ni reglas para dicho proceso. Este paquete te permite asignar un set de reglas existente al proceso de aplicación. Además permite un número de opciones como manipuladores de aplicación, aplicaciones de usuario.

Un proceso de aplicación puede aplicar mensajes extrayéndolos de una cola buffered o de una cola persistente. Si un proceso de aplicación aplica LCRs capturados, entonces no podrá aplicar LCRs persistentes.

Ejemplo de Configuración de un Proceso de Aplicación usando DBMS_STREAMS_ADM.

El siguiente ejemplo corre el procedimiento ADD_SCHEMA_RULE que esta en el paquete DBMS_ STREAMS_ADM para crear un proceso de aplicación que aplica solo LCRs capturados:

```
BEGIN
DBMS_STREAMS_ADM.ADD_SCHEMA_RULES (
schema_name => 'hr',
streams_type => 'apply',
streams_name => 'strm01_apply',
queue_name => 'streams_queue',
include_dml => TRUE,
include_ddl => FALSE,
include_tagged_lcr => FALSE,
source_database => 'dbs1.example.com',
inclusion_rule => TRUE);
END;
```

Correr este método ejecuta las siguientes acciones:

- Se crea una propagación llamada strm01_propagation, solo si no existe ya.
- Asocia el proceso de aplicación con una cola ya existente llamada streams_queue.
- Crea un set positivo de reglas y lo asocia con el proceso de aplicación, siendo SYS.STREAMS$_EVALUATION_CONTEXT su contexto de evaluación. El nombre de este set de ragas es generado por el sistema.
- Crea una regla que evalúa en true todas la filas LCRs que se producen como resultado de un cambio DML a todos los objetos del esquema hr. El nombre de esta regla es generado por el sistema.

Ejemplo de Configuración de un Proceso de Aplicación usando DBMS_APPLY _ADM.

El siguiente ejemplo corre el procedimiento CREATE_APPLY que esta en el paquete DBMS_APPLY_ADM para crear un proceso de aplicación que aplica solo LCRs capturados:

```
BEGIN
DBMS_APPLY_ADM.CREATE_APPLY (
queue_name => 'strm03_queue',
apply_name => 'strm03_apply',
rule_set_name => 'strmadmin.strm03_rule_set',
message_handler => NULL,
ddl_handler => 'strmadmin.history_ddl',
apply_user => 'hr',
```

```
apply_database_link => NULL,
apply_tag => HEXTORAW ('5'),
apply_captured => TRUE,
precommit_handler => NULL,
negative_rule_set_name => NULL,
source_database => 'dbs1.example.com');
END;
```

Correr este método se ejecutan las siguientes acciones:

- Crea un proceso de aplicación llamado strm03_apply.
- Asocia el proceso de aplicación con una cola llamada strm03_queue que ya existe.
- Asocia el proceso de aplicación con un set de reglas llamado strm03_rule_set que ya existe.
- Especifica que el proceso de aplicación no usa manipulador de mensajes.
- Especifica que el manipulador DDL es un procedimiento PL/SQL llamado history_ddl que se encuentra en el esquema strmadmin.
- Especifica que el usuario que aplica los cambios es hr, no el que corre el procedimiento CREATE_APPLY (el administrador de Oracle Streams).
- Especifica que el proceso de aplicación aplica los cambios en la base de datos loca, ya que el parámetro apply_database_link esta en NULL.
- Especifica que el proceso de aplicación aplica los LCRs capturados y no los persistentes.
- Especifica que el proceso de aplicación no usa manipulador precommit.
- Especifica que el proceso de aplicación no usa set negativo de reglas.

2.7 Procedimientos para la Administración y Monitoreo de la tecnología Streams

Si se desarrolla un ambiente de replicación con esta tecnología, se podrá monitorear y ajustar en todo momento los valores y variables de todos los procesos y componentes de este ambiente. Streams, también consta de asistentes que ayudan en la administración de estos elementos, detectando cuellos de botellas, posibilidad de la ocurrencia de algún conflictos, etc. y además propone posibles soluciones a estos. A continuación de mostrarán las herramientas que esta tecnología nos brinda para llevar acabo estas tareas, que son de suma importancia para lograr un ambiente de replicación eficiente.

2.7.1 Estructuras de Datos y Procesos Streams

Estructuras de Datos Fundamentales para el proceso de captura:

```
STREAMS_CAPTURE (Cola)
AQ$_ STREAMS_QUEUE_TABLE_E (Cola de excepciones)
STREAMS_QUEUE_TABLE (Tabla de Cola)
AQ$_ STREAMS_QUEUE_TABLE_C, D, G, H, I, P, S, T (Tablas Auxiliares)
DBA_CAPTURE (Diccionario de captura)
DBA_QUEUE_TABLES (Diccionario de tablas de cola)
DBA_QUEUES (Diccionario de colas)
DBA_REGISTERED_ARCHIVED_LOG (Diccionario de archived log)
V$BUFFERED_QUEUES (Vista de desempeño para las colas)
V$STREAMS_CAPTURE (Vista de desempeño para la captura)
V$BUFFERED_PUBLISHERS (Vista de desempeño para el encolado de mensajes)
```

Estructuras de Datos Fundamentales para el proceso de propagación:

```
DBA_PROPAGATION (Diccionario de propagacion)
DBA_QUEUE_SCHEDULES (Diccionario de scheduler para las propagaciones)
V$PROPAGATION_SENDER (Vista de desempeño para el trasmisor)
V$PROPAGATION_RECEIVER (Vista de desempeño para el receptor)
V$BUFFERED_SUBSCRIBERS (Vista de desempeño para el desencolado de mensajes)
```

Estructuras de Datos Fundamentales para el proceso de aplicación:

```
DBA_APPLY (Diccionario de apply)
DBA_APPLY_ERROR (Diccionario de errores de apply)
DBA_APPLY_DML_HANDLERS (Diccionario de manipuladores)
V$STREAMS_APPLY_COORDINATOR (Vista de desempeño para el coordinador)
V$STREAMS_APPLY_READER (Vista de desempeño para el lector)
V$STREAMS_APPLY_SERVER (Vista de desempeño para el servidor)
V$BUFFERED_SUBSCRIBERS (Vista de desempeño para el desencolado de mensajes)
```

Estructuras de Datos Fundamentales para el Desempeño

Las estructuras de datos asociadas al desempeño constituyen elementos indispensables a monitorear en entornos complejos de replicación por Streams, de forma que puedan determinarse los retardos mas comunes en cada componente.

DBA_STREAMS_TP_COMPONENT (Componentes)

DBA_STREAMS_TP_COMPONENT_LINK (Flujo de mensajes)

DBA_STREAMS_TP_COMPONENT_STAT (Estadísticas por componentes)

DBA_STREAMS_TP_DATABASE (Bases de Datos con componentes Streams)

DBA_STREAMS_TP_PATH_BOTTLENECK (Embotellamientos)

DBA_STREAMS_TP_PATH_STAT (Estadísticas por Path)

2.7.2 Monitoriando Oracle Stremas

Estado de la Captura

```
select capture_name, queue_name, status, error_message from dba_capture;
```

CAPTURE_NAME	QUEUE_NAME	STATUS	ERROR_MESSAGE
STRMADMIN_CAPTURE_STREAMS	STREAMS_CAPTURE	**ENABLED**	

```
select capture_name, state, enqueue_time from v$streams_capture;
```

CAPTURE_NAME	STATE	ENQUEUE_TIME
STRMADMIN_CAPTURE_STREAMS	**CAPTURING CHANGES**	**3/24/2009 12:55:20 PM**

Estado de las Propagaciones

```
select propagation_name, status, error_message from dba_propagation;
```

PROPAGATION_NAME	STATUS	ERROR_MESSAGE
PROPAGATION_STREAMS_MTZ	**ENABLED**	
PROPAGATION_STREAMS_CAV	**ENABLED**	
PROPAGATION_STREAMS_STG	**ENABLED**	
PROPAGATION_STREAMS_CMG	**ENABLED**	
PROPAGATION_STREAMS_VCL	**ENABLED**	

```
select dblink, schedule_status, high_water_mark, acknowledgement from
v$propagation_sender;
```

DBLINK	SCHEDULE_STATUS	HIGH_WATER_MARK	ACKNOWLEDGEMENT

MTZ.DATA	**SCHEDULE ENABLED**	728202	728202
CAV.DATA	**SCHEDULE ENABLED**	728770	728767
STG.DATA	**SCHEDULE ENABLED**	728806	728801

```
select last_lcr_creation_time, last_lcr_propagation_time, dst_database_name
from v$propagation_sender;
```

LAST_LCR_CREATION_TIME	LAST_LCR_PROPAGATION_TIME	DST_DATABASE_NAME
3/24/2009 1:13:12 PM	3/24/2009 1:13:28 PM	HSTAT.DATA
3/24/2009 1:13:12 PM	3/24/2009 1:13:42 PM	CSTLC.DATA
3/24/2009 1:13:12 PM	3/24/2009 1:13:20 PM	CSQRO.DATA
3/24/2009 1:13:12 PM	3/24/2009 1:13:41 PM	CSANZ.DATA
3/24/2009 1:13:12 PM	3/24/2009 1:13:40 PM	CJTLC.DATA

Estado de los Apply Remotos

```
DECLARE
r_apply   dba_apply%ROWTYPE;
v_sqltext VARCHAR2 (255);
BEGIN
FOR C_DP IN (select destination_dblink from dba_propagation)
LOOP
    BEGIN
    v_sqltext:='select apply_name, status
                from  dba_apply@'||C_DP.destination_dblink;
    EXECUTE IMMEDIATE v_sqltext INTO r_apply.apply_name, r_apply.status;
    dbms_output.put_line (r_apply.apply_name||' '||r_apply.status);
    EXCEPTION WHEN OTHERS THEN
        dbms_output.put_line ('Error al comprobar '||C_DP.destination_dblink);
    END;
END LOOP;
END;
```

APPLY_NAME	STATUS
APPLY_CETGZ	**ENABLED**
APPLY_CJCUU	**ENABLED**
APPLY_CJMXL	**ENABLED**
APPLY_CEMTA	**ENABLED**
APPLY_CSTLC	**ENABLED**
APPLY_CETIR	**ENABLED**

Estado del Reader Remoto (Extrae mensajes de la cola)

```
DECLARE
r_apply   v$streams_apply_reader%ROWTYPE;
v_sqltext VARCHAR2 (255);
BEGIN
FOR C_DP IN (select destination_dblink from dba_propagation)
LOOP
    BEGIN
    v_sqltext:='select apply_name, state, dequeue_time from
    v$streams_apply_reader@'||C_DP.destination_dblink;
    EXECUTE IMMEDIATE v_sqltext INTO r_apply.apply_name, r_apply.state,
    r_apply.dequeue_time;
    dbms_output.put_line (r_apply.apply_name||' | '||r_apply.state||' |
    '||r_apply.dequeue_time);
    EXCEPTION WHEN OTHERS THEN
        dbms_output.put_line ('Error al comprobar '||C_DP.destination_dblink);
    END;
END LOOP;
END;
```

APPLY_NAME	STATE	DEQUEUE_TIME
APPLY_PROV_MTZ	**DEQUEUE MESSAGES**	**3/26/2009 10:08:36 AM**
APPLY_ PROV_STG	**DEQUEUE MESSAGES**	**3/26/2009 9:53:05 AM**

APPLY_ PROV_HOL **DEQUEUE MESSAGES** **3/26/2009 10:00:06 AM**

Estado del Coordinador Remoto (Coordina operaciones entre Reader y Server)

```
DECLARE
r_apply   v$streams_apply_coordinator%ROWTYPE;
v_sqltext VARCHAR2 (255);
BEGIN
FOR C_DP IN (select destination_dblink from dba_propagation)
LOOP
    BEGIN
    v_sqltext:='select apply_name, state, lwm_time from
    v$streams_apply_coordinator@'||C_DP.destination_dblink;
    EXECUTE IMMEDIATE v_sqltext INTO r_apply.apply_name, r_apply.state,
    r_apply.lwm_time;
    dbms_output.put_line (r_apply.apply_name||' | '||r_apply.state||' |
    '||r_apply.lwm_time);
    EXCEPTION WHEN OTHERS THEN
        dbms_output.put_line ('Error al comprobar '||C_DP.destination_dblink);
    END;
END LOOP;
END;
```

APPLY_NAME	STATE	LWM_TIME
APPLY_PROV_MTZ	**IDLE**	**3/26/2009 10:31:13 AM**
APPLY_ PROV_STG	**APPLYING**	**3/26/2009 9:02:58 AM**
APPLY_ PROV_HOL	**IDLE**	**3/26/2009 8:47:37 AM**

Estado del Server Remoto (Aplica mensajes)

```
DECLARE
r_apply   v$streams_apply_server%ROWTYPE;
```

```
v_sqltext VARCHAR2 (255);
BEGIN
FOR C_DP IN (select destination_dblink from dba_propagation)
LOOP
    BEGIN
    v_sqltext:='select apply_name, state, apply_time from
    v$streams_apply_server@'||C_DP.destination_dblink;
    EXECUTE IMMEDIATE v_sqltext INTO r_apply.apply_name, r_apply.state,
    r_apply.apply_time;
    dbms_output.put_line (r_apply.apply_name||' | '||r_apply.state||' |
    '||r_apply.apply_time);
    EXCEPTION WHEN OTHERS THEN
        dbms_output.put_line ('Error al comprobar '||C_DP.destination_dblink);
    END;
END LOOP;
END;
```

APPLY_NAME	STATE	APPLY_TIME
APPLY_HOTEL_CJCUU	**IDLE**	**3/26/2009 10:32:13 AM**
APPLY_HOTEL_CSANZ	**EXECUTE TRANSACTION**	**3/26/2009 9:03:58 AM**
APPLY_HOTEL_CESIL	**IDLE**	**3/26/2009 8:47:37 AM**

Estadísticas de Apply Local

```
select apply_name, total_received,total_applied, total_errors, (total_assigned
-(total_rollbacks + total_applied)) being_applied from
v$streams_apply_coordinator;
```

APPLY_NAME	TOTAL_RECEIVED	TOTAL_APPLIED	TOTAL_ERRORS	BEING_APPLIED
APPLY_HOTEL_CJCUU	**37**	**37**	**0**	0
APPLY_HOTEL_CSANZ	8	8	2	0
APPLY_HOTEL_CESIL	28	28	5	0

Desencolado y Retardos (LAG) en el proceso de Apply

```
Select subscriber_name, last_browsed_seq, last_dequeued_seq, message_lag from
v$buffered_subscribers
Where subscriber_type='SUBSCRIBER';
```

SUBSCRIBER_NAME	LAST_BROWSED_SEQ	LAST_DEQUEUED_SEQ	MESSAGE_LAG
APPLY_HOTEL_CJCUU	17048185	17048185	**0**
APPLY_HOTEL_CESAL	17062320	17062320	**0**
APPLY_HOTEL_CESLP	17053526	17053526	**0**

Desencolado y Retardos (LAG) en el proceso de Propagación

```
Select subscriber_address, last_browsed_seq, last_dequeued_seq, message_lag
from v$buffered_subscribers
Where subscriber_type='PROXY';
```

SUBSCRIBER_ADDRESS	LAST_BROWSED_SEQ	LAST_DEQUEUED_SEQ	MESSAGE_LAG
CETUL.DATA	0	0	738428
CJCUU.DATA	738414	738412	0
CESAL.DATA	738861	738861	0

Estado de las Colas

```
Select queue_name, (num_msgs -spill_msgs) mem_msg, spill_msgs, num_msgs,
queue_state from v$buffered_queues;
```

QUEUE_NAME	MEM_MSG	SPILL_MSGS	NUM_MSGS	QUEUE_STATE
STREAMS_APPLY	5	0	5	**NORMAL**
STREAMS_CAPTURE	93	55098	55191	**NORMAL**

Desempeño de Componentes

```
exec dbms_streams_advisor_adm.analyze_current_performance;
exec dbms_streams_advisor_adm.analyze_current_performance;
```

Se calculan las estadisticas entre una ejecucion y otra a partir de los snapshot tomados en cada una.

