UNIVERSIDAD TECNICA DE AMBATO

FACULTAD DE INGENIERIA EN SISTEMAS

DISEÑO DE BASES DE DATOS DISTRIBUIDAS EMPLEANDO LA ARQUITECTURA DE REPLICACION ORACLE

AUTORES: Luis Roberto Oñate Llerena

Marco Fabián Guangashi Guangasi

DIRECTOR: Ing. Oswaldo Paredes M.Sc.

ASESOR: Ing. Galo López

Tesis Informática de Grado, previa a la obtención del Título de Ingeniero en Sistemas

Ambato - Ecuador

Junio/2005

AGRADECIMIENTO

Al Personal Docente de la Facultad de Ingeniería en Sistemas de la Universidad Técnica de Ambato, que nos supieron guiar y brindar sus conocimientos para lograr terminar la carrera universitaria.

Agradecimiento especial a nuestro Director Ing. Oswaldo Paredes y a nuestro Asesor Ing. Galo López por su buena voluntad y apoyo en el desarrollo de este trabajo.

DEDICATORIA

A **Dios** y a nuestra **Madre la Dolorosa**, ya que su voluntad ha sido que yo termine mi carrera universitaria.

A mis **padres** y **hermanos** que sin su apoyo moral y económico no hubiese sido posible alcanzar este objetivo que me propuse, de manera muy especial a Marcia ¡mil gracias! por haber confiado en mi. A Beatriz que con su optimismo y carisma supo brindarme aliento al igual que a mi querida QK.

ROBERTO

El presente trabajo se lo dedico a todas las personas que han influenciado en mi, directa o indirectamente: A mis **padres** que ha pesar de sus fracasos son símbolos del éxito, a mis **maestros** que han hecho del pasado una experiencia, del presente un momento y del futuro un éxito, a mis **amigos** que compartieron mis penas y glorias, demostrándome que no todos podemos ser grandes pero si podemos ser buenos, y en definitiva a **Dios** por ser todo esto en uno solo.

MARCO

DECLARACION DE AUTENTICIDAD Y RESPONSABILIDAD

Nosotros, Luis Roberto Oñate Llerena con cédula de identidad 18-0264809-5 y Marco Fabián Guangashi Guangasi con cédula de identidad 18-0288714-9.

Declaramos, que la investigación realizada para el diseño y desarrollo de la presente tesis informática, es original auténtica y personal, por esta razón, el contenido así como los efectos legales que se desprendan de este trabajo de Tesis, son y serán de nuestra exclusiva responsabilidad.

---	---
Luis Roberto Oñate Llerena	Marco Fabián Guangashi Guangasi

PROLOGO

Ponemos a consideración la presente tesis que constituye un trabajo investigativo previa la obtención del título de Ingeniero en Sistemas, titulado DISEÑO DE BASES DE DATOS DISTRIBUIDAS EMPLEANDO LA ARQUITECTURA DE REPLICACION ORACLE. Con este trabajo se pretende conocer más la parte teórica de las Bases de Datos Distribuidas, su aplicabilidad y arquitectura funcional, así como su utilización en aplicaciones prácticas de tal forma que la distribución de los datos mediante la Replicación nos ayude a administrar de mejor manera una base de datos distribuida para que está sea mas tolerante a fallas.

INDICE

INDICE DE FIGURAS

INDICE DE GRAFICOS

INDICE DE TABLAS

INDICE DE ANEXOS

ANEXO I

ANEXO II

INTRODUCCION

Los sistemas de información empezaron a utilizar las bases de datos distribuidas aproximadamente a mediados de la década de los 70's, pero no fue sino hasta 1980 cuando la distribución de la información empezó a tomar auge. Originalmente se había pensado en almacenar la información de manera centralizada utilizando un conjunto de herramientas que facilitarán este tipo de almacenamiento. Pero con el paso del tiempo esto produjo ciertos inconvenientes que no eran posibles solucionar.

Estos problemas impulsaron la creación de almacenamientos distribuidos, los cuales hoy en día proveen características indispensables en el manejo de información; es decir, la combinación de las redes de comunicación y las bases de datos.

En años más recientes se ha observado una marcada tendencia hacia la distribución de los sistemas de cómputo en múltiples sitios que se interconectan a través de una red de comunicaciones. La cantidad de innovaciones tecnológicas que ha habido ha promovido un cambio en la forma de observar a los sistemas de información y, en general, a las aplicaciones computacionales. Existen avances tecnológicos que se realizan continuamente en circuitos, dispositivos de almacenamiento, programas y metodologías. Sin embargo, los cambios tecnológicos van de la mano con la demanda de los usuarios y programas para la explotación exhaustiva de tales dispositivos mejorados. Por tanto, existe un continuo desarrollo de nuevos productos los cuales incorporan ideas nuevas desarrolladas por compañías e instituciones académicas.

Aún cuando es posible que un usuario común no perciba los desarrollos relevantes de nuevos productos, para las aplicaciones existe una demanda permanente por mayor funcionalidad, mayor número de servicios, más flexibilidad y mejor rendimiento. Así, al diseñar un nuevo sistema de información o al prolongar la vida de uno ya existente, se debe buscar siempre formas para enlazar las soluciones ofrecidas por la tecnología disponible a las necesidades de las aplicaciones de los usuarios.

Un área en la cual las soluciones están integrando tecnología con nuevas arquitecturas o formas de hacer las cosas es, sin lugar a dudas, el área de los sistemas distribuidos de información. Ellos se refieren al manejo de datos almacenados en facilidades de cómputo localizadas en muchos sitios ligados a través de una red de comunicaciones. Un caso específico de estos sistemas distribuidos es lo que se conoce como bases de datos distribuidas.

Las razones por las que compañías y negocios migran hacia bases de datos distribuidas incluyen razones organizacionales y económicas, para obtener una interconexión confiable y flexible con las bases de datos existentes, y por un crecimiento futuro. El enfoque distribuido de las bases de datos se adapta más naturalmente a la estructura de las organizaciones. Además, la necesidad de desarrollar una aplicación global (que incluya a toda la organización), se resuelve fácilmente con bases de datos distribuidas. Si una organización crece por medio de la creación de unidades o departamentos

nuevos, entonces, el enfoque de bases de datos distribuidas permite un crecimiento suave.

En el presente documento se describen los pasos para el diseño de una base de datos distribuida y la forma de distribuir los datos a través de la replicación.

En el Capítulo I se tratan aspectos generales sobre el tema planteado, en el Capítulo II se tocan aspectos de los Sistemas de Base de Datos Distribuido, en el Capítulo III se tiene información sobre el Manejo de una Base de Datos Distribuida, en el Capítulo IV se tiene conceptos de Transacciones Distribuidas y su manejo, en el Capítulo V se tiene información de la Replicación de datos en Oracle, en el Capítulo VI se tiene información del Diseño e Implementación de un Prototipo de Base de Datos Distribuida con su Replicación, en el Capítulo VII se emiten conclusiones y recomendaciones. Luego se tiene un glosario de términos y anexos.

CAPITULO I

GENERALIDADES

1.1. Planteamiento del problema

La disponibilidad de las bases de datos y de las redes de computadoras ha promovido el desarrollo de un nuevo campo denominado bases de datos distribuidas. Una base de datos distribuida es una base de datos integrada la cual se construye por encima de una red de computadoras en lugar de una sola computadora. Las bases de datos distribuidas ofrecen diversas ventajas a los diseñadores y usuarios de bases de datos. Entre las más importantes se encuentra la transparencia en el acceso y localización de información. Sin embargo, el diseño y administración de bases de datos distribuidas constituye un gran desafío que incorpora problemas no encontrados en bases de datos centralizadas. Por ejemplo, la localización de información, el manejo de consultas a sitios distribuidos y los mecanismos de control de concurrencia y confiabilidad en bases de datos distribuidas.

El Estudio de Bases de Datos Distribuidas tiene como finalidad conocer las características de los sistemas distribuidos como: Descentralización de datos, Eficiencia de acceso a los datos, Redundancia, Transparencia al usuario, Complejidad. Así como

también los métodos principales para la distribución de almacenamiento de datos en los sistemas distribuidos.

Por esta razón se decidió formular el siguiente tema como trabajo de investigación: **“DISEÑO DE BASES DE DATOS DISTRIBUIDAS EMPLEANDO LA ARQUITECTURA DE REPLICACION ORACLE ”.**

Este trabajo servirá a todas las personas interesadas en aprender más acerca de las bases de datos distribuidas en Oracle.

1.2. Justificación

El presente Trabajo de Investigación se justifica por varias razones, entre las cuales se ponen a consideración las más significativas:

- Este tipo de base de datos no es tan explotado en nuestro medio, razón por la cual sería importante realizar una demostración de cómo se diseña e implementa una base de datos distribuida.
- Muchas organizaciones están descentralizadas, y una aproximación de BDD se acopla más naturalmente a la estructura de la organización. Con los recientes desarrollos en tecnología, la motivación de economías de escala de tener grandes ordenadores centrales está siendo cuestionado.

- Reducida sobrecarga de comunicación: En una base de datos distribuida geográficamente, el hecho de que muchas aplicaciones sean locales claramente reduce la sobrecarga de comunicación con respecto a las bases de datos centralizadas.

1.3. Objetivos

1.3.1. General

- Estudiar las Bases de Datos Distribuidas su aplicabilidad y arquitectura funcional, así como su utilización en aplicaciones prácticas.

1.3.2. Específicos

- Conocer los fundamentos teóricos de las Bases de Datos Distribuidas.
- Conocer las formas de distribución de datos y sus técnicas.
- Establecer los requisitos para la implementación de bases de datos distribuidas.
- Diseñar e implementar un prototipo de base de datos distribuida en Oracle.

1.4. Formulación de hipótesis

El Estudio de Bases de Datos Distribuidas nos permitirá contar con información para el diseño, distribución y optimización de datos para lograr sistemas más tolerantes a fallas.

1.5. Metodología

Para este trabajo de investigación, se utilizó el Método Científico por que se siguió etapas lógicas. El tipo de Investigación que se aplicó es la Bibliográfica y Experimental, en la primera se usó la técnica bibliográfica para la recolección de la información por que los datos fueron tomados de libros, revistas, Internet, etc. Y en la segunda para realizar las pruebas necesarias de la aplicación.

CAPITULO II

SISTEMA DE BASE DE DATOS DISTRIBUIDO

2.1. Sistema de base de datos distribuido

Un Sistema de Bases de Datos Distribuida (SBDD) es un sistema en el cual múltiples sitios de bases de datos están ligados por un sistema de comunicaciones de tal forma que un usuario, en cualquier sitio, puede acceder a los datos en cualquier parte de la red exactamente como si los datos estuvieran residiendo en su máquina. Los principales factores que distinguen un SBDD de un sistema centralizado son los siguientes:

- Hay múltiples computadores, llamados sitios o nodos.
- Estos sitios deben de estar comunicados por medio de algún tipo de red de comunicaciones para transmitir datos y órdenes entre los sitios (Ver figura 1.)

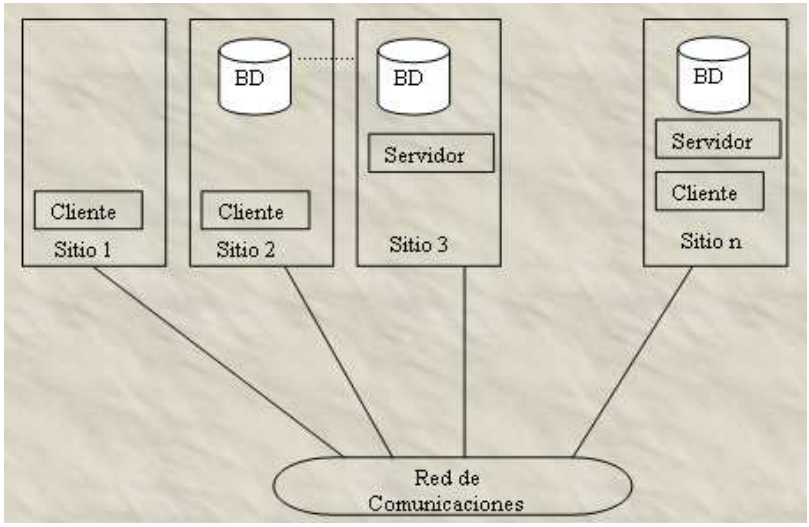


Figura 1. Componentes de un sistema distribuido de bases de datos

Un Sistema de Base de Datos Distribuido se compone de un conjunto de sitios, cada uno de los cuales mantiene un sistema de base de datos local. Cada sitio puede procesar transacciones locales, es decir, aquellas que sólo acceden a datos que residen en ese sitio. Además un sitio puede participar en la ejecución de transacciones globales, es decir, aquellas que acceden a datos de varios sitios.

La ejecución de transacciones globales requiere de comunicación entre los sitios, esta comunicación puede ser, por ejemplo, mediante líneas telefónicas o cables de alta velocidad.

2.2. Características de los sistemas distribuidos

- Cada sitio es un sistema de base de datos en sí mismo.

- Los sitios pueden trabajar juntos (si es necesario) con el fin de que un usuario de cualquier sitio pueda obtener acceso a los datos de cualquier punto de la red tal como si todos estuvieran almacenados en el sitio propio del usuario.

2.3. Beneficios de los sistemas distribuidos

- La descentralización evita la sobrecarga en un sitio y elimina la dependencia del mismo.
- Mejor tiempo de respuesta, gracias al procesamiento local en cada uno de los nodos sobre los que se distribuye el sistema. Esto incrementa el rendimiento a través de un alto grado de paralelismo (*Transacciones al igual que reportes más rápidos*).
- Sobrecarga de comunicación reducida, con la maximización de la localidad de las aplicaciones.
- Reducción de costos, por la minimización del uso del canal y por el hecho que es más rentable adquirir múltiples equipos que un gran computador central (*por lo menos esto es válido en el caso de los mainframes*).
- La extensibilidad en el hardware, al adicionar una estación a la red fácilmente. En la base de datos, al permitir interconectar las bases de datos preexistentes, conformando una base de datos distribuida. La heterogeneidad (*Diferentes tipos de bases de datos. Por ejemplo, una base de datos en Oracle, otra en Access,*

etc.) es representativa aquí ya que se rompen dependencias con un proveedor, permitiendo que el sistema se extienda con productos diferentes.

- Disponibilidad, se logra por la redundancia del hardware, del software y de los datos, también por la dispersión de recursos.

2.4. Problemas de los sistemas distribuidos

- El control del acceso a datos dispersos se vuelve inmanejable.
- La coordinación de procesos concurrentes y remotos resulta más compleja.
- El montaje de sistemas distribuidos en ambientes heterogéneos no está completamente resuelto. A nivel de protocolos está resuelto pero los niveles aplicativos siguen presentando problemas.
- Asegurar el desarrollo conjunto de software es un inconveniente que tiene que ver con la disciplina, la heterogeneidad de los grupos de trabajo y la misma idiosincrasia y costumbres de las personas.
- Ningún sitio puede observar el estado global de todo el sistema, por lo cual se corre con el riesgo de tomar decisiones inconsistentes en el procesamiento.
- Cuando hay una inadecuada migración hacia los sistemas distribuidos se pueden generar problemas adicionales como los siguientes:
    a. Pérdida del control gerencial.
    b. Pérdida del control en el manejo de los sistemas de información.
    c. Suboptimización.

d. Fallas en el aprovechamiento del SMBD (Sistema Manejador de Bases de Datos), ya que los usuarios implementan archivos locales en otras herramientas.
e. Problemas de mantenimiento.

2.5. Definición de base de datos distribuida (BDD)

Es un conjunto de múltiples bases de datos lógicamente relacionadas las cuales se encuentran distribuidas entre diferentes sitios interconectados por una red de comunicaciones, los cuales tienen la capacidad de procesamiento autónomo, lo cual indica que puede realizar operaciones locales o distribuidas.

2.6. Ventajas y desventajas de las bases de datos distribuidas

Ventajas

- Los datos son localizados en un lugar más cercano, por tanto, el acceso es más rápido.
- El procesamiento es rápido debido a que varios nodos intervienen en el procesamiento de una carga de trabajo, nuevos nodos se pueden agregar fácil y rápidamente.

- La comunicación entre nodos se mejora, los costos de operación se reducen, son amigables al usuario, la probabilidad que una falla en un solo nodo afecte al sistema es baja y existe una autonomía e independencia entre los nodos.
- Los datos se pueden colocar físicamente en el lugar donde se accedan más frecuentemente, haciendo que los usuarios tengan control local de los datos con los que interactúan.
- Mediante la replicación de información, las bases de datos distribuidas pueden presentar cierto grado de tolerancia a fallas haciendo que el funcionamiento del sistema no dependa de un solo lugar como en el caso de las bases de datos centralizadas.
- La naturaleza distribuida de algunas aplicaciones de Bases de Datos: Muchas de estas aplicaciones están distribuidas naturalmente en diferentes lugares. Por ejemplo, una compañía puede tener oficinas en varias ciudades, un banco puede tener múltiples sucursales.
- Mayor fiabilidad y disponibilidad: Estas son dos de las ventajas potenciales de las Bases de Datos Distribuidas que se citan comúnmente, la fiabilidad se define a grandes rasgos como la probabilidad de que un sistema este en funciones en un momento determinado, y la disponibilidad es la probabilidad de que el sistema esté disponible continuamente durante un intervalo de tiempo.
- Mejor rendimiento: Cuando una base de datos grande esta muy distribuida en múltiples sitios, hay bases de datos más pequeñas en cada uno de estos. En consecuencia, las consultas locales y las transacciones que tienen acceso a datos

a un solo sitio tienen un mejor rendimiento porque las bases de datos son más pequeñas, además cada sitio tiene un mejor número de transacciones en ejecución que si todas las transacciones se enviaran a una sola base de datos centralizada.

Desventajas

- La principal desventaja se refiere al control y manejo de los datos. Dado que éstos residen en muchos nodos diferentes y se pueden consultar por nodos diversos de la red, la probabilidad de violaciones de seguridad es creciente si no se toman las precauciones debidas.
- Asegurar la integridad de la información en presencia de fallas no predecibles, tanto de componentes de hardware como de software, es compleja. La integridad se refiere a la consistencia, validez y exactitud de la información.
- Dado que los datos pueden estar replicados, el control de concurrencia y los mecanismos de recuperación son mucho más complejos que en un sistema centralizado.
- La distribución produce un aumento en la complejidad del diseño y en la implementación del sistema.

2.7. Sistema manejador de bases de datos distribuidas (SMBDD)

Es aquel que se encarga del manejo de la BDD, y proporciona un mecanismo de acceso que hace que la distribución sea transparente a los usuarios. El término transparente significa que la aplicación trabajaría, desde un punto de vista lógico, como si un solo SMBD ejecutado en una sola máquina administrara esos datos.

2.8. Arquitectura de una base de datos distribuida

Un sistema de base de datos distribuida permite desarrollar aplicaciones para acceder a información local y remota de una base de datos. En un sistema homogéneo de base de datos distribuida, cada base de datos es una base de datos Oracle, mientras que en un sistema heterogéneo de base de datos distribuida, por lo menos una de las bases de datos no es una base de datos Oracle. Las bases de datos distribuidas usan arquitectura cliente / servidor para procesar la información requerida.

2.8.1. Sistemas homogéneos de base de datos distribuidas

Un sistema homogéneo de base de datos distribuida es una red de dos o más bases de datos que residen en una o más máquinas.

La Figura 2. ilustra un sistema distribuido que conecta tres bases de datos: Oficina Principal (***op*),** Fábrica (***fa***) y Ventas (***ve***). Una aplicación puede acceder o modificar simultáneamente el dato en varias bases de datos en un solo ambiente distribuido.

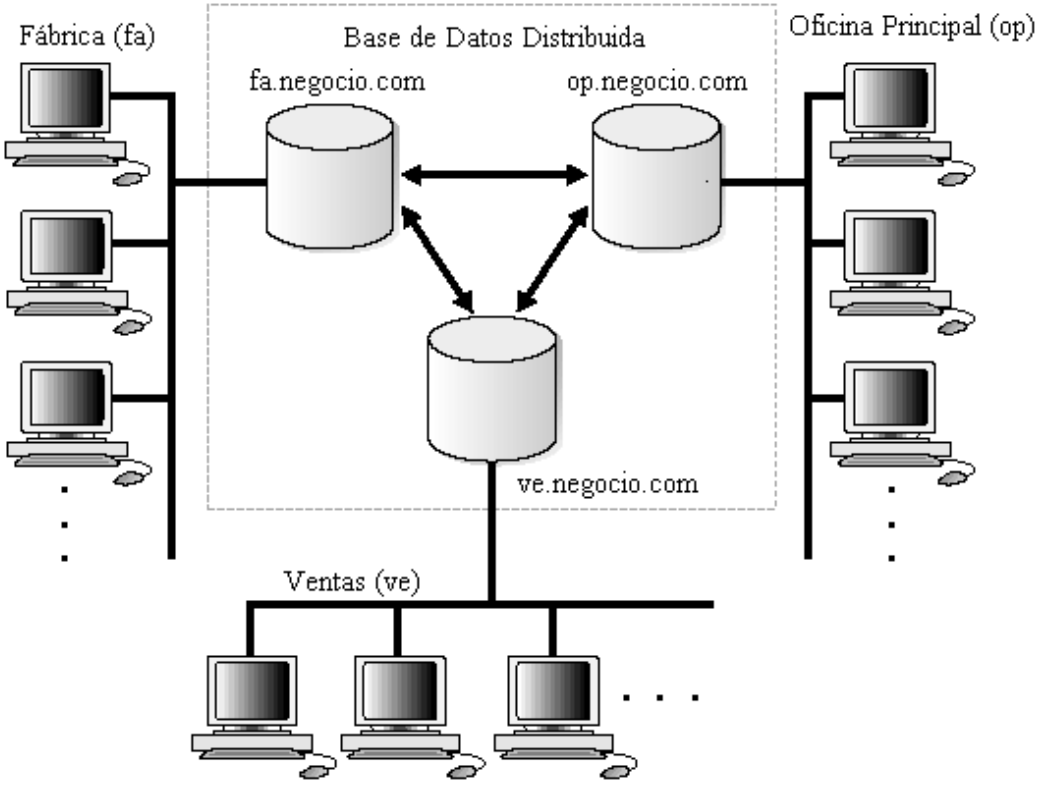


Figura 2. Base de datos distribuida homogénea

En Oracle, un sistema de base de datos distribuido puede incorporar diferentes versiones de bases de datos, de tal forma que pueden interactuar en un sistema de base de datos distribuidos. No obstante, las aplicaciones que trabajan con bases de datos distribuidas deben entender su funcionalidad que es habilitar a cada nodo del sistema.