Estadísticas:

Select component_name, component_type, statistic_name, statistic_value, statistic_unit from DBA_STREAMS_TP_COMPONENT_STAT;

COMP_NAME	COMP_TYPE	STATISTIC_NAME	STATISTIC_VALUE	STATISTIC_UNIT
APPLY_HOTEL_CETLC	APPLY	IDLE	100	PERCENT
APPLY_CETLC	APPLY	IDLE	100	PERCENT
CAPTURE_CETLC	CAPTURE	IDLE	100	PERCENT
APPLY_STREAMS	APPLY	IDLE	99.66	PERCENT
CAPTURE_STREAMS	CAPTURE	IDLE	100	PERCENT
CAPTURE_STREAMS	CAPTURE	FLOW CONTROL	0	PERCENT
APPLY_HOTEL_CETLC	APPLY	FLOW CONTROL	0	PERCENT
CAPTURE_CETLC	CAPTURE	FLOW CONTROL	0	PERCENT
APPLY_HOTEL_CETLC	APPLY	FLOW CONTROL	0	PERCENT
CAPTURE_STREAMS	CAPTURE	FLOW CONTROL	0	PERCENT
APPLY_CETLC	APPLY	FLOW CONTROL	0	PERCENT
APPLY_HOTEL_CETLC	APPLY	FLOW CONTROL	0	PERCENT
APPLY_CETLC	APPLY	FLOW CONTROL	0	PERCENT
APPLY_CETLC	APPLY	db file sequential read	0.3333	PERCENT
CAPTURE_CETLC	CAPTURE	control file seq read	20.66	PERCENT
CAPTURE_CETLC	CAPTURE	CPU + Wait for CPU	7	PERCENT

Embotellamientos:

```
Select path_id, bottleneck_identified, advisor_run_time, advisor_run_reason
from DBA_STREAMS_TP_PATH_BOTTLENECK;
```

PATH_ID	BOTTLENECK_IDENTIFIED	ADVISOR_RUN_TIME	ADVISOR_RUN_REASON
2	NO	3/27/2009 12:18:00 PM	NO BOTTLENECK IDENTIFIED
1	NO	3/27/2009 12:18:00 PM	NO BOTTLENECK IDENTIFIED
4	NO	3/27/2009 12:18:00 PM	NO BOTTLENECK IDENTIFIED
3	NO	3/27/2009 12:18:00 PM	NO BOTTLENECK IDENTIFIED

2.7.3 Problemáticas que pueden surgir en el Proceso de Captura

Problemas de ejecución por escasos recursos de memoria, provocan el estado ABORTED de este componente:

Error ORA-04031: No pueden ser asignados n bytes de memoria compartida.
Causa: Se necesita más memoria compartida en SHARED_POOL
Acción: Reducir el uso de memoria compartida o incrementar la cantidad disponible asociada a los parámetros SHARED_POOL_RESERVED_SIZE, SHARED_POOL_SIZE, LARGE_POOL_SIZE.

Error ORA-23603: Encolado de STREAMS abortado por baja disponibilidad de SGA.
Causa: Operaciones de encolar han fallado porque la instancia dispone de poca SGA para operaciones de STREAMS.
Acción: Revisar estado de procesos consumidores o incrementar la cantidad disponible asociada al parámetro STREAMS_POOL_SIZE.

Problemas en el proceso de Startup provocan ciclos infinitos en estados INITIALIZING o DICTIONARY INITIALIZATION:

Causa: Redolog o Archived log no disponibles para el startup de la Captura.
Accion: Estados asociados a problemas con redolog o archived log. Deben revisarse las columnas asociadas a los SCN en DBA_REGISTERED_ARCHIVED_LOG y DBA_CAPTURE del diccionario de datos para determinar acciones a seguir en la localización de los archivos en disco.

Error ORA-01291: Redo log File o archived log no encontrado.
Causa: No todos los log files asociados a los SCN en proceso han sido encontrados.
Acción: Chequear v$logmnr_logs para determinar el rango de SCN faltante y localizar los logfiles a los cuales pertenecen.

Error interno de Oracle reportado por los monitores de Cola después de un Shutdown y Startup de la instancia.

Error ORA-00600:
Causa: Errores de funcionamiento interno de Oracle reportados por el monitor de colas QMON.

Acción: Reiniciar (stop y start) el proceso de Captura.

Proceso de captura detenido por mucho tiempo en los estados PAUSED BY FLOW CONTROL o ENQUEUING MESSAGE.

Causa: Acumulación excesiva de mensajes en la cola y embotellamientos en las descargas a disco.
Accion: Chequear si hay cantidad excesiva de mensajes descargados a disco (Spilled) y verificar en **v$buffered_subscribers (message_lag) si algunos de los sitios remotos no está consumiendo mensajes.**

2.7.4 Problemáticas que pueden surgir en el Proceso de Propagación

Proceso de propagación detenido por mucho tiempo ante el reporte de error: Enqueue rate too high, flow control enabled.

Error ORA-25307
Causa: Control de flujo habilitado por bajo nivel de procesamiento en el destino ante un elevado nivel de encolado.
Acción: Revise el estado del apply remoto y si es necesario reinicie la propagación con control de flujo habilitado.

Scheduler de propagación en estado BROKEN.

Causa: Scheduler Job deshabilitado por 16 errores consecutivos en el proceso de propagacion, generalmente asociados a problemas de conectividad.
Accion: Revise la conectividad y habilite nuevamente el scheduler job de la propagacion.

Oracle Enterprise Manager bloqueando procesos de Streams.

Causa: Bloqueos de los agentes del Enterprise Manager en las colas de Streams, han sido reportados en el BUG 5330663.
Accion: Deshabilite el Oracle Enterprise Manager, ejecute los procedimientos deseados y si es necesario rehabilitelo.

2.7.5 Problemáticas que pueden surgir en el Proceso de Apply

Comunmente, no se han reportado errores de funcionamiento para los apply, las tareas de administracion han estado encaminadas a reiniciar los mismos ante situaciones o estados derivados de otros componentes. Otros errores tratados han estado vinculados al proceso de manipulacion de transacciones con conflictos en la base de datos destino.

2.7.6 Procedimientos de Soluciones

Cada uno de los estados descritos en el tópico de Monitoreo para los elementos que participan en una replicación por Streams constituyen los estados normales mas comunes para la captura, propagación y apply. Existen otros estados por los que puede transitar esta tecnología que en un desempeño correcto de sus funciones ocurren rápidamente.

En este tópico se describen procedimientos elementales para restaurar componentes Streams en caso de fallos o funcionamiento irregular descritos en el tópico anterior.

Lo mas importante para la implementacion de estas soluciones es la identificacion del punto exacto donde se encuentra embotellado el ciclo de replicación por Streams a partir de las sentencias descritas en el tópico de Monitoreo. El proceso a seguir para determinar el embotellamiento debe contemplar cada uno de los puntos de transición de mensajes en el orden siguiente:

<< LOCAL	->	**REMOTO >>**
REDOLOG	->	CAPTURA
CAPTURA	->	COLA
COLA	->	PROPAGACION DE ENVIO
PROPAGACION DE ENVIO	->	**PROPAGACION DE RECEPCION**
PROPAGACION DE RECEPCION	->	**COLA**
COLA	->	**APLICADOR**
APLICADOR	->	**BASE DE DATOS**

Captura

Reiniciar captura:

```
exec dbms_capture_adm.stop_capture ('CAPTURE_NAME');
exec dbms_capture_adm.start_capture ('CAPTURE_NAME');
```

Reiniciar cola:

```
exec dbms_aqadm_stop_queue ('QUEUE_NAME', TRUE, TRUE,TRUE);
exec dbms_aqadm_start_queue ('QUEUE_NAME', TRUE,TRUE);
```

Reiniciar monitor de colas Streams:

```
ALTER SYSTEM SET aq_tm_processes=0 SCOPE = MEMORY;
ALTER SYSTEM SET aq_tm_processes=1 SCOPE = MEMORY;
```

Validar cola de Streams:

```
ALTER SYSTEM SET job_queue_processes=0 SCOPE = SPFILE;
ALTER SYSTEM SET aq_tm_processes=0 SCOPE = SPFILE;
SHUTDOWN IMMEDIATE
STARTUP RESTRICT
EXEC DBMS_AQADM_SYSCALLS.KWQA_3GL_VALIDATEQUEUE ('STRMADMIN','QUEUE_CAPTURE');
COMMIT
SHUTDOWN IMMEDIATE
STARTUP
ALTER SYSTEM SET job_queue_processes=# SCOPE = SPFILE;
ALTER SYSTEM SET aq_tm_processes=1 SCOPE = SPFILE;
```

Propagacion

Reiniciar Propagación:

```
exec dbms_propagation_adm.stop_propagation ('PROPAGATION_NAME');
exec dbms_propagation_adm.start_propagation ('PROPAGATION_NAME');
```

Replanificar Propagación:

```
exec dbms_propagation_adm.stop_propagation ('PROPAGATION_NAME');
```

```
exec dbms_aqadm.unschedule_propagation ('QUEUE_NAME', 'DESTINATION');
exec dbms_aqadm.schedule_propagation ('QUEUE_NAME', 'DESTINATION');
```

Habilitar y Deshabilitar scheduler de la Propagación:

```
exec dbms_aqadm.disable_propagation_schedule ('OWNER.QUEUE_NAME',
'DESTINATION');
exec dbms_aqadm.enable_propagation_schedule (' OWNER.QUEUE_NAME', 'DESTINATION
');
CONNECT SYS/PASSWORD AS SYSDBA;
EXEC DBMS_SCHEDULER.DISABLE ('NAME_SCHEDULER_JOBS');
EXEC DBMS_SCHEDULER.ENABLE (' NAME_SCHEDULER_JOBS ');
```

Apply

Reiniciar Apply:

```
exec dbms_apply_adm.stop_apply (' APPLY BCircul');
exec dbms_apply_adm.start_apply (' APPLY BCircul ');
```

Insertar Errores de Apply:

```
BEGIN
    FOR C_DA IN (SELECT local_transaction_id FROM DBA_APPLY_ERROR)
    LOOP
     BEGIN
        DBMS_APPLY_ADM.EXECUTE_ERROR (C_DA.local_transaction_id);
     EXCEPTION
     WHEN OTHERS THEN NULL;
     END;
    END LOOP;
END;
```

2.7.7 Procedimientos de Soporte y Mantenimiento.

La mayoría de los procedimientos descritos en el tópico constituyen métodos implementados hasta el momento en un paquete PL-SQL para realizar de forma fácil y segura operaciones de soporte y mantenimiento.

Testear estado de las Propagaciones y ejecutar Soluciones asociadas.

Procedimiento que comprueba el estado de las propagaciones y sesiones asociadas para restablecerlas al estado ENABLED o desbloquearlas ante ciclos infinitos de espera por eventos de RED.

```
CREATE PROCEDURE TEST_PROPAGATION IS
BEGIN
    /* REINICIAR SESSIONES CON TIEMPOS DE ESPERA SUPERIORES A LOS 900 SEGUNDOS POR
EVENTOS DE RED */
    FOR C_SP IN (SELECT distinct sid, serial#, destination
                    FROM v$session s, dba_queue_schedules qs
                    WHERE s.sid||', '||serial# =    qs.session_id
                        AND PROGRAM            LIKE '%( J%'
                        AND UPPER (ACTION)      =    'PROPAGATION SENDER'
                        AND UPPER (WAIT_CLASS) =    'NETWORK'
                        AND SECONDS_IN_WAIT    >    900)
    LOOP
        IF TEST_LINK (C_SP.destination) THEN
            EXECUTE      IMMEDIATE      'ALTER      SYSTEM      DISCONNECT      SESSION
'''||C_SP.sid||',''||C_SP.serial#||''' IMMEDIATE';
        END IF;
    END LOOP;
    /* HABILITAR PROPAGACIONES BROKEN O DISABLED */
    FOR C_PS IN (SELECT queue_schema, queue_name, dblink
                    FROM V$PROPAGATION_SENDER
                    WHERE schedule_status IN ('SCHEDULE DISABLED','ABORTED'))
    LOOP
        IF TEST_LINK (C_PS.dblink) THEN
DBMS_AQADM.ENABLE_PROPAGATION_SCHEDULE(C_PS.queue_schema||'.'||C_PS.queue_name,
C_PS.dblink);
        END IF;
    END LOOP;
END TEST_PROPAGATION;
```

Chequear Conectividad

Función que comprueba el ciclo completo de conectividad, pues para que retorne TRUE es necesario que haya conectividad TCP-IP, LISTENER OK y DATABASE OPEN.

```
CREATE FUNCTION TEST_LINK (p_destination VARCHAR2) RETURN BOOLEAN IS
state CHAR;
BEGIN
    EXECUTE IMMEDIATE 'SELECT 1 FROM DUAL@'||p_destination INTO state;
    RETURN TRUE;
  EXCEPTION
    WHEN OTHERS THEN RETURN FALSE;
END TEST_LINK;
```

Testear estado de los Apply y ejecutar Soluciones asociadas

```
CREATE PROCEDURE TEST_APPLY_LOCAL IS
  BEGIN
    /* REINICIAR SESSIONES CON TIEMPOS DE ESPERA SUPERIORES A LOS 900
       SEGUNDOS */
    FOR C_SA IN (SELECT distinct s.sid, s.serial#, ase.apply_name
                  FROM v$session s, v$streams_apply_server ase
                  WHERE   s.sid              =    ase.sid
                      AND PROGRAM            LIKE '%( A%'
                      AND UPPER (ACTION)      =    'STREAMS APPLY SERVER'
                      AND SECONDS_IN_WAIT    >    900)
    LOOP
         DBMS_APPLY_ADM.stop_apply (C_SA.apply_name);
         DBMS_APPLY_ADM.start_apply (C_SA.apply_name);
    END LOOP;
    /* HABILITAR APPLY */
    FOR C_DA IN (SELECT apply_name
                  FROM DBA_APPLY
                  WHERE STATUS IN ('DISABLED','ABORTED'))
    LOOP
           DBMS_APPLY_ADM.start_apply (C_DA.apply_name);
    END LOOP;
END TEST_APPLY_LOCAL;
```

Conclusiones del capítulo

En este capitulo hemos abarcado varios aspectos fundamentales para la asimilación y comprensión del Proceso de Replicación de Datos de Oracle Stream 11g, describiendo sus características, definiciones y ventajas. Además se realiza un estudio detallado de la arquitectura de Oracle Streams y de las principales temáticas para su configuración y posterior monitoreo. Haciendo énfasis en cada uno de los aspectos más importantes para el despliegue de forma optima de esta Tecnología.

CAPITULO 3: REPLICACIÓN CON STREAMS EN EL SISTEMA DE CIRCULADOS NACIONALES

Introducción

En el presente capitulo se abordarán temáticas relacionadas con el proceso actual de Replicación entre las bases de datos que soportan el Sistema de Circulados Nacionales, además se brindará una solución para optimizar dicho proceso, mediante una nueva propuesta de configuración para la Replicación de Datos. La nueva propuesta se implementará con la Tecnología Streams de Oracle 11g. Describiendo de forma detallada y organizada los procedimientos para la configuración de dicha tecnología.

3.1 Proceso de Replicación Actual en Circulados Nacionales

Las bases de datos provinciales del Sistema Circulados Nacionales constan de tres tablas: (CPersona) que contiene las personas circuladas, (WVEHCIR) que contiene los vehículos circulados y (Circulación) con los objetos circulados, pudiendo estar cada una de estas entidades, circuladas provincial o nacionalmente.

La base de datos que contiene todos los objetos, personas y vehículos circulados a nivel nacional esta formada por tres tablas similares a las tablas provinciales, llamadas: CIRN, CPERN, VEHNAC, las cuales agrupan las entidades de todas las provincias, circuladas con carácter nacional.