Las bases de datos distribuidas constituyen un sistema distribuido de tal forma que, todas las aplicaciones parecen ser una sola fuente generadora de datos.

Los procesos distribuidos constituyen las operaciones que realiza una aplicación distribuyéndola por todas las computadoras dentro de la red. Por ejemplo, una aplicación de base de datos típicamente distribuye las tareas de presentación front end a todas las computadoras clientes y permite un back end al servidor de base de datos para manejar el acceso compartido de la base de datos. Por lo tanto, un sistema procesando una aplicación de base de datos distribuida es normalmente llamado un sistema de aplicación de base de datos cliente/servidor.

En una ***base de datos distribuida pura*** (*es decir, no replicada*) el sistema maneja una copia de todos los datos y objetos que soporta la base de datos. Típicamente las aplicaciones de base de datos distribuidas usan transacciones distribuidas para acceder a datos locales y remotos, y modificar la base de datos en tiempo real.

El término ***replicación*** se refiere a las operaciones de copiado y mantenimiento de los objetos de base de datos en múltiples bases de datos que pertenecen a un sistema distribuido, la replicación confía en la tecnología de base de datos distribuida y ofrece beneficios que no son posibles dentro de un ambiente de base de datos distribuido puro.

La replicación es usada para mejorar el cumplimiento de las bases de datos locales y proteger a las aplicaciones que están disponibles, ya que una aplicación puede acceder

normalmente a una base de datos local en lugar de un servidor remoto que minimiza el tráfico de la red.

2.8.2. Sistemas heterogéneos de base de datos distribuidas

En un sistema heterogéneo de base de datos distribuida, al menos una base de datos no es Oracle. En una aplicación los sistemas heterogéneos aparecen como una sola base de datos local. El servidor de base de datos local esconde la distribución y la heterogeneidad de los datos.

El servidor de base de datos Oracle accede al sistema que no es Oracle usando servicios heterogéneos en conjunto con un agente. Si se accede a los datos almacenados no Oracle usando una entrada transparente, entonces, el agente es un sistema de aplicación específica.

Se utilizan los siguientes servicios para acceder a sistemas no Oracle:

- **Servicios heterogéneos (*SH*),** es un componente integrado del servidor de base de datos Oracle, proporcionan una arquitectura común y mecanismos de administración así como otros medios de acceso heterogéneos. También, proporciona alta funcionalidad compatible para usuarios remotos con entradas transparentes libres.

- **Agentes de entrada transparentes,** al acceder a un sistema no Oracle, los servicios heterogéneos pueden usar un agente de entrada transparente para unirlo con el sistema Oracle. El agente es específico para el sistema no Oracle, así que cada tipo de sistema requiere un agente diferente. Los agentes de entradas transparentes facilitan la comunicación entre las diferentes bases y usan los componentes de servicios heterogéneos en el servidor de base de datos Oracle. El agente ejecuta *SQL* y las demandas transaccionales sobre el sistema no Oracle en el servidor de base de datos Oracle.
- **Conectividad genérica,** permite conectar datos que no son Oracle, usando un agente de servicio *ODBC* u *OLE DB*. Cualquier dato fuente compatible con *ODBC* u *OLE DB* puede acceder a los datos, usando simplemente conectividad genérica. Su ventaja es que puede no requerir configuración separada para un agente de un sistema específico y usar un controlador *ODBC* u *OLE DB* que conecte al agente. Sin embargo, algunos datos solo son accesibles con agentes de entradas transparentes.

2.8.3. Arquitectura de base de datos cliente/servidor

Un Servidor de Base de Datos, es el software Oracle que administra una base de datos, y un Cliente es una aplicación que pide información del servidor. Cada computadora en la red es un nodo, es decir, un host o varias bases de datos. Cada nodo en un sistema de

base de datos distribuida puede actuar como un cliente, un servidor o ambos a la vez, dependiendo de la situación.

En la Figura 3. el host para la base Oficina Principal (***op***) está actuando como un servidor de base de datos, cuando una declaración es emitida contra el dato local (por ejemplo, la segunda declaración dentro de cada transacción emite una declaración en contra de la tabla local Departamentos (***dept***)), pero está actuando como un cliente cuando emite una declaración en contra del dato remoto (por ejemplo, la primera declaración de cada transacción es emitida en contra de la tabla remota Empleados (***emp***) en la base de datos Ventas (***ve***)).

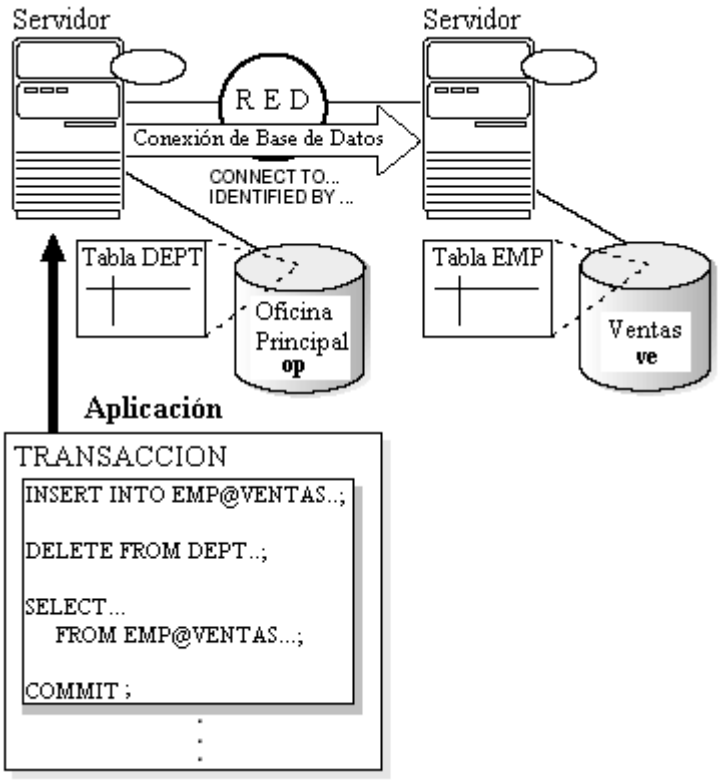


Figura 3. Sistema de base de datos distribuido en oracle

Un cliente puede conectarse directa o indirectamente al servidor de base de datos. Una conexión directa ocurre cuando un cliente se conecta al servidor y accede a la información contenida en ese servidor. Por ejemplo, si se conecta a la base oficina principal (***op***) y se accede a la tabla departamentos (***dept***) de esta base de datos, como se muestra en la Figura 3. se puede emitir la siguiente sentencia SQL:

```
SELECT * FROM dept;
```

Este tipo de consulta es directa porque no se conecta por medio de un objeto a la base de datos remota. En contraste, una conexión indirecta ocurre cuando un cliente se conecta al servidor y dicha información se encuentra contenida en un servidor diferente.

Por ejemplo, si se conecta a la base de datos oficina principal (***op***) pero se accede a la tabla empleados (***emp***) de la base de datos remota ***ventas*** como en la Figura 3. se puede emitir la siguiente consulta SQL:

```
SELECT * FROM  emp@ventas;
```

Esta consulta es indirecta porque el objeto al que está accediendo no esta en la base de datos a la que se conecta directamente.

2.9. Enlaces de bases de datos (Links)

Permite una conexión física entre dos servidores de base de datos para que un cliente pueda acceder a una base de datos lógica.

Los enlaces de base de datos son punteros que definen una vía de comunicación desde un servidor de base de datos Oracle a otro servidor de base de datos. El puntero enlazador esta definido como una entrada a la tabla del diccionario de datos. Para acceder al enlazador, se debe estar conectado a la base de datos local que contiene la entrada al diccionario de datos. Por ejemplo; la vía de comunicación de un cliente conectado a la base local A, puede usar un enlace almacenado en la base de datos A para acceder a la información remota de la base de datos B, pero un usuario conectado a la base de datos B no puede usar el mismo enlace para acceder a los datos en la base de datos A. Sí un usuario local de la base de datos B quiere acceder a los datos en la base de datos A, estos deben usar un link almacenado en el diccionario de datos de la base B. Una conexión de enlace de base de datos permite a los usuarios locales acceder a información de una base de datos remota. Para que esta conexión ocurra cada base de datos en un sistema distribuido debe tener un único nombre de base de datos global dentro del dominio de la red. El nombre de la base de datos global únicamente identifica a un servidor de base de datos dentro de un sistema distribuido.

La Figura 4. muestra un ejemplo del usuario ***Scott*** accediendo a la tabla empleados (***emp***) de una base de datos remota con el nombre global ***op.negocio.com.***

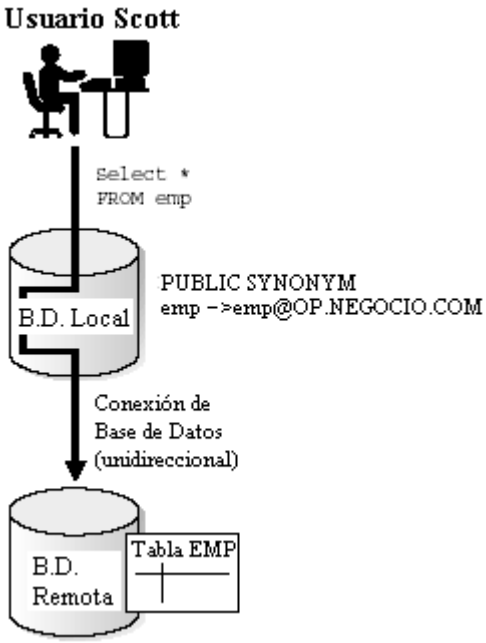


Figura 4. Enlace de base de datos

Los links pueden ser públicos o privados. Si son privados solo el usuario que creó el enlace puede usarlo; Si el link es público todos los usuarios de la base de datos pueden usarlo.

La diferencia principal de los link es la manera en que se conectan a una base de datos remota. Los usuarios pueden acceder a una base de datos remota a través de los siguientes tipos de link.

Link de usuarios conectados (Connected user link)

Son aquellos usuarios que se conectan como ellos mismos, es decir tienen una cuenta en la base de datos remota con el mismo nombre de usuario como su cuenta en la base de datos local.

Link de usuarios fijos (Fixed user link)

Los usuarios se conectan usando un nombre de usuario y una contraseña referenciada en el link. Por ejemplo, si ***María*** usa un link fijo para conectarse a la base de datos oficina principal (***op***) con el nombre de usuario y contraseña ***scott/tiger***, entonces ***María*** conectada como ***Scott***, tiene todos los privilegios en oficina principal (***op***) concedidos a ***Scott*** directamente y todos los roles por defecto de ***Scott*** en la base oficina principal (***op***).

Link de usuario actual (Current user link)

Es un usuario conectado como un usuario global. Un usuario local puede conectarse como un usuario global en un procedimiento almacenado, sin guardar las contraseñas de los usuarios globales dentro del link definido. Por ejemplo, ***María*** puede acceder a un procedimiento que ***Scott*** escribió, accediendo así a la cuenta y esquema de ***Scott*** en la base de datos oficina principal (***op***).

Para crear los link se usa la sentencia:

CREATE DATABASE LINK

2.9.1. Links de base de datos compartidos (Shared Database Links)

Es un enlace entre un proceso del servidor local y la base de datos remota. El link es compartido porque los procesos de múltiples clientes pueden usar el mismo link simultáneamente.

Cuando una base de datos local se conecta a una base de datos remota a través de un link de base de datos, cualquier base de datos puede ejecutarse en modo de servidor dedicado (*especializado*) o compartido. La Tabla 1 indica las posibilidades.

Modo de Base de Datos Local	**Modo de Base de Datos Remoto**
Dedicado	Dedicado
Dedicado	Servidor Compartido
Servidor Compartido	Dedicado
Servidor Compartido	Servidor Compartido

Tabla 1. Posibilidades en que se puede ejecutar la base de datos

Un link de base de datos compartido puede tener cualquiera de las cuatro posibilidades anteriores. Los link compartidos difieren de los link normales en:

- Diferentes usuarios pueden acceder al mismo esquema del objeto a través de un link de base de datos, estos pueden compartir una conexión de la red.

- Cuando un usuario necesita establecer una conexión a un servidor remoto desde un proceso particular del servidor, el proceso puede volver a usar las conexiones establecidas para el servidor remoto. El reuso de la conexión ocurre si la conexión fue establecida en el mismo proceso del servidor, con el mismo link de base de datos, posiblemente en una sesión diferente. En un link de base de datos no compartido, una conexión no es compartida por las múltiples sesiones.
- Cuando se usa un link de base de datos compartido en una configuración de servidor compartido, una conexión en red se establece directamente fuera del proceso del servidor compartido en el servidor local. Para un link de base de datos no compartido en un servidor local compartido, esta conexión se debería establecer a través del distribuidor local, solicitando el cambio al distribuidor local, y exigiendo a los datos que vayan a través del distribuidor.

2.9.2. ¿Por qué usar link de base de datos?

Por que permiten a los usuarios acceder a otros objetos de usuario en una base de datos remota, de tal manera que estos tienen privilegios limitados sobre los objetos. En otras palabras, los usuarios locales pueden acceder a un link remoto de base de datos sin tener que ser usuarios de la base de datos remota.

Los links de base de datos permiten el acceso limitado a bases de datos remotas a usuarios locales. Al usar link se puede crear usuarios globales administrados

centralmente cuya información de contraseña está oculta de administradores y no administradores.

Usando el link de usuarios fijos, se puede crear usuarios no globales cuya información de contraseña se guarda de forma desencriptada en la tabla del diccionario de datos *LINK$*. Los link de usuarios fijos son fáciles de crear y tienen baja sobrecarga, pero tienen riesgo de seguridad en el almacenamiento de contraseñas en el diccionario de datos.

2.9.3. Nombres de bases de datos globales en enlaces de base de datos

Cada base de datos en una base distribuida se identifica únicamente por el nombre de la base de datos global. Oracle forma un nombre de base de datos global prefijando el dominio de la red en la base de datos, especificado por los parámetros de inicialización de DB_DOMAIN en la creación de base de datos, con el nombre de base de datos individual, especificado por los parámetros de inicialización de DB_NAME.

Por ejemplo la Figura 5. muestra un arreglo jerárquico representativo de base de datos a lo largo de una red.

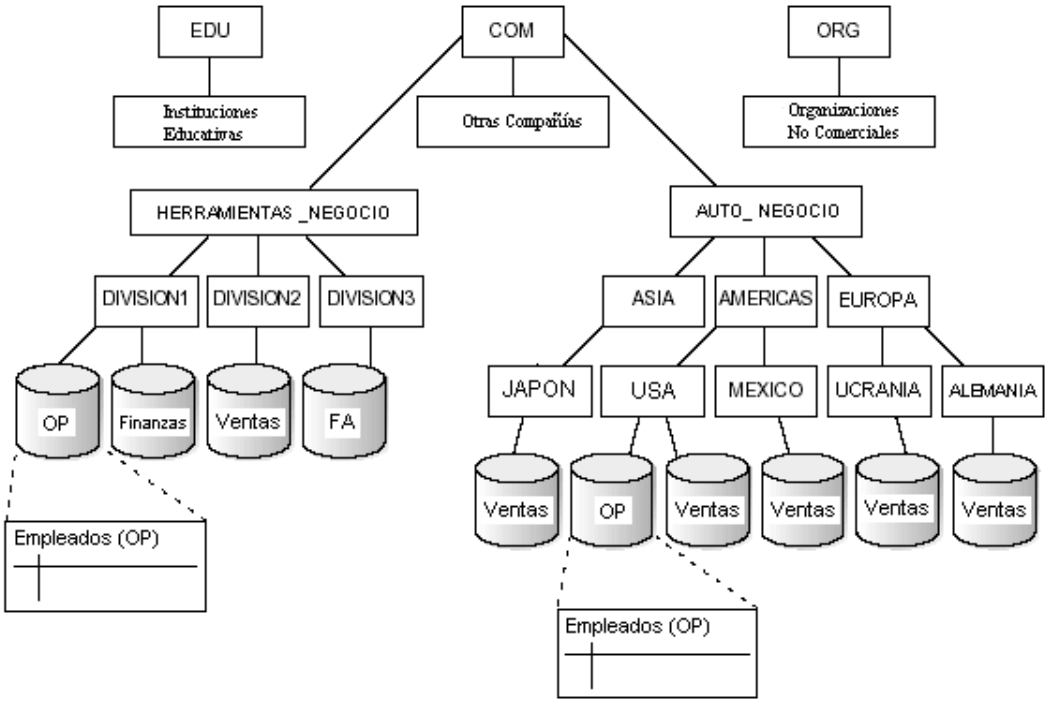


Figura 5. Arreglo jerárquico representativo de una base de datos en una red.

El nombre de una base de datos está formado por el inicio de la hoja del árbol y siguiendo un camino a la raíz. Por ejemplo, la base de datos ***fa*** está en ***división3*** del ***herramientas_negocio*** de la rama del dominio ***com***. El nombre de la base de datos global para ***fa*** está creada por el encadenamiento de nodos en el árbol así:

fa.division3.herramientas_negocio.com

Mientras varias bases de datos pueden compartir un nombre individual, cada base debe tener un único nombre de base de datos global. Por ejemplo, el dominio de red ***usa.americas.auto_negocio.com*** y ***ucrania.europa.auto_negocio.com*** cada uno contiene una base de datos ***ventas***. El sistema nombrado de base de datos global

distingue la base ***ventas*** en la división ***americas*** de la base de datos ***ventas*** en la división ***europa*** así:

```
ventas.usa.americas.auto_negocio.com
ventas.ucrania.europa.auto_negocio.com
```

2.9.4. Nombres para enlaces de base de datos

Un link de base de datos tiene el mismo nombre como el nombre global de base de datos remota a la cual hace referencia. Por ejemplo, si el nombre de la base de datos global de una base de datos es ***ventas.usa.oracle.com***, entonces el link de la base de datos será llamado también ***ventas.usa.oracle.com***.

Cuando se ponen los parámetros de inicialización GLOBAL_NAMES en VERDADERO, Oracle asegura que el nombre de la base de datos enlazada es igual al nombre de la base de datos global en la base de datos remota.

Por ejemplo, si el nombre de la base de datos global para la oficina principal (***op***) es ***op.negocio.com***, y el GLOBAL_NAMES es Verdadero, entonces el nombre de link debe llamarse ***op.negocio.com*.**

Si se ponen los parámetros de inicialización GLOBAL_NAMES a FALSO, entonces significa que no se está usando el nombre global. Se puede entonces nombrar el link de

la base de datos como se desee. Por ejemplo, se puede nombrar un link de base de datos a ***op.negocio.com*** como ***foo***.

Después de habilitar el nombre global, los link de base de datos son esencialmente transparentes a los usuarios de una base de datos distribuida porque el nombre del link de la base de datos es igual al nombre de la base de datos global en el link señalado.
Por ejemplo, la siguiente declaración crea un link de base de datos en la base de datos local para la base de datos remota ***ventas***:

```
CREATE PUBLIC DATABASE LINK ventas.division3.negocio.com
USING 'ventas1';
```

2.9.5. Tipos de links de bases de datos

Oracle permite crear links de base de datos privados, públicos, y globales. Estos tipos básicos de link difieren de acuerdo a los permisos de usuarios para acceder a la base de datos remota.

La Tabla 2. muestra los tipos básicos de links para acceder a bases de datos remotas.

Tipo	Creador	Descripción
Privado	El usuario que crea el link. Ve los datos de la propiedad a través de: - DBA_DB_LINKS - ALL_DB_LINKS - USER_DB_LINKS	Crea el link en un esquema especifico de la base de datos local. Solo el dueño del link de base de datos privado o subprogramas de PL/SQL, en el esquema puede usar esta conexión para tener acceso a los objetos de base de datos en la correspondiente base de datos remota.
Público	El usuario llamado PUBLICO. Ve los datos de la propiedad a través de vistas mostradas anteriormente.	Crea un link amplio de base de datos. Todas las subrutinas de usuarios y PL/SQL en la base de datos pueden usar la conexión para acceder a los objetos de base de datos en la correspondiente base de datos remota.
Global	El usuario llamado PUBLICO. Ve los datos de la propiedad a través de vistas mostradas anteriormente.	Crea un amplio link de red. Cuando una red de Oracle usa nombres de Oracle, los servidores de nombres del sistema automáticamente crean y administran las conexiones globales para cada base de datos de Oracle en la red. Las subrutinas de usuarios y PL/SQL en cualquier base de datos pueden usar una conexión global para acceder a los objetos en la correspondiente base de datos remota.