La replicación de datos del sistema de circulados nacionales esta formada por dos procesos de replicación. El primer proceso, propaga de las bases de datos provinciales, hacia la base de datos nacional, dicho proceso crea en la base de datos nacional una vista materializada de cada una de las tablas existentes en cada provincia, con los circulados que tengan categoría nacional. Los cambios que se realizan en las bases de datos provinciales se guardan en los logs de vistas materializadas, que son actualizados mediante la utilización de triggers internos de Oracle. Luego con la utilización de jobs de refrescamientos se actualizan las vistas materializadas, con los cambios guardados en los logs. Para cada vista materializada se utilizan tres triggers (inserts, updates, deletes) que son los encargados de actualizar las tablas de la base de datos nacional. Los jobs de refrescamiento se ejecutan en dependencia de la distancia que exista entre la base de datos nacional y provincial. Al culminar el proceso se obtendrán las

tres tablas de la base de datos nacional, con los datos actualizados de todas las personas objetos o vehículos, circulados con carácter nacional. (Ver figura 3.1)

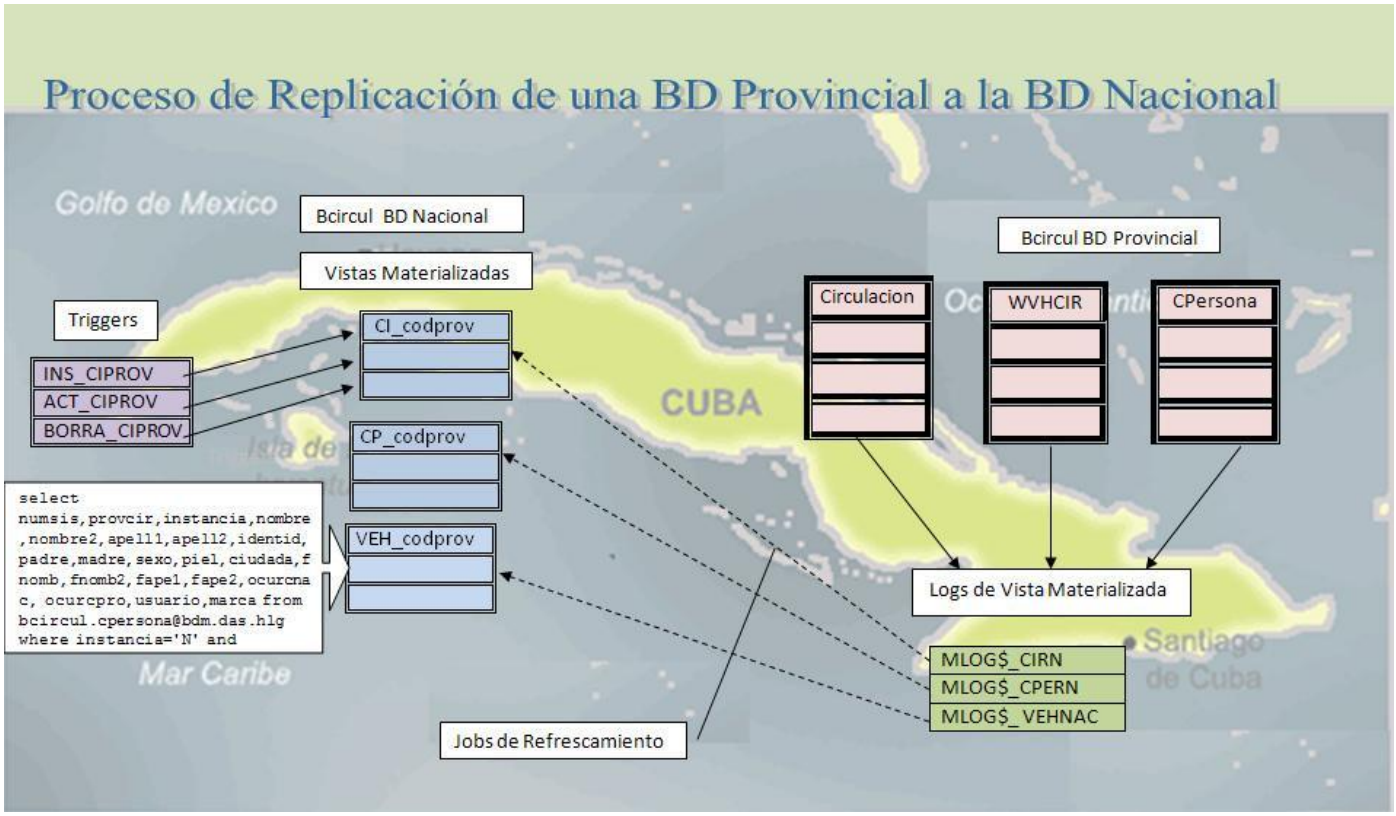

Figura 3.1

Con el fin de lograr una mayor autonomía de las bases de datos provinciales, disminuir el flujo de datos por la red y evitar las caídas del sistema, por fallos en la red, se implementa un segundo proceso de replicación, pero esta vez, de las bases de datos nacionales a las bases de datos provinciales. Este proceso crea una vista materializada en cada provincia, para cada tabla de la base de datos nacional, exceptuando los datos propios de la provincia; con estas vistas y la tabla de la provincia, se crea una vista que es utilizada para consultar, desde la provincia, los datos de los circulados nacionales.

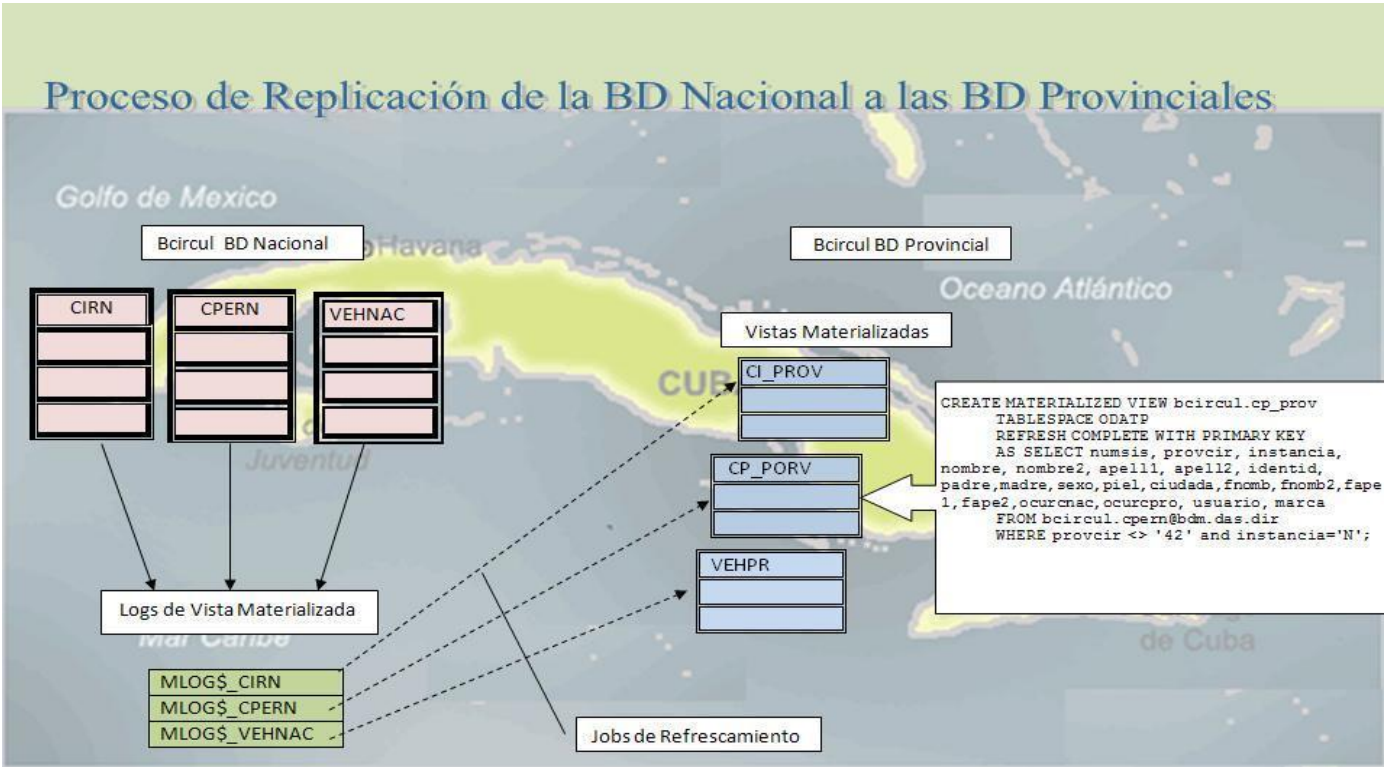

Figura 3.

3.2 Propuesta de Configuración para la replicación de Circulados Nacionales, usando Streams

El Sistema de Circulados Nacionales se desarrolló para usarse con la tecnología de Replicación Avanzada y a partir de la investigación realizada, se conoció de la complejidad de dicho sistema, al encontrar numerosas vistas, vistas materializadas y triggers, creadas para poder llevar a cabo la replicación con la utilización de esta tecnología, además de la influencia desde el punto de vista del desempeño que tienen sobre la base de datos estas estructuras. La propuesta de configuración de la tecnología Streams, para la replicación de datos en el Sistema de Circulados Nacionales, resolverá este problema con la utilización de tres tablas en la base de datos nacional y en cada una de las bases de datos provinciales; posteriormente para el acoplamiento de esta propuesta con dicho sistema, será necesaria la adición de

nuevos elementos, aspecto que no se abordará en la propuesta de Configuración para la replicación de Circulados Nacionales, usando Streams.

Con el objetivo de lograr una mejor asimilación de la tecnología que se implementa y profundizar en la configuración y arquitectura de cada uno de sus componentes, se realiza en algunos casos el proceso de configuración a partir de los paquetes PL/SQL, que permiten personalizar cada proceso y sus componentes, no se descarta la utilización del paquete `DBMS_STREAMS_ADM` en otros casos debido a la complejidad interna de algunos componentes como es el caso de las reglas.

La base de datos nacional quedará conformada por tres tablas: CIRN, CPERN, VEHNAC, las cuales agruparán las entidades de todas las provincias circuladas con carácter nacional; mientras que las bases de datos provinciales, contarán con las tablas Circulacion, WVHCIR y CPersona, guardando los datos de los circulados desde la provincia, e introduciéndose una nueva variación con respecto al sistema de circulados nacionales actual; lo que anteriormente se guardaba en las vistas materializadas (todos los circulados con carácter nacional), con esta propuesta se guardarán en las tablas de la base de datos provincial.

A continuación se describirá, utilizando la tecnología Oracle Streams, la configuración que tendrá el ambiente de replicación del sistema, especificando los procesos y detallando en cada uno de ellos, los procedimientos que serán utilizados para crearlos.

Se comenzará por la configuración de un proceso de captura, para cada tabla de la base de datos provincial, utilizando el procedimiento `ADD_SUBSET_RULES` del paquete `DBMS_STREAMS_ADM`; que guardará en la cola de captura, que fue creada con el procedimiento `SET_UP_QUEUE` del paquete `DBMS_STREAMS_ADM`, los cambios realizados a cualquiera de sus tres tablas, luego utilizando las reglas generadas por el sistema al crear la captura, se filtrarán solamente los cambios realizados a los circulados pertenecientes a la provincia y que tengan la categoría de Circulados Nacionales. Debido a la diferencia de los nombres de las tablas en ambas bases de datos (nacional y provincial) es necesario realizar una transformación al cambio captado, mediante la utilización de una regla basada en transformación, utilizando el procedimiento `RENAME_TABLE` del paquete `DBMS_STREAMS_ADM` cuyo objetivo sería lograr un entendimiento entre la base de datos fuente (base de datos provincial) y la base de datos destino (base de datos nacional). Posteriormente se creará un proceso de propagación, con el procedimiento

CREATE_PROPAGATION del paquete DBMS_PROPAGATION_ADM, que enviará todos los cambios almacenados en la cola de captura de la base de datos provincial, hacia su cola de aplicación correspondiente creada con el procedimiento SET_UP_QUEUE del paquete DBMS_STREAMS_ADM, en la base de datos nacional, para luego continuar con un proceso de aplicación, creado con el procedimiento CREATE_APPLY del paquete DBMS_APPLY_ADM; que extraerá de la cola de aplicación todos los cambios y los ejecutará en la base de datos nacional; existiendo en esta base de datos, una cola y un proceso aplicación para cada provincia. (Ver figura 3.3)

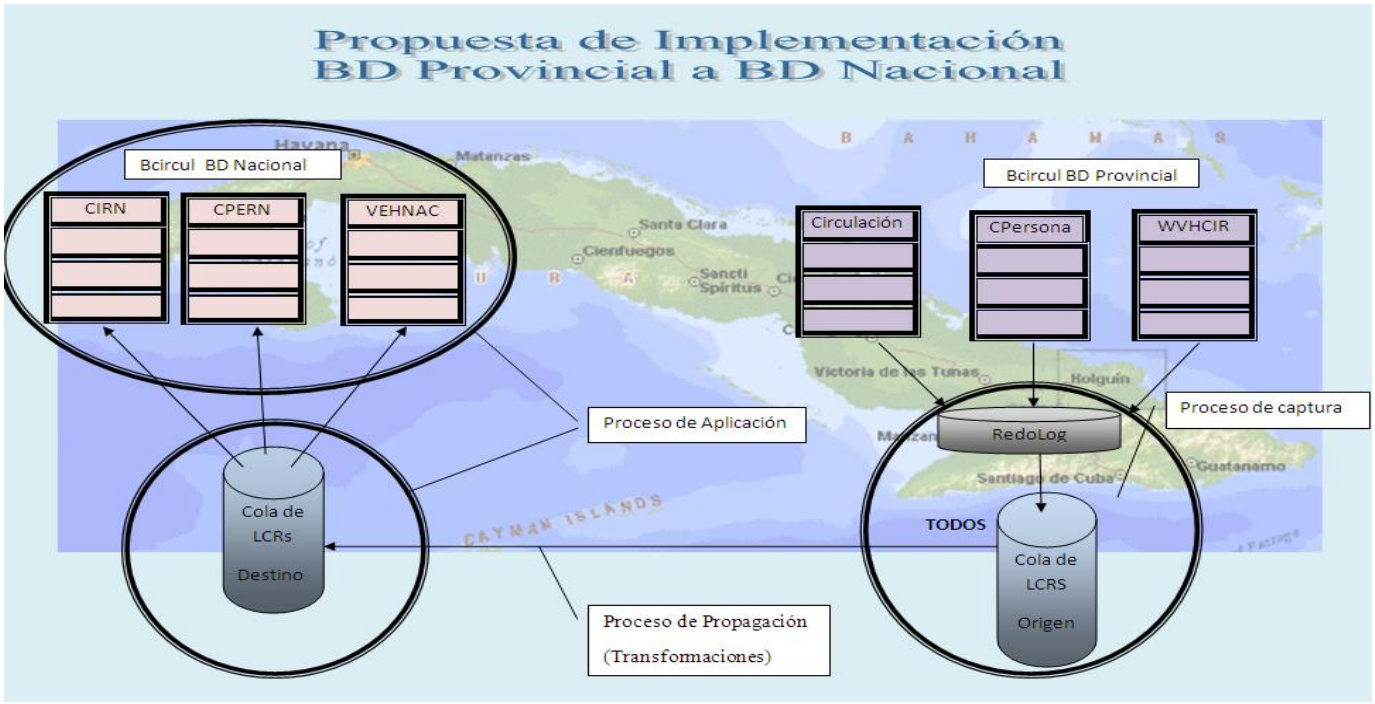

Figura 3.3

Después de haber culminado con el proceso de replicación de datos, desde la base de datos provincial hacia la base de datos nacional, se procederá a configurar un proceso de captura en la base de datos nacional, utilizando el procedimiento CREATE_CAPTURE del paquete DBMS_CAPTURE_ADM, que almacene en su cola de captura, creada con el procedimiento SET_UP_QUEUE del paquete DBMS_STREAMS_ADM, todos los cambios que fueron realizados en esta base de datos, por los procesos de aplicación anteriormente mencionados. A continuación se configurará un proceso de propagación para cada una de

las provincias involucradas, con el procedimiento ADD_SUBSET_PROPAGATION_RULES del paquete DBMS_STREAMS_ADM; enviando los cambios de la cola de captura de la base de datos nacional, a la cola de aplicación de las bases de datos provincial, creada con el procedimiento SET_UP_QUEUE del paquete DBMS_STREAMS_ADM. Este proceso propagará a cada provincia, los cambios correspondientes a todas las provincias, excluyendo los cambios pertenecientes a la provincia en cuestión; mediante la utilización de reglas generadas por con el sistema al crear la propagación. En este punto de la replicación, se llevará a cabo el mismo proceso de transformación que fue explicado con anterioridad en el proceso de propagación de la base de datos provincial hacia la base de datos nacional, con la utilización del procedimiento RENAME_TABLE del paquete DBMS_STREAMS_ADM. Finalmente se implementará un proceso de aplicación en cada provincia, utilizando el procedimiento CREATE_APPLY del paquete DBMS_APPLY_ADM, con el objetivo de extraer y aplicar los cambios que se encuentran en la cola de aplicación. (Ver figura 3.4)