Tabla 2. Tipos básicos de conexión para acceder a bases de datos remotas.

El tipo de conexión de base de datos a ser utilizado en una base de datos distribuida, depende de los requerimientos específicos de las aplicaciones que usa el sistema.

A continuación la tabla 3. indica las características de los tipos de conexión.

Tipo de Conexión	Características
Conexión privada de base de datos	Esta conexión es más segura que una pública o global, porque sólo el dueño de la conexión privada, o de los subprogramas dentro del mismo esquema, puede utilizar la conexión para acceder a la base de datos remota.
Conexión pública de base de datos	Cuando muchos usuarios necesitan un camino de acceso a una base de datos remota de Oracle, se puede crear una sola conexión pública de la base de datos para todos usuarios en una base de datos.
Conexión global de base de datos	Cuando una red de Oracle utiliza los nombres de Oracle, un administrador puede manejar convenientemente las conexiones globales de la base de datos para todas bases de datos en el sistema. La administración de la conexión de la base de datos es centralizada y sencilla.

Tabla 3. Características de los tipos de conexión

2.9.6. Conexión de usuarios de bases de datos

Al crear un link se determina qué usuario debe conectarse a la base de datos remota para acceder a los datos.

La Tabla 4. muestra las diferentes categorías de usuarios involucrados en una conexión de base de datos.

Tipo de usuario	Significado	Sintaxis de creación de conexión
Usuario conectado	Un usuario local que desea acceder a una conexión de base de datos sin especificar el nombre de usuario y contraseña. Si SYSTEM consigue acceso a un link público en una consulta, entonces el usuario conectado es SYSTEM, y Oracle conecta el esquema de SYSTEM en la base de datos remota. **Nota:** El usuario conectado no tiene que ser el usuario que creó la conexión, pero algún usuario puede acceder a la conexión.	CREATE PUBLIC DATABASE LINK op USING 'op';
Usuario actual	Un usuario global en una conexión de la base de datos CURRENT_USER. El usuario global debe ser autenticado por un certificado X.509 (un usuario SSL autentificado de la empresa) o una contraseña (una contraseña autentificada de la empresa), y es un usuario en ambas bases de datos implicadas en la conexión. Las conexiones de usuario actual son un aspecto de la seguridad avanzada de Oracle.	CREATE PUBLIC DATABASE LINK op CONNECT TO CURRENT_USER using 'op';
Usuario fijo	Un usuario cuyo nombre de usuario/contraseña forma parte de la conexión. Si una conexión incluye a un usuario fijo, entonces el nombre de usuario y la contraseña del usuario fijo se utiliza para conectar a la base de datos remota.	CREATE PUBLIC DATABASE LINK op CONNECT TO maría IDENTIFIED BY sol USING 'op';

Tabla 4. Diferencias de usuarios en una conexión de base de datos

Link de base de datos de usuarios conectados

Los usuarios conectados por medio de link no tienen una cadena de conexión asociada con ésta. La ventaja de un usuario conectado por medio de un link es que un usuario hace referencia al link y se conecta a la base de datos remota como si fuese el mismo

usuario. Debido a que ninguna cadena de conexión es asociada con el link, ninguna contraseña se almacena de forma clara en el diccionario de datos.

Los usuarios conectados por medio de link tienen algunas desventajas, porque estos links requieren usuarios para tener cuentas y privilegios sobre la base de datos remota, a la que se están intentando conectar, requieren de algunos privilegios administrativos.
Al dar más privilegios de los necesarios, los usuarios pueden violar los conceptos de seguridad, de tal forma que, solo se les debe dar privilegios que ellos necesitan para ejecutar sus trabajos.

Para usar un usuario conectado a una base de datos por medio de un link, depende de varios factores, el administrador entre ellos, ya sea que el usuario sea autentificado por Oracle usando una contraseña o externamente autentificado por el sistema operativo o un servicio de autentificación de red. Si el usuario es autentificado externamente, entonces, para usar un usuario conectado al link depende que la base de datos remota acepte la autentificación remota de los usuarios, en la que se coloco los parámetros de inicialización REMOTE_OS_AUTHENT.

La Tabla 5. indica la forma como opera el parámetro REMOTE_OS_AUTHENT.

Si REMOTE_OS_AUTHENT es	Entoces
Verdadero para la base de datos remota	Un usuario autenticado externamente puede conectarse a la base de datos remota usando un link de base de datos de usuario conectado.
Falso para la base de datos remota	Un usuario autentificado externamente no puede conectarse con la base de datos remota usando un link de la base de datos de usuario conectado al menos que utilice un protocolo seguro o un servicio de autentificación de red.

Tabla 5. Forma de operación del parámetro remote_os_authent

Link de base de datos de usuarios fijos

Un beneficio de un link de usuario fijo, es que éste se conecta a un usuario en una base de datos primaria, por medio de una base de datos remota con la misma seguridad del usuario especificado en la cadena de conexión.

Por ejemplo: el usuario local ***joel*** puede crear un link público de base de datos dentro del esquema de ***joel,*** que especifica el usuario fijo ***scott*** con la contraseña ***tiger***. Si ***María*** usa el link de usuario fijo en una consulta, entonces ***María*** es el usuario de la base de datos local, pero ella se conecta a la base de datos remota como ***scott/tiger***.

Los links de usuarios fijos tienen un nombre de usuario y una contraseña asociadas con la cadena de conexión. El nombre de usuario y la contraseña se almacenan de forma desencriptada en el diccionario de datos en la tabla *LINK$*.

El hecho que el nombre de usuario y la contraseña estén almacenados en forma desencriptada en el diccionario de datos, crea una debilidad potencial de seguridad en los links de base de datos fijos de usuario. Si el parámetro de la inicialización de O7_DICTIONARY_ACCESSIBILITY es TRUE, un usuario puede hacer un SELECT a cualquier tabla y el sistema de privilegios dará acceso al diccionario de datos, y la autentificación asociada a un usuario fijo estará comprometida. El valor por defecto para el parámetro de la inicialización O7_DICTIONARY_ACCESSIBILITY es FALSE.

Link de base de datos de usuarios actuales

Los link de usuarios actuales pueden hacer uso de un usuario global. Un usuario global puede estar autentificado por un certificado X.509 o una contraseña y es un usuario de ambas bases de datos involucradas en la conexión.

- El usuario que invoca la conexión CURRENT_USER no tiene que ser un usuario global.

2.9.7. Esquema de objetos y conexión de base de datos

Después que se ha creado una conexión de base de datos, ya se puede ejecutar declaraciones SQL que acceden a objetos de una base de datos remota. Por ejemplo,

para acceder a objetos remotos de empleados (***emp***) usando el link de base de datos ***foo***, la sentencia sería:

```
SELECT * FROM emp@foo;
```

El construir objetos bien formados por nombres usando links de base de datos es un aspecto esencial en la manipulación de datos en sistemas distribuidos.

Link de base de datos usando nombres de Esquema de Objetos

Oracle usa el nombre global de base de datos para nombrar el objeto del esquema global, usando el siguiente esquema:

```
esquema.esquema_objeto@nombre_global_base_datos
```

donde:

esquema, es una colección de estructuras lógicas de datos o esquema objetos. Un esquema es propiedad de un usuario de base de datos y tiene el mismo nombre como ese usuario. Cada usuario posee un solo esquema.

esquema_objeto, es una estructura lógica de datos enlazado a una tabla, índice, vista, sinónimo, procedimiento, paquete o link de base de datos.

nombre_global_base_datos, es el nombre que singularmente identifica una base de datos remota. Este nombre debe ser el mismo como el concatenado en los parámetros de inicialización de la base de datos remota DB_NAME y DB_DOMAIN, a menos que el parámetro GLOBAL_NAMES se ponga a FALSO, donde cualquier nombre es aceptado.

Por ejemplo, usando un link de base de datos a la base de datos ***ventas.division3.negocio.com***, un usuario o aplicación puede mencionar a los datos remotos como:

```
SELECT * FROM scott.emp@ventas.division3.negocio.com;
```

la tabla empleados (***emp***) dentro del esquema de ***scott***.

```
SELECT local FROM scott.dept@ventas.division3.negocio.com;
```

Si GLOBAL_NAMES se pone a FALSO, entonces se puede usar cualquier nombre para el link ***ventas.division3.negocio.com***. Por ejemplo, usted puede llamar al link de nombre ***foo*** luego usted pueda acceder a la base de datos remota como:

```
SELECT nombre FROM scott.emp@foo;
```

el nombre del link es diferente del nombre global.

Sinónimos para el esquema de objetos

Oracle permite crear sinónimos para poder ocultar el nombre de la conexión de base de datos del usuario. Un sinónimo permite acceder a una tabla o a una base de datos remota usando la misma sintaxis que se usa para acceder a una tabla o a una base de datos. Por ejemplo, suponer la siguiente consulta de una tabla en una base de datos remota:

```
SELECT * FROM emp@op.negocio.com;
```

Se puede crear el sinónimo ***emp*** para ***emp@op.negocio.com*** así que se podría hacer la siguiente consulta usando este sinónimo para acceder a los mismos datos:

```
SELECT * FROM emp;
```

Resolución de nombres del esquema de objetos

Para resolver aplicaciones referentes al esquema de objetos (*un proceso llamado resolución de nombres*), Oracle forma el nombre de objeto jerárquicamente. Por ejemplo, Oracle garantiza que cada esquema dentro de una base de datos tiene un único nombre y que dentro de un esquema igualmente el objeto tiene un único nombre. Como resultado, un nombre de un esquema de objeto es siempre único dentro de la base de datos. Además, Oracle resuelve aplicaciones referentes a un nombre de objeto local.

En una base de datos distribuida, un esquema de objeto, tal como una tabla, puede acceder a todas las aplicaciones del sistema. Oracle se refiere al modelo jerárquico como el nombre de bases de datos global para crear eficazmente nombres de objetos globales y resolver referencias a los objetos de esquema dentro del sistema de bases distribuidas. Por ejemplo, al hacer una consulta, ésta puede involucrar una tabla remota especificando su nombre e incluyendo la base de datos en la que reside.

Por ejemplo, suponer que se conecta a una base de datos local como el usuario ***SYSTEM***:

```
CONECT SYSTEM/contraseña@ventas1
```

Entonces el resultado de la siguiente declaración usando el link de base de datos ***op.negocio.com*** para acceder a los objetos del esquema tanto de ***scott*** como de ***María*** en la base de datos remota oficina principal (***op***) será:

```
SELECT * FROM scott.emp@op.negocio.com;
INSERT INTO maría.cuentas@op.negocio.com (numcuenta,
nombrecuenta, balance)
VALUES (5001, ' BOWER', 2000);
UPDATE maria.cuentas@op.negocio.com
SET balance = balance + 500;
DELETE FROM maria.cuentas@op.negocio.com
WHERE nombrecuenta = ' BOWER';
```

2.9.8. Restricciones de los link de base de datos

No se puede realizar las siguientes operaciones usando link de base de datos:

- Dar privilegios sobre objetos remotos.
- Ejecutar la operación DESCRIBE en algunos objetos remotos. Los siguientes objetos remotos soportan, a menudo, operaciones DESCRIBE: Tablas, Vistas, Procedimientos, Funciones.
- Analizar objetos remotos.
- Definir o hacer cumplir la integridad referencial.
- Dar roles a usuarios en una base de datos remota. Por ejemplo, si ***María*** se conecta a una base de datos local y ejecuta un procedimiento almacenado que usa un link de un usuario fijo, como ***scott*** entonces, ***María*** recibe roles que tiene por defecto ***scott*** para la base de datos remota. ***María*** no puede asignar un nuevo ROLE que no sea el que obtiene por defecto.
- Ejecutar consultas, joins que usen conexiones del servidor compartido.
- Utilizar un link de una cuenta de usuario que no esté autentificado a través de SSL, contraseña, o autenticación nativa NT.

2.10. Administración de base de datos distribuidas

2.10.1. Autonomía de sitio

Autonomía de sitio significa que cada servidor participante en una base de datos distribuida es un administrador independiente del resto de base de datos. Aunque varias bases de datos pueden trabajar juntas, cada base de datos es un almacén separado de datos que se maneja individualmente. Algunos de los beneficios de una autonomía de sitio en una base de datos distribuida son:

- Los nodos del sistema pueden reflejar la organización lógica de la compañía o los grupos que necesitan para mantener la independencia.
- Los administradores locales son los encargados de controlar los correspondientes datos locales. Por consiguiente, cada dominio del administrador de base de datos tiene una responsabilidad más pequeña y más manejable.
- Los fracasos independientes son menos probables que interrumpan a los otros nodos de la base de datos distribuida. Al surgir un fracaso de una base o un problema de cuello de botella no se requiere parar todas las operaciones para solucionarlo.
- Los administradores pueden recuperar otros nodos en el sistema independientemente de los fracasos aislados del sistema.

- Cada base de datos local tiene su propio diccionario de datos, así que no es necesario recurrir al catálogo global para acceder a los datos locales.
- Los nodos pueden mejorar el software independientemente.

Aunque Oracle permite administrar cada base de datos independientemente dentro de un sistema de base de datos distribuido, no se debe ignorar los requerimientos globales del sistema. Por ejemplo, se puede necesitar para:

- Crear cuentas adicionales de usuario en cada base de datos para dar soporte a los links creados para facilitar las conexiones de servidor a servidor.
- Poner adicionalmente los parámetros de inicialización tales como DISTRIBUTED_TRANSACTIONS y COMMIT_POINT_STRENGTH.

2.10.2. Seguridad de base de datos distribuidas

Oracle incluye todas las características de seguridad disponibles en un ambiente no distribuido dentro de un sistema de base de datos distribuido, incluyendo:

- Autentificación de contraseña para usuarios y roles.
- Algunos tipos de autentificación externa para usuarios y roles incluyendo:
    - Kerberos (*Subsistema de seguridad, supervisor*) versión 5 para usuarios conectados a través de links.
    - DCE para usuarios conectados a través de links.

- Paquetes de ingreso encriptados para conexiones, cliente a servidor y servidor a servidor.

Tópicos adicionales que se deben considerar al configurar un sistema de base de datos distribuido en Oracle:

- Autentificación de base de datos a través de links.
- Autentificación sin contraseñas.
- Soporte a cuentas de usuarios y roles.
- Usuarios centralizados y privilegios de administración.
- Encriptación de datos.

Autentificación de base de datos a través de links

Los links de base de datos pueden ser públicos o privados, autentificados o no autentificados. Al crear un link público debe especificar la palabra clave *PUBLIC*. Así por ejemplo:

```
CREATE PUBLIC DATABASE LINK foo USING  'ventas';
```

Un link autentificado se crea usando la cláusula CONNECT TO o la cláusula AUTHENTICATED BY o también usando ambas cláusulas en la creación de un link. Por ejemplo:

```
CREATE DATABASE LINK ventas CONNECT TO scott IDENTIFIED BY tiger USING 'ventas';
```

```
CREATE SHARED PUBLIC DATABASE LINK ventas CONNECT TO mick IDENTIFIED BY jagger AUTHENTICATED BY david IDENTIFIED BY bowie USING 'ventas';
```

La Tabla 6. describe como los usuarios se conectan a una base de datos remota a través de un link.

Tipo de Link	Autentificación	Acceso de Seguridad
Privado	No	Al conectarse a la base de datos remota, Oracle utiliza información de seguridad (nombre de usuario / contraseña) de la sesión local. De ahí, que la conexión es un link de base de datos de usuario. Las contraseñas se deben sincronizar en las dos bases de datos.
Privado	Si	El nombre de usuario / contraseña se toma de la definición de la conexión, antes que de la sesión local. De ahí, que la conexión de usuario es un link de base de datos fijo. Esta configuración permite que las contraseñas sean diferentes en las dos bases de datos, pero la contraseña de la conexión local de la base de datos debe ser igual a la contraseña de la base de datos remota. La contraseña se guarda en forma de texto en el catálogo del sistema local, lo cual es un riesgo de seguridad.
Público	No	Trabaja como una conexión privada no autentificada, sino que todos usuarios pueden hacer referencia a la base de datos remota.
Público	Si	Todos los usuarios en la base de datos local pueden acceder a la base de datos remota y todos usan el mismo nombre de usuario / contraseña para conectarse. También, la contraseña se guarda en forma de texto en el catálogo local, así que se puede ver la contraseña si se tienen los privilegios suficientes en la base de datos local.

Tabla 6. Conexión de usuarios a una base de datos remota mediante un link.

Autentificación sin contraseñas

Cuando se usa un usuario conectado o conexión actual de base de datos de usuario, se puede utilizar una autentificación externa, tal como los Kerberos, para obtener una seguridad de punta a punta. En la autentificación de punta a punta, las credenciales

pasan de un servidor a otro, y pueden ser autentificados por un servidor de base de datos que pertenece al mismo dominio.

Soporte a cuentas de usuarios y roles

En un sistema de base de datos distribuido, se debe plantear cuidadosamente la cuenta de usuario y los roles que son necesarios para que den soporte a aplicaciones que usan el sistema. Porque:

- La cuenta de usuario que se establece para las conexiones de servidor a servidor debe estar disponible en todas las bases de datos del sistema distribuido.
- Para que las aplicaciones estén disponibles en un sistema distribuido, se deben tener los roles y privilegios necesarios, y los usuarios deben estar presentes en todas las bases de datos del sistema distribuido.

Usuarios centralizados y privilegios de administración

Oracle proporciona diferentes vías para que se pueda administrar los usuarios y privilegios en un sistema distribuido. Por ejemplo, se tiene estas opciones:

- **Administración de usuarios empresariales**, se puede crear usuarios globales, quienes son autentificados a través de un SSL o por el uso de contraseñas, entonces se puede manejar a estos usuarios y sus privilegios dentro de un directorio a través de un servicio de directorio empresarial independiente.

- **Servicio de autentificación de red,** esta técnica simplifica la seguridad administrativa de ambientes distribuidos. Se puede usar las opciones de seguridad avanzada de Oracle para reforzar la red y la seguridad de sistemas de bases de datos distribuidos. La autentificación nativa de Windows NT es un ejemplo de una solución de autentificación que no es de Oracle.
- **Esquemas dependientes de usuarios globales,** una opción para centralizar usuario y administrar privilegios es crear lo siguiente:
    - Un usuario global dentro de un directorio centralizado.
    - Un usuario en cada base de datos al que el usuario global debe conectarse.

Por ejemplo, se puede crear un usuario global llamado ***freddy*** con la siguiente declaración SQL:

```
CREATE USER freddy IDENTIFIED GLOBALLY AS 'CN=freddy adams,0=Oracle,C=Inglaterra';
```

Esta solución permite a un solo usuario global ser autentificado por un directorio centralizado.

La solución de un esquema dependiente de usuarios globales, tiene como consecuencia que se debe crear un usuario llamado ***freddy*** en cada base de datos

a la que el usuario debe acceder. Esto se debe a que la mayoría de usuarios necesitan permiso para acceder a un esquema de la aplicación, pero no necesitan de su propio esquema. Debido a este problema, Oracle también soporta usuarios de esquemas independientes, que son usuarios globales que acceden a un solo esquema genérico en cada base de datos.

- **Esquemas independientes de usuarios globales,** Oracle brinda funcionalidad que permite a un usuario global ser administrado de manera centralizada por un servicio de directorio empresarial. Los usuarios que son administrados en el directorio empresarial se llaman usuarios empresariales o usuarios de la empresa. Este directorio contiene información de:

    - Los usuarios empresariales que pueden acceder a las bases de datos en un sistema distribuido.
    - Los roles que los usuarios de la empresa pueden usar dentro de cada base de datos.
    - Los usuarios empresariales que pueden conectarse a los esquemas de cada base de datos.

El administrador de cada base de datos no necesita crear una cuenta de usuario global para cada usuario empresarial dentro de cada base de datos a la que el usuario empresarial necesite conectarse. En cambio, los múltiples usuarios empresariales pueden conectarse al mismo esquema de base de datos, llamado ***esquema compartido*.**

Encriptación de datos

Permiten que los datos no puedan ser leídos o alterados. Protege los datos para que no sean vistos usando la seguridad de datos ASL (*Algoritmo de Seguridad de Lectura – RSA Read Security Algorithm*) o el algoritmo AEDE (*Algoritmo de Encriptación de Datos Estándar – DES Data Encriptation Standard*).

Para asegurar que los datos no sean modificados, borrados o reemplazados durante la transmisión, los servicios de seguridad de la opción avanzada de Oracle pueden generar un código de seguridad encriptado para incluirlo en cada paquete enviado a través de la red.

2.10.3. Auditar enlaces de base de datos

Se debe realizar una operación de auditoria local, es decir, si un usuario actúa en una base de datos local y accede a una base de datos remota a través de una conexión de base de datos, las acciones locales son auditadas en la base de datos local y las acciones remotas se auditan en la base de datos remota, ya que las opciones de auditoria son almacenadas en sus respectivas bases de datos.

La base de datos remota no puede determinar si una demanda de conexión ha sido exitosa o si las declaraciones subsecuentes de SQL vienen de otro servidor o de un cliente conectado localmente. Por ejemplo, asumir:

- El link de usuario fijo ***op.negocio.com*** conecta al usuario local ***María*** a la base de datos remota oficina principal (***op***) como un usuario remoto de ***scott***.
- El usuario ***scott*** es auditado en la base de datos remota.