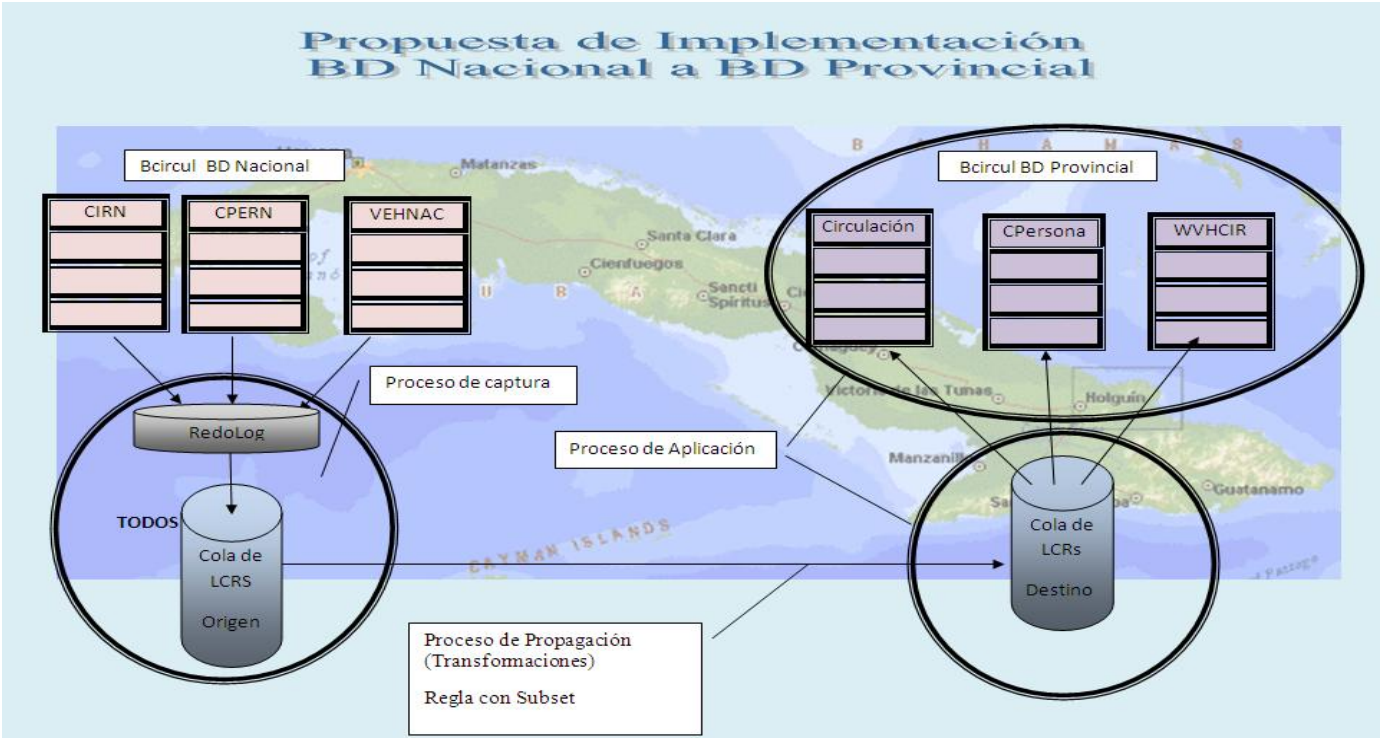

Figura 3.4

A cada uno de lo procesos anteriormente explicados (proceso de captura, propagación y aplicación), excepto los creados con el paquete DBMS_STREAMS_ADM, en el momento de su creación se les asignará un set de reglas, el cual se creará utilizando el procedimiento CREATE_RULE_SET del paquete DBMS_RULE_ADM, donde se agruparán todas las reglas creadas con el procedimiento CREATE_RULE del paquete DBMS_RULE_ADM. Estas reglas se adicionan a cada set con la utilización del procedimiento ADD_RULE del paquete DBMS_RULE_ADM y se crean con el fin de especificar el cambio deseado (DDL o DML) y las tablas de las cuales se quieren captar dichos cambios.

Con el objetivo de llevar a la práctica los conocimientos asimilados y la configuración propuesta, se crea un polígono de prueba compuesto por dos servidores; uno de ellos simulando el funcionamiento de la base de datos nacional y el otro simulando el funcionamiento de dos de las bases de datos provinciales, mediante la creación de dos instancias de Oracle. En cada uno de estos servidores, se instaló como sistema operativo, Linux Suse versión 10 y como Sistema Gestor de Base de Datos Oracle, en su versión 11g.

Los detalles para la implementación y configuración de todos los componentes de la tecnología asimilada sobre el polígono descrito en el párrafo anterior se describen como parte del anexo de configuración. Ver anexo 1

Conclusiones del capítulo

Después de conocer cómo lleva a cabo actualmente la Configuración para la replicación de Circulados Nacionales y la propuesta futura para la misma, se puede inferir que el procedimiento basado en la tecnología Streams aportará los elementos descritos durante el trabajo de tesis con respecto al ahorro de estructuras programáticas internas y por consiguiente la elevación del desempeño.

CONCLUSIONES

Con el desarrollo de este trabajo, se considera que se han cumplido todos los objetivos propuestos inicialmente. Por lo tanto se puede concluir expresando que:

- Se logró asimilar una parte significativa de la tecnología Stream, conociendo los beneficios que ofrece en cuanto a desempeño, configuración y administración.
- Se caracterizó la replicación actual de Circulados Nacionales, haciendo énfasis en las limitaciones de la Replicación Avanzada, con el objetivo de realizar una propuesta, que utilizando Stream, nos permita elevar el desempeño de las bases de datos distribuidas y contar con una tecnología flexible y segura para estos propósitos.
- Se llevó a la práctica la implementación de la tecnología asimilada, con la ayuda de un polígono de prueba que simula el sistema de Circulados Nacionales, evidenciándose de esta manera, los detalles a tener en cuenta a la hora de configurar y administrar un ambiente de replicación.

Finalmente, se considera que los aspectos tratados a lo largo del trabajo puedan ayudar a la construcción de futuros sistemas, donde sea necesaria la utilización de bases de datos distribuidas. Además se pretende que el documento sirva de apoyo a los ingenieros en sistema que necesiten utilizar Oracle y más específicamente Stream, como tecnología de replicación en bases de datos.

RECOMENDACIONES

Ya terminado este trabajo, teniendo en cuenta que se han cumplido todos los objetivos planteados, se recomienda:

- Poner en conocimiento de los desarrolladores las características de la Tecnología Streams, con el objetivo de utilizarla en los proyectos futuros de Bases de Datos Distribuidas, logrando el aprovechamiento, por parte de los desarrolladores, de todas las ventajas derivadas de su uso.
- Poner en conocimiento de los Ingenieros en Sistemas de la Institución los procedimientos de Administración de la Tecnología, con el objetivo de garantizar la disponibilidad y el desempeño de cada uno de sus componentes.
- Dar continuidad al estudio e implementación práctica de la Tecnología por parte del Departamento de Infraestructura Tecnológica, con el objetivo de asimilar otras posibilidades que han estado fuera del alcance de nuestro trabajo, pero que son de suma importancia para el dominio total de la misma, como pueden ser: la captura downstream (real-time y archived log), los manipuladores, las redes direccionadas, la captura sincrónica, contexto de evaluación, contexto de acción, entre otras.

BIBLIOGRAFÍA REFERENCIADA

[1]. Wikipedia. (30 de Noviembre de 2006). "Base de Datos" Revisado 21 de Marzo de 2009, desde http://es.wikipedia.org/wiki/Base_de_datos

[2]. Wikipedia. (14 de febrero de 2007). "DBMS " Revisado 22 de Marzo de 2009, desde http://es.wikipedia.org/wiki/DBMS

[3]. Wikipedia. (12 de junio de 2007). "Replicación " Revisado 30 de Marzo de 2009, desde http://es.wikipedia.org/wiki/Replicacion

[4]. Yusbel García Rodríguez y Jorge Luis Martínez." Estrategia para la Gestión Centralizada de las Bases de Datos Oracle del Data Center ministerial." MININT .Ciudad Habana. 2008. pág. 25-26.

[5]. Documentación Oficial de Oracle. (2007). "New to Oracle Database 11g" Revisado 8 de Abril 2009, desde http://www.oracle.com/technology/documentation/database.html

[6]. Documentación Oficial de Oracle. (2007) "Advanced Replication" Revisado 2 de Abril 2009, desde http://www.oracle.com/technology/documentation/database.html

[7]. Documentación Oficial de Oracle. (2007) "Concepts and Administration 11*g* Release 1 (11.1)" Revisado 30 de Abril 2009, desde http://www.oracle.com/technology/documentation/database.html

[8] Documentación Oficial de Oracle. (2007) "PL/SQL Packages and Types Reference 11*g* Release 1 (11.1.)" Revisado 11 de Mayo 2009, desde http://www.oracle.com/technology/documentation/database.html

BIBLIOGRAFIA

[Documentación Oficial de Oracle. (2007) "Replication Administrator's Guide 11g Release 1 (11.1)" Revisado 17 de Mayo 2009, desde http://www.oracle.com/technology/documentation/database.html

Documentación Oficial de Oracle. (2007) "Oracle Streams Performance" Revisado 22 de Mayo 2009, desde http://www.oracle.com/technology/documentation/database.html.

Documentación Oficial de Oracle. (2007) "Oracle Streams Replication" Revisado 29 de Abril 2009, desde http://www.oracle.com/technology/documentation/database.html.

Documentación Oficial de Oracle. (2007). "New to Oracle Database 11g" Revisado 12 de Mayo 2009, desde http://www.oracle.com/lang/es/database/index.html

Wikipedia. (12 de julio de 2007). "PL/SQL " Revisado 30 de Marzo de 2009, desde http://es.wikipedia.org/wiki/PL/SQL

Documentación Oficial de Oracle. (2007) "SQL, PL/SQL, and PL/SQL Packages" Revisado 22 de Abril 2009, desde http://www.oracle.com/technology/documentation/database.html.

ANEXO 1 Pasos para la configuración e implementación del Polígono de Prueba

Proceso de Replicación de CIRNAC hacia STRCAV y STRCHA

Preparando para configurar Streams

1- La base de datos tiene que encontrarse en modo archived log

2- Configurando parámetros de Streams

```
ALTER SYSTEM SET STREAMS_POOL_SIZE = 200M SCOPE = SPFILE;
ALTER SYSTEM SET UNDO_RETENTION = 7200 SCOPE = BOTH;
ALTER SYSTEM SET GLOBAL_NAMES = TRUE SCOPE = BOTH;
ALTER SYSTEM SET AQ_TM_PROCESSES = 1 SCOPE = BOTH;
ALTER SYSTEM SET JOB_QUEUE_PROCESSES = 100 SCOPE = BOTH;
```

3- Configurando los tnsnames de la base de datos fuente y de la base de datos destino

```
CIRNAC =
(DESCRIPTION =
        (ADDRESS = (PROTOCOL = TCP) (HOST = 96.34.1.91) (PORT = 1521))
        (CONNECT_DATA = (SERVER = DEDICATED) (SERVICE_NAME = CIRNAC)))

STRCAV =
(DESCRIPTION =
        (ADDRESS = (PROTOCOL = TCP) (HOST = 96.34.1.92) (PORT = 1521))
        (CONNECT_DATA = (SERVER = DEDICATED) (SERVICE_NAME = STRCAV)))

STRCHA =
(DESCRIPTION =
        (ADDRESS = (PROTOCOL = TCP) (HOST = 96.34.1.92) (PORT = 1521))
        (CONNECT_DATA = (SERVER = DEDICATED) (SERVICE_NAME = STRCHA)))
```

4- Preparando condiciones para funcionamiento del Log Miner

```
CREATE TABLESPACE LOGMINER DATAFILE
'+DATOS' SIZE 1M AUTOEXTEND ON NEXT 1M MAXSIZE UNLIMITED
EXTENT MANAGEMENT LOCAL AUTOALLOCATE
BLOCKSIZE 8K
SEGMENT SPACE MANAGEMENT AUTO
FLASHBACK ON;

EXEC DBMS_LOGMNR_D.SET_TABLESPACE ('LOGMINER');
```

Creando el tablespace para el usuario STRMADMIN en CIRNAC

```
CREATE TABLESPACE STRMADMIN DATAFILE
'+DATOS' SIZE 1M AUTOEXTEND ON NEXT 1M MAXSIZE UNLIMITED
EXTENT MANAGEMENT LOCAL AUTOALLOCATE
BLOCKSIZE 8K
SEGMENT SPACE MANAGEMENT AUTO
FLASHBACK ON;
```

5- Creando el usuario STRMADMIN en CIRNAC

```
CREATE USER STRMADMIN IDENTIFIED BY strmadmin
DEFAULT TABLESPACE STRMADMIN
TEMPORARY TABLESPACE TEMPTBS;
```

6- Asignando al usuario STRMADMIN todos los privilegios que necesita

```
GRANT DBA, IMP_FULL_DATABASE, EXP_FULL_DATABASE TO STRMADMIN;

BEGIN
    DBMS_STREAMS_AUTH.GRANT_ADMIN_PRIVILEGE (
    grantee => 'STRMADMIN',
    grant_privileges => true);
END;
```

7- Conectándose a CIRNAC con STRMADMIN

```
connect STRMADMIN/strmadmin;
```

8- Creando dblink desde CIRNAC hacia STRCAV y STRCHA

```
CREATE DATABASE LINK STRCAV.DAS.CITES connect to “STRMADMIN” identified by strmadmin
using 'STRCAV';

CREATE DATABASE LINK STRCHA.DAS.CITES connect to “STRMADMIN” identified by strmadmin
using 'STRCHA';
```

Configurando Streams

1- Creando Cola

Componente: Captura
Sitio: CIRNAC
Usuario: STRMADMIN
Descripción: Donde el proceso de Captura almacena los LCRs para evaluarlos
Código:
```
BEGIN
    DBMS_STREAMS_ADM.SET_UP_QUEUE (
    queue_table => 'STREAMS_CAPT_CIRNAC_QT',
    queue_name => 'STREAMS_CAPT_CIRNAC_Q',
    queue_user => 'STRMADMIN');
END;
```

2- Asignando permiso

Componente: Captura
Sitio: CIRNAC
Usuario: STRMADMIN
Descripción: Permite visualizar la tabla de la cola desde OEM
Código:
```
grant select on AQ$STREAMS_CAPT_CIRNAC_QT to dbsnmp;
COMMIT;
```

3- Creando Set de Reglas

Componente: Captura
Sitio: CIRNAC
Usuario: STRMADMIN
Descripción: Donde se agrupan todas las reglas del proceso de Captura.
Código:
```
BEGIN
    DBMS_RULE_ADM.CREATE_RULE_SET (
    rule_set_name => 'RULE_SET_CAPT_CIRNAC',
    evaluation_context => 'SYS.STREAMS$_EVALUATION_CONTEXT');
END;
```

4- Creando reglas

Componente: Captura
Sitio: CIRNAC
Usuario: STRMADMIN
Descripción: Regla que evalúa en TRUE todos los cambios en la tabla CIRN, CPERN y VHENAC.
Código:
```
BEGIN
```

```
        DBMS_RULE_ADM.CREATE_RULE (
        rule_name=>'RULE_CAPT_CIRNAC_CIRN',
        condition=>':dml.get_object_owner () = "BCIRCUL"
        AND: dml.get_object_name () = "CIRN"
        AND: dml.is_null_tag () = "Y" ‘);
    END;

    BEGIN
        DBMS_RULE_ADM.CREATE_RULE (
        rule_name=>'RULE_CAPT_CIRNAC_CPERN',
        condition=>':dml.get_object_owner () = "BCIRCUL"
        AND: dml.get_object_name () = "CPERN"
        AND: dml.is_null_tag () = "Y" ‘);
    END;

    BEGIN
        DBMS_RULE_ADM.CREATE_RULE (
        rule_name=>'RULE_CAPT_CIRNAC_VEHNAC',
        condition=>':dml.get_object_owner () = "BCIRCUL"
        AND: dml.get_object_name () = "VEHNAC"
        AND: dml.is_null_tag () = "Y" ');
    END;
```

5- Adicionando Reglas al Set Reglas

Componente: Captura
Sitio: CIRNAC
Usuario: STRMADMIN
Descripción: Se adjunta al Set de Reglas de Captura la Regla para la tabla CIRN, CPERN y VHENAC.
Código:

```
    BEGIN
        DBMS_RULE_ADM.ADD_RULE (
        rule_name => 'RULE_CAPT_CIRNAC_CIRN',
        rule_set_name => 'RULE_SET_CAPT_CIRNAC');
    END;

    BEGIN
        DBMS_RULE_ADM.ADD_RULE (
        rule_name => 'RULE_CAPT_CIRNAC_CPERN',
        rule_set_name => 'RULE_SET_CAPT_CIRNAC');
    END;

    BEGIN
        DBMS_RULE_ADM.ADD_RULE (
        rule_name => 'RULE_CAPT_CIRNAC_VEHNAC',
```

```
    rule_set_name => 'RULE_SET_CAPT_CIRNAC');
END;
```

6- Creando Captura

Componente: Captura
Sitio: CIRNAC
Usuario: STRMADMIN
Descripción: Se crea el proceso de Captura, especificando la Cola y el Set de Regla.
Código:

```
BEGIN
    DBMS_CAPTURE_ADM.CREATE_CAPTURE (
    queue_name => 'STREAMS_CAPT_CIRNAC_Q',
    capture_name => 'STREAMS_CAPT_CIRNAC',
    rule_set_name => 'RULE_SET_CAPT_CIRNAC',
    start_scn => NULL,
    source_database => NULL,
    first_scn => NULL);
END;
```

7- Creando Colas

Componente: Aplicación
Sitio: STRCAV
Usuario: STRMADMIN
Descripción: Donde el proceso de Propagación coloca los LCRs para que sean evaluados por las reglas del proceso de Aplicación. Se deben crear las colas de la base de datos destinos antes de crear la Propagación.
Código:

```
BEGIN
    DBMS_STREAMS_ADM.SET_UP_QUEUE (
    queue_table => 'STREAMS_APPLY_CIRNAC_QT',
    queue_name => 'STREAMS_APPLY_CIRNAC_Q',
    queue_user => 'STRMADMIN');
END;
```

Componente: Aplicación
Sitio: STRCHA
Usuario: STRMADMIN
Descripción: Donde el proceso de Propagación coloca los LCRs para que sean evaluados por las reglas del proceso de Aplicación. Se deben crear las colas de la base de datos destinos antes de crear la Propagación.
Código:

```
BEGIN
    DBMS_STREAMS_ADM.SET_UP_QUEUE (
    queue_table => 'STREAMS_APPLY_CIRNAC_QT',
```

```
    queue_name => 'STREAMS_APPLY_CIRNAC_Q',
    queue_user => 'STRMADMIN');
END;
```

8- Asignando permisos

Componente: Captura
Sitio: STRCAV
Usuario: STRMADMIN
Descripción: Permite visualizar la tabla de la Cola desde OEM
Código:
```
grant select on AQ$STREAMS_APPLY_CIRNAC_QT to dbsnmp;
COMMIT;
```

Componente: Captura
Sitio: STRCHA
Usuario: STRMADMIN
Descripción: Permite visualizar la tabla de la Cola desde OEM.
Código:
```
grant select on AQ$STREAMS_APPLY_CIRNAC_QT to dbsnmp;
COMMIT;
```

9- Creando Propagaciones

Componente: Propagación
Sitio: CIRNAC
Usuario: STRMADMIN
Descripción: Se crea un Proceso de Propagación de CIRNAC hacia STRCHA llamado STREAMS_PROP_STRCHA que propaga solo las entidades circuladas con carácter nacional, que no pertenezcan a STRCHA (provincia Ciudad de La Habana) y se encuentren en la tabla CIRN. A cada regla generada (Insert, Update, Delete), se le asocia una regla basada en transformación, ya que las tablas correspondientes en el sitio destino presentan nombres diferentes.
Código:
```
DECLARE
  v_insert_rule VARCHAR2 (255);
  v_update_rule VARCHAR2 (255);
  v_delete_rule VARCHAR2 (255);
BEGIN
  DBMS_STREAMS_ADM.ADD_SUBSET_PROPAGATION_RULES (
  table_name => 'BCIRCUL.CIRN',
  dml_condition => 'INSTA = "N" AND PRCIR! = "21"
  AND F_SIN_EFECTO IS NULL',
  streams_name => 'STREAMS_PROP_STRCHA',
  source_queue_name => 'STREAMS_CAPT_CIRNAC_Q',
  destination_queue_name => 'STREAMS_APPLY_CIRNAC_Q
  @STRCHA.DAS.CITES',
```

```
    include_tagged_lcr => TRUE,
    source_database => 'CIRNAC.DAS.CITES',
    insert_rule_name => v_insert_rule,
    update_rule_name => v_update_rule,
    delete_rule_name => v_delete_rule,
    queue_to_queue => TRUE);

    DBMS_STREAMS_ADM.RENAME_TABLE (
    rule_name => v_insert_rule,
    from_table_name => 'BCIRCUL.CIRN',
    to_table_name => 'BCIRCUL.CI_PROV');

    DBMS_STREAMS_ADM.RENAME_TABLE (
    rule_name => v_update_rule,
    from_table_name => 'BCIRCUL.CIRN',
    to_table_name => 'BCIRCUL.CI_PROV');

    DBMS_STREAMS_ADM.RENAME_TABLE (
    rule_name => v_delete_rule,
    from_table_name => 'BCIRCUL.CIRN',
    to_table_name => 'BCIRCUL.CI_PROV');
END;
```

Componente: Propagación
Sitio: CIRNAC
Usuario: STRMADMIN
Descripción: En este caso, como la propagación ya esta creada, solamente se crean las reglas para propagar solamente las entidades circuladas con carácter nacional, que no pertenezcan a esta provincia (STRCHA) y se encuentren en la tabla CPERN. A cada regla generada (Insert, Update, Delete), se le asocia una regla basada en transformación, ya que las tablas correspondientes en el sitio destino presentan nombres diferentes.
Código:

```
DECLARE
    v_insert_rule VARCHAR2 (255);
    v_update_rule VARCHAR2 (255);
    v_delete_rule VARCHAR2 (255);
BEGIN
    DBMS_STREAMS_ADM.ADD_SUBSET_PROPAGATION_RULES (
    table_name => 'BCIRCUL.CPERN',
    dml_condition => 'INSTA = "N" AND PRCIR! = "21"
    AND F_SIN_EFECTO IS NULL',
    streams_name => 'STREAMS_PROP_STRCHA',
    source_queue_name => 'STREAMS_CAPT_CIRNAC_Q',
    destination_queue_name => 'STREAMS_APPLY_CIRNAC_Q
    @STRCHA.DAS.CITES',
    include_tagged_lcr => TRUE,
    source_database => 'CIRNAC.DAS.CITES',
```

```
    insert_rule_name => v_insert_rule,
    update_rule_name => v_update_rule,
    delete_rule_name => v_delete_rule,
    queue_to_queue => TRUE);

    DBMS_STREAMS_ADM.RENAME_TABLE (
    rule_name => v_insert_rule,
    from_table_name => 'BCIRCUL.CPERN',
    to_table_name => 'BCIRCUL.CP_PROV');

    DBMS_STREAMS_ADM.RENAME_TABLE (
    rule_name => v_update_rule,
    from_table_name => 'BCIRCUL.CPERN',
    to_table_name => 'BCIRCUL.CP_PROV');

    DBMS_STREAMS_ADM.RENAME_TABLE (
    rule_name => v_delete_rule,
    from_table_name => 'BCIRCUL.CPERN',
    to_table_name => 'BCIRCUL.CP_PROV');
END;
```

Componente: Propagación

Sitio: CIRNAC

Usuario: STRMADMIN

Descripción: En este caso, como la propagación ya esta creada, solamente se crean las reglas para propagar solamente las entidades circuladas con carácter nacional, que no pertenezcan a esta provincia (STRCHA) y se encuentren en la tabla VEHNAC. A cada regla generada (Insert, Update, Delete), se le asocia una regla basada en transformación, ya que las tablas correspondientes en el sitio destino presentan nombres diferentes.

Código:

```
DECLARE
  v_insert_rule VARCHAR2 (255);
  v_update_rule VARCHAR2 (255);
  v_delete_rule VARCHAR2 (255);
BEGIN
  DBMS_STREAMS_ADM.ADD_SUBSET_PROPAGATION_RULES (
  table_name => 'BCIRCUL.VEHNAC',
  dml_condition => 'INSTA = "N" AND PRCIR! = "21"
  AND F_SIN_EFECTO IS NULL',
  streams_name => 'STREAMS_PROP_STRCHA',
  source_queue_name => 'STREAMS_CAPT_CIRNAC_Q',
  destination_queue_name => 'STREAMS_APPLY_CIRNAC_Q
  @STRCHA.DAS.CITES',
  include_tagged_lcr => TRUE,
  source_database => 'CIRNAC.DAS.CITES',
  insert_rule_name => v_insert_rule,
  update_rule_name => v_update_rule,
```

```
    delete_rule_name => v_delete_rule,
    queue_to_queue => TRUE);

    DBMS_STREAMS_ADM.RENAME_TABLE (
    rule_name => v_insert_rule,
    from_table_name => 'BCIRCUL.VEHNAC',
    to_table_name => 'BCIRCUL.VEHPR');

    DBMS_STREAMS_ADM.RENAME_TABLE (
    rule_name => v_update_rule,
    from_table_name => 'BCIRCUL.VEHNAC',
    to_table_name => 'BCIRCUL.VEHPR');

    DBMS_STREAMS_ADM.RENAME_TABLE (
    rule_name => v_delete_rule,
    from_table_name => 'BCIRCUL.VEHNAC',
    to_table_name => 'BCIRCUL.VEHPR');
  END;
```

Componente: Propagación

Sitio: CIRNAC

Usuario: STRMADMIN

Descripción: Se crea un Proceso de Propagación de CIRNAC hacia STRCAV llamado STREAMS_PROP_STRCAV que propaga solo las entidades circuladas con carácter nacional, que no pertenezcan a STRCAV (provincia Ciego de Ávila) y se encuentren en la tabla CIRN. A cada regla generada (Insert, Update, Delete), se le asocia una regla basada en transformación, ya que las tablas correspondientes en el sitio destino presentan nombres diferentes.

Código:

```
DECLARE
   v_insert_rule VARCHAR2 (255);
   v_update_rule VARCHAR2 (255);
   v_delete_rule VARCHAR2 (255);
 BEGIN
   DBMS_STREAMS_ADM.ADD_SUBSET_PROPAGATION_RULES (
   table_name => 'BCIRCUL.CIRN',
   dml_condition => 'INSTA = "N" AND PRCIR! = "22"
   AND F_SIN_EFECTO IS NULL',
   streams_name => 'STREAMS_PROP_STRCAV',
   source_queue_name => 'STREAMS_CAPT_CIRNAC_Q',
   destination_queue_name => 'STREAMS_APPLY_CIRNAC_Q
   @STRCAV.DAS.CITES',
   include_tagged_lcr => TRUE,
   source_database => 'CIRNAC.DAS.CITES',
   insert_rule_name => v_insert_rule,
   update_rule_name => v_update_rule,
   delete_rule_name => v_delete_rule,
   queue_to_queue => TRUE);
```

```
  DBMS_STREAMS_ADM.RENAME_TABLE (
  rule_name => v_insert_rule,
  from_table_name => 'BCIRCUL.CIRN',
  to_table_name => 'BCIRCUL.CI_PROV');

  DBMS_STREAMS_ADM.RENAME_TABLE (
  rule_name => v_update_rule,
  from_table_name => 'BCIRCUL.CIRN',
  to_table_name => 'BCIRCUL.CI_PROV');

  DBMS_STREAMS_ADM.RENAME_TABLE (
  rule_name => v_delete_rule,
  from_table_name => 'BCIRCUL.CIRN',
  to_table_name => 'BCIRCUL.CI_PROV');
END;
```

Componente: Propagación

Sitio: CIRNAC

Usuario: STRMADMIN

Descripción: En este caso, como la propagación ya esta creada, solo se crean las reglas para propagar solamente las entidades circuladas con carácter nacional, que no pertenezcan a esta provincia (STRCAV) y se encuentren en la tabla CPERN. A cada regla generada (Insert, Update, Delete), se le asocia una regla basada en transformación, ya que las tablas correspondientes en el sitio destino presentan nombres diferentes.

Código:

```
DECLARE
  v_insert_rule VARCHAR2 (255);
  v_update_rule VARCHAR2 (255);
  v_delete_rule VARCHAR2 (255);
BEGIN
  DBMS_STREAMS_ADM.ADD_SUBSET_PROPAGATION_RULES (
  table_name => 'BCIRCUL.CPERN',
  dml_condition => 'INSTA = "N" AND PRCIR! = "22"
  AND F_SIN_EFECTO IS NULL',
  streams_name => 'STREAMS_PROP_STRCAV',
  source_queue_name => 'STREAMS_CAPT_CIRNAC_Q',
  destination_queue_name => 'STREAMS_APPLY_CIRNAC_Q
  @STRCAV.DAS.CITES',
  include_tagged_lcr => TRUE,
  source_database => 'CIRNAC.DAS.CITES',
  insert_rule_name => v_insert_rule,
  update_rule_name => v_update_rule,
  delete_rule_name => v_delete_rule,
  queue_to_queue => TRUE);

  DBMS_STREAMS_ADM.RENAME_TABLE (
  rule_name => v_insert_rule,
```

```
    from_table_name => 'BCIRCUL.CPERN',
    to_table_name => 'BCIRCUL.CP_PROV');

    DBMS_STREAMS_ADM.RENAME_TABLE (
    rule_name => v_update_rule,
    from_table_name => 'BCIRCUL.CPERN',
    to_table_name => 'BCIRCUL.CP_PROV');

    DBMS_STREAMS_ADM.RENAME_TABLE (
    rule_name => v_delete_rule,
    from_table_name => 'BCIRCUL.CPERN',
    to_table_name => 'BCIRCUL.CP_PROV');
END;
```

Componente: Propagación

Sitio: CIRNAC

Usuario: STRMADMIN

Descripción: En este caso, como la propagación ya esta creada, solamente se crean las reglas para propagar solamente las entidades circuladas con carácter nacional, que no pertenezcan a esta provincia (STRCAV) y se encuentren en la tabla VEHNAC. A cada regla generada (Insert, Update, Delete), se le asocia una regla basada en transformación, ya que las tablas correspondientes en el sitio destino presentan nombres diferentes.