Las acciones realizadas durante la sesión de la base de datos remota son auditadas como si ***Scott*** se hubiese conectado localmente a oficina principal (***op***) y ejecutado las acciones allí. Entonces se debe poner opciones de auditoria en la base de datos remota para capturar las acciones de los usuarios (*nombre de usuario*).

2.10.4. Herramientas administrativas

El administrador de base de datos puede escoger las herramientas que se van usar para administrar un sistema de base de datos distribuido en Oracle.

- **Administrador empresarial,** es una herramienta de administración de base de datos que proporciona una interface de usuario gráfica (IGU). El Administrador Empresarial mantiene la funcionalidad administrativa de las bases de datos distribuidas a través de una interface de fácil uso. Se puede usar para:

    - Administrar una sola base de datos o múltiples bases de datos simultáneamente.

- Centralizar las tareas de administración de base de datos. Se puede administrar al mismo tiempo las dos bases de datos, local y remota, que se encuentren ejecutándose en una plataforma Oracle y estén en cualquier parte del mundo.
- Ejecutar dinámicamente SQL, PL/SQL, y comandos de administración empresarial, se puede usar el administrador empresarial para entrar, editar y ejecutar declaraciones.
- Administrar las características de seguridad tales como usuarios globales, roles globales y los servicios del directorio empresarial.

- **Soporte SNMP (Simple Network Management Protocol),** además de las capacidades de administración. Esta herramienta permite que el servidor de base de datos Oracle, pueda ser localizado y consultado por cualquier sistema administrador de red basado en SNMP. SNMP es un estándar popular aceptado por muchos sistemas administradores de red.

2.11. Procesando transacciones en un sistema distribuido

Una transacción es una unidad lógica de trabajo, constituida por uno o más declaraciones SQL ejecutadas por un simple usuario. Una transacción empieza con la ejecución de la primera declaración SQL del usuario, y termina cuando este hace un commit o rollback.

Una **transacción remota** contiene sólo declaraciones que acceden a un simple nodo remoto. Una **transacción distribuida** contiene declaraciones que acceden a más de un nodo.

2.11.1. Declaraciones sql remotas

Una consulta remota es la que selecciona información de una o más tablas remotas, todas estas residen en un mismo nodo remoto. Por ejemplo, la siguiente consulta accede a los datos de la tabla departamentos (***dept***) en el esquema ***scott*** de la base de datos remota ***ventas***.

```
SELECT *  FROM
scott.dept@ventas.usa.americas.auto_negocio.com;
```

Una declaración de actualización remota es la que modifica los datos en una o más tablas, todas estas tablas igualmente se encuentran localizadas en un mismo nodo remoto. Por ejemplo, la siguiente consulta actualiza la tabla departamentos (***dept***) dentro del esquema ***scott*** de la base de datos remota ***ventas***.

```
UPDATE scott.dept@marketing.usa.americas.auto_negocio.com
SET local = 'NEW YORK'
WHERE numdept = 10;
```

2.11.2. Declaraciones sql distribuidas

Una consulta distribuida recupera información de dos o más nodos. Por ejemplo: La siguiente consulta accede a los datos de la base de datos local así como de la base de datos remota ***ventas***.

```
SELECT nombreempleado, nombredept
FROM scott.emp e,
scott.dept@ventas.usa.americas.auto_negocio.com d
WHERE e.numdept = d.numdept;
```

Una declaración de actualización distribuida modifica los datos en dos o más nodos. En una actualización distribuida es posible usar subprogramas PL/SQL tales como procedimientos, triggers que pueden incluir dos o más actualizaciones remotas que acceden a datos de diferentes nodos. Por ejemplo: La siguiente unidad de programa PL/SQL actualiza tablas en la base de datos local y la base de datos remota ***ventas***.

```
BEGIN
UPDATE scott.dept@ventas.usa.americas.auto_negocio.com
SET local = 'NEW YORK'
WHERE numdept = 10;
UPDATE scott.emp
SET numdept = 11
WHERE numdept = 10;
END;
COMMIT;
```

Oracle envía declaraciones en el programa para los nodos remotos e indica si su ejecución tiene éxito o fracaso como unidad.

2.11.3. Transacciones remotas

Contiene una o más declaraciones remotas, cada una de las cuales hace referencia a un simple nodo remoto. Por ejemplo: La siguiente transacción contiene dos declaraciones cada una de las cuales acceden a la base de datos remota ***ventas***.

```
UPDATE scott.dept@ventas.usa.americas.auto_negocio.com
SET local = 'NEW YORK'
WHERE numdept = 10;
UPDATE scott.emp@ventas.usa.americas.auto_negocio.com
SET numdept = 11
WHERE numdept = 10;
COMMIT;
```

2.11.4. Transacciones distribuidas

Es una transacción que incluye una o más declaraciones, que individualmente o en grupo, actualizan datos de dos o más nodos distintos de una base de datos distribuida. Por ejemplo, la siguiente transacción actualiza la base de datos local y la base de datos remota ***ventas***:

```
UPDATE scott.dept@ventas.usa.americas.auto_negocio.com
SET local = 'NEW YORK'
WHERE numdept = 10;
UPDATE scott.emp
SET numdept = 11
WHERE numdept = 10;
COMMIT;
```

2.11.5. Mecanismo commit de dos fases

Una base de datos debe garantizar que todas las declaraciones en una transacción, distribuida o no distribuida, no comprometan un commit o un rollback como unidad.

Los efectos de una transacción progresiva deben ser invisibles para el resto de las transacciones en todos los nodos; esta transparencia debe ser verdadera para transacciones que incluyen cualquier tipo de operación, incluso consultas, actualizaciones o llamadas a procedimientos remotos.

En una base de datos distribuida, Oracle debe coordinar el control de transacciones con las características de la red y mantener la consistencia de los datos, aún si el sistema o la red tengan fallos.

El mecanismo commit de dos fases garantiza que todos los servidores que estén participando en una transacción distribuida, cumplan con un commit o rollback de las

declaraciones en la transacción. Este mecanismo también protege implícitamente operaciones de integridad, llamadas a procedimientos remotos y triggers.

2.11.6. Resolución de nombres para enlaces de base de datos

Un nombre de objeto global es un objeto específico usado para la conexión de base de datos. Los componentes principales de un nombre de objeto global son:

- Nombre del objeto
- Nombre de la base de datos
- Dominio

La Tabla 7. muestra los componentes de un nombre de objeto de base de datos global especificado explícitamente:

Declaración	Objeto	Base de datos	Dominio
SELECT * FROM juan.dept@ventas.negocio.com	dept	Ventas	negocio.com
SELECT * FROM emp@marketing.usa.negocio.com	emp	Marketing	usa. negocio.com

Tabla 7. Componentes de un nombre de objeto de base de datos global

Siempre que una declaración SQL incluya una referencia a un nombre del objeto global, Oracle busca un link de base de datos que haga juego con el nombre de base especificada en el nombre del objeto global.

Por ejemplo, en la siguiente declaración:

```
SELECT * FROM scott.emp@ordenes.usa.negocio.com;
```

Oracle busca un link de base de datos llamado ***ordenes.usa.negocio.com***. Oracle realiza estas operaciones para determinar el camino a la base de datos especificada.

Resolución de nombres cuando el nombre de la base de datos global es completo

Asumir que la siguiente declaración SQL, especifica un nombre de base de datos global completo.

```
SELECT * FROM emp@prod1.usa.oracle.com
```

En este caso, los dos nombres de base de datos productos1 (***prod1***) y los componentes del dominio (***usa.oracle.com***) son específicos, así que Oracle busca link de base de datos privados, públicos y globales. Oracle busca sólo en links específicos que hagan juego con el nombre de base de datos global.

Resolución de nombres cuando el nombre de la base de datos global es parcial

Si cualquier parte del dominio se especifica, Oracle asume que el nombre de base de datos global se especificó completamente. Si una declaración SQL específica parte del nombre de una base de datos global (*es decir, solo sé específica el componente de la base de datos*), Oracle añade el valor en el parámetro de inicialización DB_DOMAIN a los valores en el parámetro de inicialización BD_NAME para construir un nombre completo. Por ejemplo, asumir las siguientes declaraciones:

```
CONNECT scott/tiger@localbd
SELECT * FROM scott.emp@ordenes;
```

Si el dominio de la red para ***localbd*** es ***usa.negocio.com***, entonces Oracle añade este dominio a ***ordenes*** para construir el nombre completo de la base de datos global ***ordenes.usa.negocio.com***. Oracle busca el link de la base de datos para el nombre de la base de datos que fue construido. Si un link no es encontrado, Oracle devuelve un error y la declaración SQL no puede ser ejecutada.

Resolución de nombres cuando el nombre de la base de datos global no se especifica

Si un nombre de objeto global hace referencia a un objeto en la base de datos local y el nombre del link de la base de datos no se especifica usando el símbolo **@**, entonces Oracle automáticamente detecta que el objeto es local y no busca o usa el link de base de datos para resolver el objeto al que se hace referencia. Por ejemplo, en la siguiente declaración:

```
CONNECT scott/tiger@localbd
SELECT * from scott.emp;
```

Debido a que en la segunda declaración no se especifica el nombre de la base de datos global usado en la cadena de conexión del link de la base de datos, Oracle no busca en los links de base de datos.

Terminando la búsqueda de resolución de nombres

Oracle no necesariamente detiene la búsqueda de los links de la base de datos cuando encuentra la primera pareja. Oracle sigue buscando parejas de links de base de datos sean estas privadas, públicas, globales o de red, mientras determina un camino completo a la base de datos remota (*las dos en una cuenta remota y nombre del servicio*). La primera pareja determina el esquema remoto tal como se ilustra en la Tabla 8.

Si	Entoces Oracle	Por ejemplo
No específica la cláusula CONNECT	Usa un link de base de datos de usuario conectado.	CREATE DATABASE LINK k1 USING 'prod'
Específica la cláusula CONNECT TO ... IDENTIFIED BY	Usa un link de base de datos de usuario fijo.	CREATE DATABASE LINK k2 CONNECT TO scott IDENTIFIED BY Tiger USING 'prod'
Específica la cláusula CONNECT TO CURRENT_USER	Usa un link de base de datos de usuario actual.	CREATE DATABASE LINK k3 CONNECT TO CURRENT_USER USING 'prod'
No específica la cláusula USING	Busca hasta encontrar una cadena de conexión especifica de base de datos. Si se emparejan las conexiones de la base de datos se ha encontrado el link y si no es encontrada Oracle devuelve un error.	CREATE DATABASE LINK k4 CONNECT TO CURRENT_USER

Tabla 8. Determinación de la primera pareja en el esquema remoto

Después que Oracle determina un camino completo, crea una sesión remota asumiendo que una conexión idéntica no existe en la misma sesión local. Si una sesión ya existe, Oracle vuelve a usarla.

2.11.7. Resolución de nombres del esquema de objeto

Luego que la base de datos Oracle se conecta a la base de datos remota; especificada por el nombre del usuario local quien hizo la declaración SQL, la resolución del objeto continúa como si el usuario remoto habría hecho la declaración SQL asociada. La primera pareja determina el esquema remoto según las reglas que muestra la tabla 9.