Código:

```
DECLARE
  v_insert_rule VARCHAR2 (255);
  v_update_rule VARCHAR2 (255);
  v_delete_rule VARCHAR2 (255);
BEGIN
  DBMS_STREAMS_ADM.ADD_SUBSET_PROPAGATION_RULES (
  table_name => 'BCIRCUL.VEHNAC',
  dml_condition => 'INSTA = "N" AND PRCIR! = "22"
  AND F_SIN_EFECTO IS NULL',
  streams_name => 'STREAMS_PROP_STRCAV',
  source_queue_name => 'STREAMS_CAPT_CIRNAC_Q',
  destination_queue_name => 'STREAMS_APPLY_CIRNAC_Q
  @STRCAV.DAS.CITES',
  include_tagged_lcr => TRUE,
  source_database => 'CIRNAC.DAS.CITES',
  insert_rule_name => v_insert_rule,
  update_rule_name => v_update_rule,
  delete_rule_name => v_delete_rule,
  queue_to_queue => TRUE);

  DBMS_STREAMS_ADM.RENAME_TABLE (
  rule_name => v_insert_rule,
  from_table_name => 'BCIRCUL.VEHNAC',
  to_table_name => 'BCIRCUL.VEHPR');
```

```
    DBMS_STREAMS_ADM.RENAME_TABLE (
    rule_name => v_update_rule,
    from_table_name => 'BCIRCUL.VEHNAC',
    to_table_name => 'BCIRCUL.VEHPR');
    DBMS_STREAMS_ADM.RENAME_TABLE (
    rule_name => v_delete_rule,
    from_table_name => 'BCIRCUL.VEHNAC',
    to_table_name => 'BCIRCUL.VEHPR');
END;
```

10- Creando Set de Reglas

Componente: Aplicación
Sitio: STRCHA
Usuario: STRMADMIN
Descripción: Donde se agrupan todas las Reglas del Proceso de Aplicación.
Código:

```
BEGIN
    DBMS_RULE_ADM.CREATE_RULE_SET (
    rule_set_name => 'RULE_SET_APPLY_CIRNAC',
    evaluation_context => 'SYS.STREAMS$_EVALUATION_CONTEXT');
 END;
```

11- Creando Reglas

Componente: Aplicación
Sitio: STRCHA
Usuario: STRMADMIN
Descripción: Regla que evalúa en TRUE todos los mensajes y los aplica en la tabla CI_PROV, CP_PROV y VEHPR.
Código:

```
BEGIN
    DBMS_RULE_ADM.CREATE_RULE (
    rule_name=>'RULE_APPLY_CIRNAC_CIRN',
    condition=>':dml.get_object_owner() = ''BCIRCUL''
    AND: dml.get_object_name () = ''CI_PROV''
    AND: dml.is_null_tag () = ''Y'' ‘);
 END;

BEGIN
    DBMS_RULE_ADM.CREATE_RULE (
    rule_name=>'RULE_APPLY_CIRNAC_CPERN',
    condition=>':dml.get_object_owner() = ''BCIRCUL''
     AND: dml.get_object_name () = ''CP_PROV''
    AND: dml.is_null_tag () = ''Y'' ‘);
```

```
END;

BEGIN
    DBMS_RULE_ADM.CREATE_RULE (
    rule_name=>'RULE_APPLY_CIRNAC_VEHNAC',
    condition=>':dml.get_object_owner() = ''BCIRCUL''
    AND: dml.get_object_name () = ''VEHPR''
    AND: dml.is_null_tag () = ''Y'' ‘);
END;
```

12- Adicionando Reglas al Set de Reglas

Componente: Aplicación
Sitio: STRCHA
Usuario: STRMADMIN
Descripción: Se adjunta al Set de Reglas de Aplicación la Regla para la tabla CI_PROV, CP_PROV y VEHPR.

Código:

```
BEGIN
    DBMS_RULE_ADM.ADD_RULE (
    rule_name => 'RULE_APPLY_CIRNAC_CIRN',
    rule_set_name => 'RULE_SET_APPLY_CIRNAC');
END;

BEGIN
    DBMS_RULE_ADM.ADD_RULE (
    rule_name => 'RULE_APPLY_CIRNAC_CPERN',
    rule_set_name => 'RULE_SET_APPLY_CIRNAC');
END;

BEGIN
    DBMS_RULE_ADM.ADD_RULE (
    rule_name => 'RULE_APPLY_CIRNAC_VEHNAC',
    rule_set_name => 'RULE_SET_APPLY_CIRNAC');
END;
```

13- Creando Aplicación

Componente: Aplicación
Sitio: STRCHA
Usuario: STRMADMIN
Descripción: Se crea el Proceso de Aplicación asignándole el Set de Reglas
Código:

```
BEGIN
    DBMS_APPLY_ADM.CREATE_APPLY (
```

```
    queue_name => 'STREAMS_APPLY_CIRNAC_Q',
    apply_name => 'STREAMS_APPLY_CIRNAC',
    apply_captured => TRUE);
END;
```

14- Iniciando Proceso de Aplicación

Componente: Aplicación
Sitio: STRCHA
Usuario: STRMADMIN
Descripción: Se inicia el Proceso de Aplicación creado anteriormente
Código:

```
BEGIN
    dbms_apply_adm.start_apply ('STREAMS_APPLY_CIRNAC');
END;
```

15- Creando Set de Reglas

Componente: Aplicación
Sitio: STRCAV
Usuario: STRMADMIN
Descripción: Donde se agrupan todas las Reglas del Proceso de Aplicación.
Código:

```
BEGIN
    DBMS_RULE_ADM.CREATE_RULE_SET (
    rule_set_name => 'RULE_SET_APPLY_CIRNAC',
    evaluation_context => 'SYS.STREAMS$_EVALUATION_CONTEXT');
END;
```

16- Creando Reglas

Componente: Aplicación
Sitio: STRCAV
Usuario: STRMADMIN
Descripción: Regla que evalúa en TRUE todos los mensajes y los aplica en la tabla CI_PROV, CP_PROV y VEHPR.
Código:

```
BEGIN
    DBMS_RULE_ADM.CREATE_RULE (
    rule_name=>'RULE_APPLY_CIRNAC_CIRN',
    condition=>':dml.get_object_owner() = ''BCIRCUL''
    AND: dml.get_object_name () = ''CI_PROV''
    AND: dml.is_null_tag () = ''Y'' ‘);
END;
```

```
BEGIN
    DBMS_RULE_ADM.CREATE_RULE (
    rule_name=>'RULE_APPLY_CIRNAC_CPERN',
    condition=>':dml.get_object_owner() = ''BCIRCUL''
     AND: dml.get_object_name () = ''CP_PROV''
    AND: dml.is_null_tag () = ''Y'' ‘);
END;

BEGIN
    DBMS_RULE_ADM.CREATE_RULE (
    rule_name=>'RULE_APPLY_CIRNAC_VEHNAC',
    condition=>':dml.get_object_owner() = ''BCIRCUL''
    AND: dml.get_object_name () = ''VEHPR''
    AND: dml.is_null_tag () = ''Y'' ‘);
END;
```

17- Adicionando Reglas al Set de Reglas

Componente: Aplicación
Sitio: STRCAV
Usuario: STRMADMIN
Descripción: Se adjunta al Set de Reglas de Aplicación la Regla para la tabla CI_PROV, CP_PROV y VEHPR.
Código:

```
BEGIN
    DBMS_RULE_ADM.ADD_RULE (
    rule_name => 'RULE_APPLY_CIRNAC_CIRN',
    rule_set_name => 'RULE_SET_APPLY_CIRNAC');
END;

BEGIN
    DBMS_RULE_ADM.ADD_RULE (
    rule_name => 'RULE_APPLY_CIRNAC_CPERN',
    rule_set_name => 'RULE_SET_APPLY_CIRNAC');
 END;

BEGIN
    DBMS_RULE_ADM.ADD_RULE (
    rule_name => 'RULE_APPLY_CIRNAC_VEHNAC',
    rule_set_name => 'RULE_SET_APPLY_CIRNAC');
END;
```

18- Creando Aplicación

Componente: Aplicación

Sitio: STRCAV
Usuario: STRMADMIN
Descripción: Se crea el Proceso de Aplicación asignándole el Set de Reglas.
Código:
```
BEGIN
    DBMS_APPLY_ADM.CREATE_APPLY (
    queue_name => 'STREAMS_APPLY_CIRNAC_Q',
    apply_name => 'STREAMS_APPLY_CIRNAC',
    apply_captured => TRUE);
END;
```

19- Iniciando Proceso de Aplicación

Componente: Aplicación
Sitio: STRCAV
Usuario: STRMADMIN
Descripción: Se inicia el Proceso de Aplicación creado anteriormente.
Código:
```
BEGIN
    dbms_apply_adm.start_apply ('STREAMS_APPLY_CIRNAC');
END;
```

20- Preparando tablas para instanciar

Componente: Captura
Sitio: CIRNAC
Usuario: STRMADMIN
Descripción: Se prepara la tabla CIRN, CPERN y VEHNAC de la base de datos fuente para instanciar y se le adiciona el supplemental logging a todos los campos de la tabla.
Código:
```
BEGIN
    DBMS_CAPTURE_ADM.PREPARE_TABLE_INSTANTIATION (
    table_name =>'BCIRCUL.CIRN',
    supplemental_logging => 'all');
END;

BEGIN
    DBMS_CAPTURE_ADM.PREPARE_TABLE_INSTANTIATION (
    table_name =>'BCIRCUL.CPERN',
    supplemental_logging => 'all');
END;

BEGIN
    DBMS_CAPTURE_ADM.PREPARE_TABLE_INSTANTIATION (
    table_name =>'BCIRCUL.VEHNAC',
     supplemental_logging => 'all');
END;
```

21- Instanciando tablas

Componente: Captura
Sitio: CIRNAC
Usuario: STRMADMIN
Descripción: Se instancia la tabla CIRN, CPERN y VEHNAC para replicar hacia la base de datos STRCHA.
Código:

```
DECLARE
    source_scn number;
BEGIN
    source_scn:= dbms_flashback.get_system_change_number();
    dbms_apply_adm.set_table_instantiation_scn@strcha.das.cites
    (source_object_name => 'BCIRCUL.CIRN',
      source_database_name => 'CIRNAC.DAS.CITES',
      instantiation_scn => source_scn);
END;

DECLARE
    source_scn number;
BEGIN
    source_scn:= dbms_flashback.get_system_change_number();
    dbms_apply_adm.set_table_instantiation_scn@strcha.das.cites
    (source_object_name => 'BCIRCUL.CPERN',
     source_database_name => 'CIRNAC.DAS.CITES',
     instantiation_scn => source_scn);
END;

DECLARE
    source_scn number;
BEGIN
    source_scn:= dbms_flashback.get_system_change_number();
    dbms_apply_adm.set_table_instantiation_scn@strcha.das.cites
    (source_object_name => 'BCIRCUL.VEHNAC',
      source_database_name => 'CIRNAC.DAS.CITES',
      instantiation_scn => source_scn);
END;
```

Componente: Captura
Sitio: CIRNAC
Usuario: STRMADMIN

Descripción: Se instancia la tabla CIRN, CPERN y VEHNAC para replicar hacia la base de datos STRCAV.

Código:

```
DECLARE
    source_scn number;
BEGIN
    source_scn:= dbms_flashback.get_system_change_number();
    dbms_apply_adm.set_table_instantiation_scn@strcav.das.cites
    (source_object_name => 'BCIRCUL.CIRN',
     source_database_name => 'CIRNAC.DAS.CITES',
     instantiation_scn => source_scn);
END;
```

```
DECLARE
    source_scn number;
BEGIN
    source_scn:= dbms_flashback.get_system_change_number();
    dbms_apply_adm.set_table_instantiation_scn@strcav.das.cites
    (source_object_name => 'BCIRCUL.CPERN',
     source_database_name => 'CIRNAC.DAS.CITES',
     instantiation_scn => source_scn);
END;
```

```
DECLARE
    source_scn number;
BEGIN
    source_scn:= dbms_flashback.get_system_change_number();
    dbms_apply_adm.set_table_instantiation_scn@strcav.das.cites
    (source_object_name => 'BCIRCUL.VEHNAC',
     source_database_name => 'CIRNAC.DAS.CITES',
     instantiation_scn => source_scn);

END;
```

22- Iniciando Proceso de Captura

Componente: Captura
Sitio: CIRNAC
Usuario: STRMADMIN
Descripción: Se inicia el Proceso de Captura.
Código:

```
BEGIN
    dbms_capture_adm.start_capture ('STREAMS_CAPT_CIRNAC');
END;
```

Proceso de Replicación de STRCHA y STRCAV hacia CIRNAC

Preparando para configurar Streams

1- La base de datos tiene que encontrarse en modo archived log

2- Configurando parámetros de Streams

```
ALTER SYSTEM SET STREAMS_POOL_SIZE = 200M SCOPE = SPFILE;
ALTER SYSTEM SET UNDO_RETENTION = 7200 SCOPE = BOTH;
ALTER SYSTEM SET GLOBAL_NAMES = TRUE SCOPE = BOTH;
ALTER SYSTEM SET AQ_TM_PROCESSES = 1 SCOPE = BOTH;
ALTER SYSTEM SET JOB_QUEUE_PROCESSES = 100 SCOPE = BOTH;
```

3- Configurando los tnsnames de la base de datos fuente y de la base de datos destino

```
CIRNAC =
(DESCRIPTION =
        (ADDRESS = (PROTOCOL = TCP) (HOST = 96.34.1.91) (PORT = 1521))
        (CONNECT_DATA = (SERVER = DEDICATED) (SERVICE_NAME = CIRNAC)))

STRCAV =
(DESCRIPTION =
        (ADDRESS = (PROTOCOL = TCP) (HOST = 96.34.1.92) (PORT = 1521))
        (CONNECT_DATA = (SERVER = DEDICATED) (SERVICE_NAME = STRCAV)))

STRCHA =
(DESCRIPTION =
        (ADDRESS = (PROTOCOL = TCP) (HOST = 96.34.1.92) (PORT = 1521))
        (CONNECT_DATA = (SERVER = DEDICATED) (SERVICE_NAME = STRCHA)))
```

4- **Preparando condiciones para funcionamiento del Log Miner**

```
CREATE TABLESPACE LOGMINER DATAFILE
'+DATOS' SIZE 1M AUTOEXTEND ON NEXT 1M MAXSIZE UNLIMITED
EXTENT MANAGEMENT LOCAL AUTOALLOCATE
BLOCKSIZE 8K
SEGMENT SPACE MANAGEMENT AUTO
FLASHBACK ON;

EXEC DBMS_LOGMNR_D.SET_TABLESPACE ('LOGMINER');
```

5- **Creando el tablespace para el usuario STRMADMIN en STRCAV y STRCHA**

```
CREATE TABLESPACE STRMADMIN DATAFILE
```

```
'+DATOS' SIZE 1M AUTOEXTEND ON NEXT 1M MAXSIZE UNLIMITED
EXTENT MANAGEMENT LOCAL AUTOALLOCATE
BLOCKSIZE 8K
SEGMENT SPACE MANAGEMENT AUTO
FLASHBACK ON;
```

6- Creando el usuario STRMADMIN en STRCAV y STRCHA

```
CREATE USER STRMADMIN IDENTIFIED BY strmadmin
DEFAULT TABLESPACE STRMADMIN
TEMPORARY TABLESPACE TEMPTBS;
```

7- Asignando al usuario STRMADMIN todos los privilegios que necesita

```
GRANT DBA, IMP_FULL_DATABASE, EXP_FULL_DATABASE TO STRMADMIN;

BEGIN
    DBMS_STREAMS_AUTH.GRANT_ADMIN_PRIVILEGE (
    grantee => 'STRMADMIN',
    grant_privileges => true);
END;
```

8- Creando dblink desde STRCAV y STRCHA hacia CIRNAC

```
CREATE DATABASE LINK CIRNAC.DAS.CITES connect to “STRMADMIN” identified by strmadmin
using 'CIRNAC';
```

Configurando Streams

1- Creando Cola

Componente: Captura
Sitio: STRCHA
Usuario: STRMADMIN
Descripción: Donde el proceso de Captura almacena los LCRs para evaluarlos
Código:

```
BEGIN
    DBMS_STREAMS_ADM.SET_UP_QUEUE (
    queue_table => 'STREAMS_CAPT_STRCHA_QT',
    queue_name => 'STREAMS_CAPT_STRCHA_Q',
    queue_user  => 'STRMADMIN');
END;
```

2- Asignando permiso

Componente: Captura

Sitio: STRCHA
Usuario: STRMADMIN
Descripción: Permite visualizar la tabla de la cola desde OEM
Código:
```
grant select on AQ$STREAMS_CAPT_CIRNAC_QT to dbsnmp;
COMMIT;
```

3- Creando Capturas

Componente: Captura
Sitio: STRCHA
Usuario: STRMADMIN
Descripción: Al ejecutar este procedimiento se crea un Proceso de Captura llamado STREAMS_CAPT_STRCHA que capta los cambios realizados a la tabla CI_PROV que las entidades circuladas sean de carácter nacional y pertenezcan la provincia Ciudad Habana. A cada regla generada (Insert, Update, Delete), se le asocia una regla basada en transformación, ya que las tablas correspondientes en el sitio destino presentan nombres diferentes.
Código:
```
DECLARE
    v_insert_rule VARCHAR2 (255);
    v_update_rule VARCHAR2 (255);
    v_delete_rule VARCHAR2 (255);
BEGIN
    DBMS_STREAMS_ADM.ADD_SUBSET_RULES (
    table_name => 'BCIRCUL.CI_PROV',
    dml_condition => 'INSTA = "N" AND PRCIR = "21"
    AND F_SIN_EFECTO IS NULL',
    streams_type => 'CAPTURE',
    streams_name => 'STREAMS_CAPT_STRCHA',
    queue_name => 'STREAMS_CAPT_STRCHA_Q',
    include_tagged_lcr => TRUE,
    source_database => 'STRCHA.DAS.CITES',
    insert_rule_name => v_insert_rule,
    update_rule_name => v_update_rule,
    delete_rule_name => v_delete_rule);
    DBMS_STREAMS_ADM.RENAME_TABLE (
    rule_name => v_insert_rule,
    from_table_name => 'BCIRCUL.CI_PROV',
    to_table_name => 'BCIRCUL.CIRN');
    DBMS_STREAMS_ADM.RENAME_TABLE (
    rule_name => v_update_rule,
     from_table_name => 'BCIRCUL.CI_PROV',
     to_table_name => 'BCIRCUL.CIRN');
    DBMS_STREAMS_ADM.RENAME_TABLE (
    rule_name => v_delete_rule,
    from_table_name => 'BCIRCUL.CI_PROV',
```

```
    to_table_name => 'BCIRCUL.CIRN');
END;
```

Componente: Captura
Sitio: STRCHA
Usuario: STRMADMIN
Descripción: En este caso, como el Proceso de Captura ya esta creado, solo se crean las reglas para captar los cambios realizados a la tabla CP_PROV que las entidades circuladas sean de carácter nacional y pertenezcan la provincia Ciudad Habana. A cada regla generada (Insert, Update, Delete), se le asocia una regla basada en transformación, ya que las tablas correspondientes en el sitio destino presentan nombres diferentes.
Código:

```
DECLARE
    v_insert_rule VARCHAR2 (255);
    v_update_rule VARCHAR2 (255);
    v_delete_rule VARCHAR2 (255);
BEGIN
    DBMS_STREAMS_ADM.ADD_SUBSET_RULES (
    table_name => 'BCIRCUL.CP_PROV',
    dml_condition => 'INSTANCIA = "N" AND PROVCIR = "21" ',
    streams_type => 'CAPTURE',
    streams_name => 'STREAMS_CAPT_STRCHA',
    queue_name => 'STREAMS_CAPT_STRCHA_Q',
    include_tagged_lcr => TRUE,
    source_database => 'STRCHA.DAS.CITES',
    insert_rule_name => v_insert_rule,
    update_rule_name => v_update_rule,
    delete_rule_name => v_delete_rule);
    DBMS_STREAMS_ADM.RENAME_TABLE (
    rule_name => v_insert_rule,
    from_table_name => 'BCIRCUL.CP_PROV',
    to_table_name => 'BCIRCUL.CPERN');
    DBMS_STREAMS_ADM.RENAME_TABLE (
    rule_name => v_update_rule,
    from_table_name => 'BCIRCUL.CP_PROV',
    to_table_name => 'BCIRCUL.CPERN');
    DBMS_STREAMS_ADM.RENAME_TABLE (
    rule_name => v_delete_rule,
    from_table_name => 'BCIRCUL.CP_PROV',
    to_table_name => 'BCIRCUL.CPERN');
END;
```

Componente: Captura
Sitio: STRCHA
Usuario: STRMADMIN

Descripción: En este caso, como el Proceso de Captura ya esta creado, solo se crean las reglas para captar los cambios realizados a la tabla VEHPR que las entidades circuladas sean de carácter nacional y pertenezcan la provincia Ciudad Habana. A cada regla generada (Insert, Update, Delete), se le asocia una regla basada en transformación, ya que las tablas correspondientes en el sitio destino presentan nombres diferentes.

Código:

```
DECLARE
    v_insert_rule VARCHAR2 (255);
    v_update_rule VARCHAR2 (255);
    v_delete_rule VARCHAR2 (255);
BEGIN
    DBMS_STREAMS_ADM.ADD_SUBSET_RULES (
    table_name => 'BCIRCUL.VEHPR',
    dml_condition => 'PRMCIR = "21" ',
    streams_type => 'CAPTURE',
    streams_name => 'STREAMS_CAPT_STRCHA',
    queue_name => 'STREAMS_CAPT_STRCHA_Q',
    include_tagged_lcr => TRUE,
    source_database => 'STRCHA.DAS.CITES',
    insert_rule_name => v_insert_rule,
    update_rule_name => v_update_rule,
    delete_rule_name => v_delete_rule);
    DBMS_STREAMS_ADM.RENAME_TABLE (
    rule_name => v_insert_rule,
    from_table_name => 'BCIRCUL.VEHPR',
    to_table_name => 'BCIRCUL.VEHNAC');
    DBMS_STREAMS_ADM.RENAME_TABLE (
    rule_name => v_update_rule,
    from_table_name => 'BCIRCUL.VEHPR',
    to_table_name => 'BCIRCUL.VEHNAC');
    DBMS_STREAMS_ADM.RENAME_TABLE (
    rule_name => v_delete_rule,
    from_table_name => 'BCIRCUL.VEHPR',
    to_table_name => 'BCIRCUL.VEHNAC');
 END;
```

4- Creando Cola

Componente: Captura
Sitio: STRCAV
Usuario: STRMADMIN
Descripción: Donde el proceso de Captura almacena los LCRs para evaluarlos
Código:

```
BEGIN
    DBMS_STREAMS_ADM.SET_UP_QUEUE (
    queue_table => 'STREAMS_CAPT_STRCAV_QT',
```

```
    queue_name => 'STREAMS_CAPT_STRCAV_Q',
    queue_user  => 'STRMADMIN');
END;
```

5- Asignando permiso

Componente: Captura
Sitio: STRCAV
Usuario: STRMADMIN
Descripción: Permite visualizar la tabla de la cola desde OEM
Código:

```
grant select on AQ$STREAMS_CAPT_CIRNAC_QT to dbsnmp;
COMMIT;
```

6- Creando Capturas

Componente: Captura
Sitio: STRCAV
Usuario: STRMADMIN
Descripción: Al ejecutar este procedimiento se crea un Proceso de Captura llamado STREAMS_CAPT_STRCAV que capta los cambios realizados a la tabla CI_PROV que las entidades circuladas sean de carácter nacional y pertenezcan la provincia Ciego de Ávila. A cada regla generada (Insert, Update, Delete), se le asocia una regla basada en transformación, ya que las tablas correspondientes en el sitio destino presentan nombres diferentes.
Código:

```
DECLARE
    v_insert_rule VARCHAR2 (255);
    v_update_rule VARCHAR2 (255);
    v_delete_rule VARCHAR2 (255);
BEGIN
    DBMS_STREAMS_ADM.ADD_SUBSET_RULES (
    table_name => 'BCIRCUL.CI_PROV',
    dml_condition => 'INSTA = "N" AND PRCIR = "21"
    AND F_SIN_EFECTO IS NULL',
    streams_type => 'CAPTURE',
    streams_name => 'STREAMS_CAPT_STRCHA',
    queue_name => 'STREAMS_CAPT_STRCHA_Q',
    include_tagged_lcr => TRUE,
    source_database => 'STRCHA.DAS.CITES',
    insert_rule_name => v_insert_rule,
    update_rule_name => v_update_rule,
    delete_rule_name => v_delete_rule);
    DBMS_STREAMS_ADM.RENAME_TABLE (
    rule_name => v_insert_rule,
    from_table_name => 'BCIRCUL.CI_PROV',
    to_table_name => 'BCIRCUL.CIRN');
```

```
    DBMS_STREAMS_ADM.RENAME_TABLE (
    rule_name => v_update_rule,
     from_table_name => 'BCIRCUL.CI_PROV',
     to_table_name => 'BCIRCUL.CIRN');
    DBMS_STREAMS_ADM.RENAME_TABLE (
    rule_name => v_delete_rule,
    from_table_name => 'BCIRCUL.CI_PROV',
    to_table_name => 'BCIRCUL.CIRN');
  END;
```

Componente: Captura

Sitio: STRCAV

Usuario: STRMADMIN

Descripción: En este caso, como el Proceso de Captura ya esta creado, solo se crean las reglas para captar los cambios realizados a la tabla CP_PROV que las entidades circuladas sean de carácter nacional y pertenezcan la provincia Ciego de Ávila. A cada regla generada (Insert, Update, Delete), se le asocia una regla basada en transformación, ya que las tablas correspondientes en el sitio destino presentan nombres diferentes.

Código:

```
DECLARE
    v_insert_rule VARCHAR2 (255);
    v_update_rule VARCHAR2 (255);
    v_delete_rule VARCHAR2 (255);
BEGIN
    DBMS_STREAMS_ADM.ADD_SUBSET_RULES (
    table_name => 'BCIRCUL.CP_PROV',
    dml_condition => 'INSTANCIA = ''N'' AND PROVCIR = ''21'' ',
    streams_type => 'CAPTURE',
    streams_name => 'STREAMS_CAPT_STRCAV',
    queue_name => 'STREAMS_CAPT_STRCAV_Q',
    include_tagged_lcr => TRUE,
    source_database => 'STRCAV.DAS.CITES',
    insert_rule_name => v_insert_rule,
    update_rule_name => v_update_rule,
    delete_rule_name => v_delete_rule);
    DBMS_STREAMS_ADM.RENAME_TABLE (
    rule_name => v_insert_rule,
     from_table_name => 'BCIRCUL.CP_PROV',
    to_table_name => 'BCIRCUL.CPERN');
    DBMS_STREAMS_ADM.RENAME_TABLE (
    rule_name => v_update_rule,
    from_table_name => 'BCIRCUL.CP_PROV',
    to_table_name => 'BCIRCUL.CPERN');
    DBMS_STREAMS_ADM.RENAME_TABLE (
    rule_name => v_delete_rule,
```

```
    from_table_name => 'BCIRCUL.CP_PROV',
    to_table_name => 'BCIRCUL.CPERN');
END;
```

Componente: Captura
Sitio: STRCAV
Usuario: STRMADMIN

Descripción: En este caso, como el Proceso de Captura ya esta creado, solo se crean las reglas para captar los cambios realizados a la tabla VEHPR que las entidades circuladas sean de carácter nacional y pertenezcan la provincia Ciego de Ávila. A cada regla generada (Insert, Update, Delete), se le asocia una regla basada en transformación, ya que las tablas correspondientes en el sitio destino presentan nombres diferentes.

Código:

```
DECLARE
    v_insert_rule VARCHAR2 (255);
    v_update_rule VARCHAR2 (255);
    v_delete_rule VARCHAR2 (255);
BEGIN
    DBMS_STREAMS_ADM.ADD_SUBSET_RULES(
    table_name => 'BCIRCUL.VEHPR',
    dml_condition => 'PRMCIR = "21" ',
    streams_type => 'CAPTURE',
    streams_name => 'STREAMS_CAPT_STRCAV',
    queue_name => 'STREAMS_CAPT_STRCAV_Q',
    include_tagged_lcr => TRUE,
    source_database => 'STRCAV.DAS.CITES',
    insert_rule_name => v_insert_rule,
    update_rule_name => v_update_rule,
    delete_rule_name => v_delete_rule);
    DBMS_STREAMS_ADM.RENAME_TABLE (
    rule_name => v_insert_rule,
    from_table_name => 'BCIRCUL.VEHPR',
    to_table_name => 'BCIRCUL.VEHNAC');
    DBMS_STREAMS_ADM.RENAME_TABLE (
    rule_name => v_update_rule,
    from_table_name => 'BCIRCUL.VEHPR',
    to_table_name => 'BCIRCUL.VEHNAC');
    DBMS_STREAMS_ADM.RENAME_TABLE (
    rule_name => v_delete_rule,
    from_table_name => 'BCIRCUL.VEHPR',
    to_table_name => 'BCIRCUL.VEHNAC');
END;
```

7- Creando Cola

Componente: Aplicación

Sitio: STRCHA
Usuario: STRMADMIN
Descripción: Donde el proceso de Aplicación extrae los LCRs para aplicarlos
Código:
```
BEGIN
    DBMS_STREAMS_ADM.SET_UP_QUEUE (
    queue_table => 'STREAMS_APPLY_STRCHA_QT',
    queue_name => 'STREAMS_APPLY_STRCHA_Q',
    queue_user => 'STRMADMIN');
END;
```

8- Creando Set de Reglas

Componente: Propagación
Sitio: STRCHA
Usuario: STRMADMIN
Descripción: Donde se agrupan todas las Reglas del Proceso de Propagación.
Código:
```
BEGIN
    DBMS_RULE_ADM.CREATE_RULE_SET (
    rule_set_name => 'RULE_SET_PROP_CIRNAC',
    evaluation_context => 'SYS.STREAMS$_EVALUATION_CONTEXT');
END;
```

9- Creando Reglas

Componente: Propagación
Sitio: STRCHA
Usuario: STRMADMIN
Descripción: Regla que evalúa en TRUE todos los mensajes de la tabla CIRN, CPERN y VEHNAC propagándolos hacia CIRNAC.
Código:
```
BEGIN
    DBMS_RULE_ADM.CREATE_RULE (
    rule_name=>'RULE_PROP_CIRNAC_CIRN',
    condition=>':dml.get_object_owner () = "BCIRCUL"
    AND: dml.get_object_name () = "CIRN"
    AND: dml.is_null_tag () = "Y" ‘);
END;

BEGIN
    DBMS_RULE_ADM.CREATE_RULE (
    rule_name=>'RULE_PROP_CIRNAC_CPERN',
    condition=>':dml.get_object_owner () = "BCIRCUL"
    AND: dml.get_object_name () = "CPERN"
    AND: dml.is_null_tag () = "Y" ‘);
```

```
END;

BEGIN
    DBMS_RULE_ADM.CREATE_RULE (
    rule_name=>'RULE_PROP_CIRNAC_VEHNAC',
    condition=>':dml.get_object_owner() = ''BCIRCUL''
    AND: dml.get_object_name () = ''VEHNAC''
    AND: dml.is_null_tag () = ''Y'' ‘);
END;
```

10- Adicionando Reglas al Set de Reglas

Componente: Propagación
Sitio: STRCHA
Usuario: STRMADMIN
Descripción: Se adjunta al Set de Reglas de Propagación la Regla para la tabla tabla CIRN, CPERN y VEHNAC**.**
Código:

```
BEGIN
    DBMS_RULE_ADM.ADD_RULE (
    rule_name => 'RULE_PROP_CIRNAC_CIRN',
    rule_set_name => 'RULE_SET_PROP_CIRNAC');
END;

BEGIN
    DBMS_RULE_ADM.ADD_RULE (
    rule_name => 'RULE_PROP_CIRNAC_CPERN',
    rule_set_name => 'RULE_SET_PROP_CIRNAC');
END;

BEGIN
    DBMS_RULE_ADM.ADD_RULE (
    rule_name => 'RULE_PROP_CIRNAC_VEHNAC',
    rule_set_name => 'RULE_SET_PROP_CIRNAC');
END;
```

11- Creando Propagación

Componente: Propagación
Sitio: STRCHA
Usuario: STRMADMIN
Descripción: Se crea el Proceso de Propagación asignándole el Set de Reglas, la cola fuete y la cola destino.
Código:

```
BEGIN
```

```
    DBMS_PROPAGATION_ADM.CREATE_PROPAGATION (
    propagation_name => 'STREAMS_PROP_CIRNAC',
    source_queue => 'STREAMS_CAPT_STRCHA_Q',
    destination_queue => 'STREAMS_APPLY_STRCHA_Q',
    destination_dblink => 'CIRNAC.DAS.CITES',
    rule_set_name => 'RULE_SET_PROP_CIRNAC',
    queue_to_queue => TRUE);
  END;
```

12- Creando Cola

Componente: Aplicación
Sitio: STRCAV
Usuario: STRMADMIN
Descripción: Donde el proceso de Aplicación extrae los LCRs para aplicarlos
Código:

```
BEGIN
    DBMS_STREAMS_ADM.SET_UP_QUEUE (
    queue_table => 'STREAMS_APPLY_STRCAV_QT',
    queue_name => 'STREAMS_APPLY_STRCAV_Q',
    queue_user => 'STRMADMIN');
  END;
```

13- Creando Set de Reglas

Componente: Propagación
Sitio: STRCAV
Usuario: STRMADMIN
Descripción: Donde se agrupan todas las Reglas del Proceso de Propagación.
Código:

```
BEGIN
    DBMS_RULE_ADM.CREATE_RULE_SET (
    rule_set_name => 'RULE_SET_PROP_CIRNAC',
    evaluation_context => 'SYS.STREAMS$_EVALUATION_CONTEXT');
  END;
```

14- Creando Reglas

Componente: Propagación
Sitio: STRCAV
Usuario: STRMADMIN
Descripción: Regla que evalúa en TRUE todos los mensajes de la tabla CIRN, CPERN y VEHNAC propagándolos hacia CIRNAC
Código:

```
BEGIN
    DBMS_RULE_ADM.CREATE_RULE (
```

```
    rule_name=>'RULE_PROP_CIRNAC_CIRN',
    condition=>':dml.get_object_owner () = "BCIRCUL"
    AND: dml.get_object_name () = "CIRN"
    AND: dml.is_null_tag () = "Y" ' );
END;

BEGIN
    DBMS_RULE_ADM.CREATE_RULE (
    rule_name=>'RULE_PROP_CIRNAC_CPERN',
    condition=>':dml.get_object_owner() = "BCIRCUL"
    AND: dml.get_object_name () = "CPERN"
    AND: dml.is_null_tag() = "Y" ' );
END;

BEGIN
    DBMS_RULE_ADM.CREATE_RULE (
    rule_name=>'RULE_PROP_CIRNAC_VEHNAC',
    condition=>':dml.get_object_owner() = "BCIRCUL"
    AND: dml.get_object_name () = "VEHNAC"
    AND: dml.is_null_tag () = "Y" ‘);
END;
```

15- Adicionando Reglas al Set de Reglas

Componente: Propagación
Sitio: STRCAV
Usuario: STRMADMIN
Descripción: Se adjunta al Set de Reglas de Propagación la Regla para la tabla CIRN, CPERN y VEHNAC.
Código:

```
BEGIN
    DBMS_RULE_ADM.ADD_RULE (
    rule_name => 'RULE_PROP_CIRNAC_CIRN',
    rule_set_name => 'RULE_SET_PROP_CIRNAC');
END;

BEGIN
    DBMS_RULE_ADM.ADD_RULE (
    rule_name => 'RULE_PROP_CIRNAC_CPERN',
    rule_set_name => 'RULE_SET_PROP_CIRNAC');
END;

BEGIN
    DBMS_RULE_ADM.ADD_RULE (
    rule_name => 'RULE_PROP_CIRNAC_VEHNAC',
```

```
    rule_set_name => 'RULE_SET_PROP_CIRNAC');
END;
```

16- Creando Propagación

Componente: Propagación
Sitio: STRCAV
Usuario: STRMADMIN
Descripción: Se crea el Proceso de Propagación asignándole el Set de Reglas, la cola fuete y la cola destino.
Código:

```
BEGIN
    DBMS_PROPAGATION_ADM.CREATE_PROPAGATION (
    propagation_name => 'STREAMS_PROP_CIRNAC',
    source_queue => 'STREAMS_CAPT_STRCAV_Q',
    destination_queue => 'STREAMS_APPLY_STRCAV_Q',
    destination_dblink => 'CIRNAC.DAS.CITES',
    rule_set_name => 'RULE_SET_PROP_CIRNAC',
    queue_to_queue => TRUE);
END;
```

17- Creando Set de Reglas

Componente: Aplicación
Sitio: CIRNAC
Usuario: STRMADMIN
Descripción: Donde se agrupan todas las Reglas del Proceso de Aplicación.
Código:

```
BEGIN
    DBMS_RULE_ADM.CREATE_RULE_SET (
    rule_set_name => 'RULE_SET_APPLY_STRCHA',
    evaluation_context => 'SYS.STREAMS$_EVALUATION_CONTEXT');
END;
```

18- Creando Reglas

Componente: Aplicación
Sitio: CIRNAC
Usuario: STRMADMIN
Descripción: Regla que evalúa en TRUE todos los mensajes y los aplica en la tabla CIRN, CPERN y VEHNAC.
Código:

```
BEGIN
    DBMS_RULE_ADM.CREATE_RULE (
    rule_name=>'RULE_APPLY_STRCHA_CIRN',
    condition=>':dml.get_object_owner () = ''BCIRCUL''
```

```
    AND: dml.get_object_name () = "CIRN"
    AND: dml.is_null_tag () = "Y" ' );
  END;

BEGIN
    DBMS_RULE_ADM.CREATE_RULE (
    rule_name=>'RULE_APPLY_STRCHA_CPERN',
    condition=>':dml.get_object_owner () = "BCIRCUL"
    AND: dml.get_object_name () = "CPERN"
    AND: dml.is_null_tag () = "Y" ‘);
  END;

BEGIN
    DBMS_RULE_ADM.CREATE_RULE (
    rule_name=>'RULE_APPLY_STRCHA_VEHNAC',
    condition=>':dml.get_object_owner() = "BCIRCUL"
    AND: dml.get_object_name () = "VEHNAC"
    AND: dml.is_null_tag() = "Y" ' );
  END;
```

19- Adicionando Reglas al Set de Reglas

Componente: Aplicación
Sitio: CIRNAC
Usuario: STRMADMIN
Descripción: Se adjunta al Set de Reglas de Aplicación la Regla para la tabla CIRN, CPERN y VEHNAC.
Código:

```
BEGIN
    DBMS_RULE_ADM.ADD_RULE (
    rule_name => 'RULE_APPLY_STRCHA_CIRN',
    rule_set_name => 'RULE_SET_APPLY_STRCHA');
  END;

BEGIN
    DBMS_RULE_ADM.ADD_RULE (
    rule_name => 'RULE_APPLY_STRCHA_CPERN',
    rule_set_name => 'RULE_SET_APPLY_STRCHA');
  END;

BEGIN
    DBMS_RULE_ADM.ADD_RULE
    (rule_name => 'RULE_APPLY_STRCHA_VEHNAC',
    rule_set_name => 'RULE_SET_APPLY_STRCHA');
  END;
```

20- Creando Aplicación

Componente: Aplicación
Sitio: CIRNAC
Usuario: STRMADMIN
Descripción: Se crea el Proceso de Aplicación asignándole el Set de Reglas
Código:
```
BEGIN
    DBMS_APPLY_ADM.CREATE_APPLY (
    queue_name => 'STREAMS_APPLY_STRCHA_Q',
    apply_name => 'STREAMS_APPLY_STRCHA',
    apply_captured => TRUE);
END;
```

Componente: Aplicación
Sitio: CIRNAC
Usuario: STRMADMIN
Descripción: Eliminar tag para que sean capturados en CIRNAC las aplicaciones de STREAMS
Código:
```
BEGIN
    DBMS_APPLY_ADM.ALTER_APPLY (
    apply_name => 'STREAMS_APPLY_STRCHA',
    remove_apply_tag => TRUE);
END;
```

21- Iniciando Proceso de Aplicación

Componente: Aplicación
Sitio: CIRNAC
Usuario: STRMADMIN
Descripción: Se inicia el Proceso de Aplicación creado anteriormente
Código:
```
BEGIN
    dbms_apply_adm.start_apply ('STREAMS_APPLY_STRCHA');
END;
```

22- Creando Set de Reglas

Componente: Aplicación
Sitio: CIRNAC
Usuario: STRMADMIN
Descripción: Donde se agrupan todas las Reglas del Proceso de Aplicación.
Código:
```
BEGIN
    DBMS_RULE_ADM.CREATE_RULE_SET (
    rule_set_name => 'RULE_SET_APPLY_STRCAV',
    evaluation_context => 'SYS.STREAMS$_EVALUATION_CONTEXT');
```

```
END;
```

23- Creando Reglas

Componente: Aplicación
Sitio: CIRNAC
Usuario: STRMADMIN
Descripción: Regla que evalúa en TRUE todos los mensajes y los aplica en la tabla CIRN, CPERN y VEHNAC.
Código:

```
BEGIN
    DBMS_RULE_ADM.CREATE_RULE (
    rule_name=>'RULE_APPLY_STRCAV_CIRN',
    condition=>':dml.get_object_owner () = "BCIRCUL"
    AND: dml.get_object_name () = "CIRN"
    AND: dml.is_null_tag () = "Y" ‘);
END;

BEGIN
    DBMS_RULE_ADM.CREATE_RULE (
    rule_name=>'RULE_APPLY_STRCAV_CPERN',
    condition=>':dml.get_object_owner () = "BCIRCUL"
    AND: dml.get_object_name () = "CPERN"
    AND: dml.is_null_tag () = "Y" ' );
END;

BEGIN
    DBMS_RULE_ADM.CREATE_RULE (
    rule_name=>'RULE_APPLY_STRCAV_VEHNAC',
    condition=>':dml.get_object_owner () = "BCIRCUL"
    AND: dml.get_object_name() = "VEHNAC"
    AND: dml.is_null_tag () = "Y" ' );
END;
```

24- Adicionando Reglas al Set de Reglas

Componente: Aplicación
Sitio: CIRNAC
Usuario: STRMADMIN
Descripción: Se adjunta al Set de Reglas de Aplicación la Regla para la tabla CIRN, CPERN y VEHNAC.
Código:

```
BEGIN
    DBMS_RULE_ADM.ADD_RULE (
    rule_name => 'RULE_APPLY_STRCAV_CIRN',
    rule_set_name => 'RULE_SET_APPLY_STRCAV');
```

```
END;

BEGIN
    DBMS_RULE_ADM.ADD_RULE (
    rule_name => 'RULE_APPLY_STRCAV_CPERN',
    rule_set_name => 'RULE_SET_APPLY_STRCAV');
END;

BEGIN
    DBMS_RULE_ADM.ADD_RULE (
    rule_name => 'RULE_APPLY_STRCAV_VEHNAC',
    rule_set_name => 'RULE_SET_APPLY_STRCAV');
END;
```

25- Creando Aplicación

Componente: Aplicación
Sitio: CIRNAC
Usuario: STRMADMIN
Descripción: Se crea el Proceso de Aplicación asignándole el Set de Reglas
Código:

```
BEGIN
    DBMS_APPLY_ADM.CREATE_APPLY (
    queue_name => 'STREAMS_APPLY_STRCAV_Q',
    apply_name => 'STREAMS_APPLY_STRCAV',
    apply_captured => TRUE);
END;
```

Componente: Aplicación
Sitio: CIRNAC
Usuario: STRMADMIN
Descripción: Eliminar tag para que sean capturados en CIRNAC las aplicaciones de STREAMS
Código:

```
BEGIN
    DBMS_APPLY_ADM.ALTER_APPLY (
    apply_name => 'STREAMS_APPLY_STRCAV',
    remove_apply_tag => TRUE);
END;
```

26- Iniciando Proceso de Aplicación

Componente: Aplicación
Sitio: CIRNAC
Usuario: STRMADMIN
Descripción: Se inicia el Proceso de Aplicación creado anteriormente

Código:
```
BEGIN
    dbms_apply_adm.start_apply ('STREAMS_APPLY_STRCAV');
 END;
```

27- Preparando tablas para instanciar

Componente: Captura
Sitio: STRCHA
Usuario: STRMADMIN
Descripción: Se prepara la tabla CI_PROV, CP_PROV y VEHPR de la base de datos fuente para instanciar y se le adiciona el supplemental logging a todos los campos de la tabla.
Código:
```
BEGIN
    DBMS_CAPTURE_ADM.PREPARE_TABLE_INSTANTIATION (
    table_name =>'BCIRCUL.CI_PROV',
    supplemental_logging => 'all');
 END;

BEGIN
    DBMS_CAPTURE_ADM.PREPARE_TABLE_INSTANTIATION (
    table_name =>'BCIRCUL.CP_PROV',
    supplemental_logging => 'all');
 END;

BEGIN
    DBMS_CAPTURE_ADM.PREPARE_TABLE_INSTANTIATION (
    table_name =>'BCIRCUL.VEHPR',
    supplemental_logging => 'all');
 END;
```

28- Instanciando tablas

Componente: Captura
Sitio: STRCHA
Usuario: STRMADMIN
Descripción: Se instancia la tabla CI_PROV, CP_PROV y VEHPR para replicar hacia la base de datos CIRNAC.
Código:
```
DECLARE
    source_scn number;
BEGIN
    source_scn:= dbms_flashback.get_system_change_number ();
    dbms_apply_adm.set_table_instantiation_scn@cirnac.das.cites
    ( source_object_name => 'BCIRCUL.CI_PROV',
      source_database_name => 'STRCHA.DAS.CITES',
      instantiation_scn => source_scn);
```

```
END;

DECLARE
    source_scn number;
BEGIN
    source_scn:= dbms_flashback.get_system_change_number ();
    dbms_apply_adm.set_table_instantiation_scn@cirnac.das.cites
    (source_object_name => 'BCIRCUL.CP_PROV',
     source_database_name => 'STRCHA.DAS.CITES',
     instantiation_scn => source_scn);
END;

DECLARE
    source_scn number;
BEGIN
    source_scn := dbms_flashback.get_system_change_number ();
    dbms_apply_adm.set_table_instantiation_scn@cirnac.das.cites
    ( source_object_name => 'BCIRCUL.VEHPR',
     source_database_name => 'STRCHA.DAS.CITES',
     instantiation_scn => source_scn);
END;
```

29- Iniciando Proceso de Captura

Componente: Captura
Sitio: CIRNAC
Usuario: STRMADMIN
Descripción: Se inicia el Proceso de Captura.
Código:

```
BEGIN
    dbms_capture_adm.start_capture ('STREAMS_CAPT_STRCHA');
END;
```

30- Preparando tablas para instanciar

Componente: Captura
Sitio: STRCAV
Usuario: STRMADMIN
Descripción: Se prepara la tabla CI_PROV, CP_PROV y VEHPR de la base de datos fuente para instanciar y se le adiciona el supplemental logging a todos los campos de la tabla.
Código:

```
BEGIN
    DBMS_CAPTURE_ADM.PREPARE_TABLE_INSTANTIATION (
    table_name =>'BCIRCUL.CI_PROV',
    supplemental_logging => 'all');
END;
```

```
BEGIN
    DBMS_CAPTURE_ADM.PREPARE_TABLE_INSTANTIATION (
    table_name =>'BCIRCUL.CP_PROV',
    supplemental_logging => 'all');
 END;

BEGIN
    DBMS_CAPTURE_ADM.PREPARE_TABLE_INSTANTIATION (
    table_name =>'BCIRCUL.VEHPR',
    supplemental_logging => 'all');
 END;
```

31- Instanciando tablas

Componente: Captura
Sitio: STRCAV
Usuario: STRMADMIN
Descripción: Se instancia la tabla CI_PROV, CP_PROV y VEHPR para replicar hacia la base de datos CIRNAC.
Código:

```
DECLARE
    source_scn number;
BEGIN
    source_scn := dbms_flashback.get_system_change_number ();
    dbms_apply_adm.set_table_instantiation_scn@cirnac.das.cites
    ( source_object_name => 'BCIRCUL.CI_PROV',
      source_database_name => 'STRCAV.DAS.CITES',
      instantiation_scn => source_scn);
 END;

DECLARE
    source_scn number;
BEGIN
    source_scn := dbms_flashback.get_system_change_number ();
    dbms_apply_adm.set_table_instantiation_scn@cirnac.das.cites
    ( source_object_name => 'BCIRCUL.CP_PROV',
      source_database_name => 'STRCAV.DAS.CITES',
      instantiation_scn => source_scn);
 END;

DECLARE
    source_scn number;
BEGIN
    source_scn := dbms_flashback.get_system_change_number ();
    dbms_apply_adm.set_table_instantiation_scn@cirnac.das.cites
    ( source_object_name => 'BCIRCUL.VEHPR',
```

```
        source_database_name => 'STRCAV.DAS.CITES',
        instantiation_scn => source_scn);
    END;
```

32- Iniciando Proceso de Captura

Componente: Captura
Sitio: STRCAV
Usuario: STRMADMIN
Descripción: Se inicia el Proceso de Captura.
Código:

```
BEGIN
    dbms_capture_adm.start_capture ('STREAMS_CAPT_STRCAV');
END;
```