Si utiliza	Entonces la resolución del objeto avanza
Un link de base de datos de usuario fijo	Al esquema especificado en la creación de la declaración del link
Un link de base de datos de usuario conectado	Al esquema remoto de los usuarios conectados
Un link de base de datos de usuario actual	Al esquema de los usuarios actuales

Tabla 9. Reglas del esquema remoto para la primera pareja

Si Oracle no encuentra el objeto, entonces verifica los objetos públicos de la base de datos remota. Si no puede resolver el objeto, entonces la sesión remota establecida permanece pero la declaración SQL no puede ejecutarse y retorna un error.

2.11.8. Resolución de nombres globales en vistas, sinónimos, y procedimientos

Una vista, sinónimo o una unidad de programa PL/SQL (*por ejemplo, un procedimiento, función o trigger*) pueden referenciar a un objeto del esquema remoto por el nombre de objeto global. Si el nombre del objeto global está completo, Oracle almacena la definición del objeto sin extender el nombre del objeto global. Si el nombre es parcial, Oracle extiende el nombre usando el nombre del dominio de la base de datos local.

La Tabla 10. explica cuando Oracle completa la extensión de un nombre de objeto global parcial en vistas, sinónimos y unidades de programa:

Si	Entonces oracle
Crea una Vista	No extiende los nombres globales parciales. El diccionario de datos almacena el texto exacto definido en la consulta. Oracle extiende un nombre de objeto global parcial cada vez que use la vista.
Crea un Sinónimo	Extiende los nombres globales parciales. La definición del sinónimo se almacena en el diccionario de datos incluyendo el nombre del objeto global expandido.
Compila una unidad de programa	Extiende los nombres globales parciales.

Tabla 10. Extensión de un nombre de objeto global parcial

¿Qué sucede cuando se cambia el nombre global?

Cambiando el nombre global se puede afectar a vistas, sinónimos y procedimientos que hacen referencia a los datos remotos usados por los nombres de objetos globales parciales. Si el nombre global al que se hace referencia la base de datos cambia, las vistas y procedimientos hacen referencia a una base de datos no existente o incorrecta. Por otra parte, los sinónimos no extienden las rutinas para los nombres de links de la base de datos, así que estos no cambian.

2.12. Desarrollo de aplicaciones de bases de datos distribuidas

El desarrollo de aplicaciones en un sistema distribuido abarca problemas que no son aplicables en un sistema no distribuido.

2.12.1. Transparencia en un sistema de base de datos distribuido

Se pueden desarrollar aplicaciones transparentes para los usuarios que trabajan en el sistema, en un sistema de base de datos distribuido Oracle. La meta de la transparencia es hacer a un sistema de base de datos distribuido parecer como si se tratara de una sola base de datos de Oracle, sino el sistema no desarrollaría y los usuarios del sistema con complejidad podrían desarrollar aplicaciones de base de datos distribuidas restringiendo la productividad de los usuarios.

Ubicación de la transparencia

Un sistema de base de datos distribuido en Oracle tiene características que permiten desarrollar y administrar aplicaciones, ocultando la ubicación física de los objetos en la base de datos de aplicaciones y usuarios. La ubicación de la transparencia existe cuando un usuario puede referirse universalmente a un objeto de la base de datos, como una tabla, sin tener en cuenta el nodo al que una aplicación se conecta. Los beneficios de la ubicación de la transparencia incluye:

- Acceder a los datos remotos es sencillo, porque los usuarios de la base de datos no necesitan saber la ubicación física de los objetos en la base de datos.
- Los administradores pueden mover objetos de la base de datos sin ningún impacto en los usuarios finales ni en las aplicaciones existentes en la base de datos.

Los administradores y desarrolladores utilizan sinónimos para establecer la ubicación de transparencia en tablas y objetos secundarios, en un esquema de aplicación. Los usuarios y desarrolladores pueden utilizar también vistas y procedimientos almacenados para establecer la ubicación de transparencia de aplicaciones que trabajen en un sistema de base de datos distribuido.

Transparencia de sql y commit

La arquitectura de distribución de base de datos proporciona consultas, actualizaciones, y transparencia de transacciones. Por ejemplo, declaraciones estándar SQL tales como SELECT, INSERT, UPDATE, DELETE trabajan como en un ambiente de base de datos no distribuido. Las declaraciones estándar SQL que controlan la aplicación de transacciones tales como COMMIT, SAVEPOINT, y ROLLBACK no tienen requisitos de programación compleja o funciones especiales para proporcionar el control de las transacciones distribuidas.

- Las declaraciones en una transacción pueden hacer referencia a un número de tablas local o remotamente.
- Oracle garantiza que todos los nodos implicados en una transacción distribuida realicen la misma acción: todos hacen un Commit o todos hacen un Rollback en la transacción.
- Si ocurre un fracaso en la red o en el sistema durante un Commit, en una transacción distribuida, la transacción es resuelta globalmente automática y

transparentemente. Específicamente, cuando la red o el sistema se restauren, o todos los nodos hacen un Commit, o todos los nodos hacen un rollback a la transacción.

Internamente en Oracle, cada transacción realizada tiene un número asociado llamado **SCN** (*System Change Number, NCS Número de cambio de Sistema*) en el cual el sistema identifica los cambios realizados por las declaraciones en una transacción. En una base de datos distribuida, el SCN coordina y comunica nodos.

2.12.2. Llamadas a procedimientos remotos (Remote procedure calls RPC's)

Las aplicaciones pueden hacer llamadas a procedimientos locales para realizar el trabajo en la base de datos local y llamadas a procedimientos remotos (RPC's) para realizar el trabajo en una base de datos remota.

Cuándo un programa llama un procedimiento remoto, el servidor local pasa todos los parámetros del procedimiento al servidor remoto en la llamada. Por ejemplo, la siguiente unidad de programa PL/SQL llama al procedimiento pasa empleado (***pasa_emp***) ubicado en la base de datos remota ***ventas*** con el parámetro ***1257***:

```
BEGIN
emp_direccion.pasa_emp@ventas.usa.americas.auto_negocio.com
(1257);
END;
```

En la orden del RPC, la llamada al procedimiento puede existir en el sitio remoto, y el usuario a ser conectado debe tener los privilegios apropiados para ejecutar el procedimiento.

2.12.3. Optimización de consultas distribuidas

Es una característica de Oracle que reduce la cantidad de datos que se requiere entre los sitios, cuando una transacción recupera los datos de tablas remotas referenciadas en una declaración SQL distribuida.

Oracle optimiza costos mediante las consultas distribuidas para encontrar o generar expresiones SQL que extraigan solo lo necesario de tablas remotas, procesando los datos en sitios remotos o en sitios locales, y envía los resultados al sitio local para el procesamiento final. Este funcionamiento reduce la cantidad de datos requeridos y el tiempo para transferir todos los datos de la tabla al sitio local.

2.13. Soporte del juego de caracteres para ambientes distribuidos

Oracle brinda soporte para ambientes de clientes y servidores de base de datos Oracle y servidores que no son Oracle, que usan juegos diferentes de caracteres. NCHAR brinda soporte a ambientes heterogéneos, Se puede tener una variedad de soporte para ambientes como NLS (*National Language Support – SLN Soporte para Lenguaje*

Nacional) y de servicios heterogéneos (*SH*), variables y parámetros de inicialización para controlar la conversión de datos entre los diferentes juegos de caracteres.

La configuración de los caracteres son definidos por NLS y los parámetros SH como muestra la tabla 11.

Parámetros	Ambiente	Definido para
NLS_LANG (variable de ambiente)	Cliente-Servidor	Cliente
NLS_LANGUAGE NLS_CHARACTERSET NLS_TERRITORY	Cliente-Servidor No Heterogéneo Distribuido Heterogéneo Distribuido	Servidor de base de datos Oracle
HS_LANGUAGE	Heterogéneo Distribuido	Servidor no Oracle Gateway Transparente
NLS_NCHAR (variable de ambiente) HS_NLS_NCHAR	Heterogéneo Distribuido	Servidor de base de datos Oracle Gateway Transparente

Tabla 11. Configuración de caracteres

2.13.1. Ambiente cliente/servidor

En un ambiente cliente servidor, el juego de caracteres del cliente debe ser igual a un subconjunto del juego de caracteres del servidor en la base de datos Oracle. Como muestra la figura 6. por ejemplo.

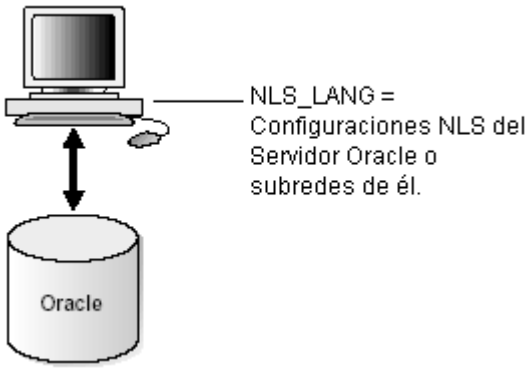


Figura 6. Parámetros de configuración NLS en un ambiente cliente-servidor

2.13.2. Ambiente distribuido homogéneo

En un ambiente homogéneo, el juego de caracteres del cliente debe ser igual al juego de caracteres del servidor o ser igual a un subconjunto del juego de caracteres del servidor principal, así como muestra la figura 7.

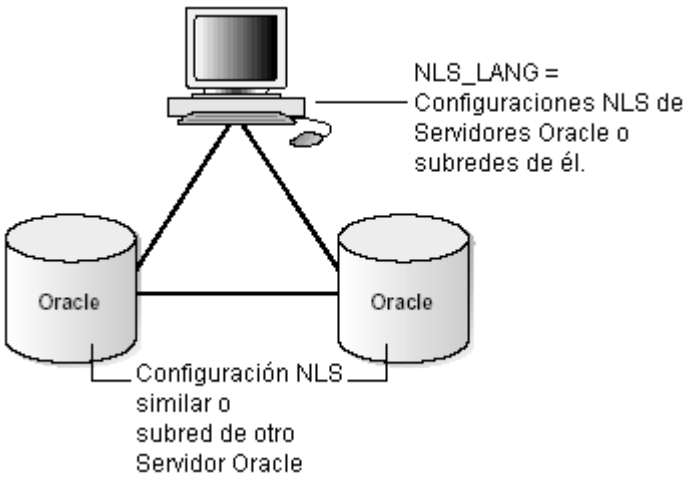


Figura 7. Parámetros de configuración NLS en un ambiente homogéneo

2.13.3. Ambiente distribuido heterogéneo

En un ambiente heterogéneo, la configuración NLS del cliente, el gateway transparente, y la fuente de datos que no son Oracle deben ser igual a un subconjunto del juego de caracteres del servidor de base de datos Oracle, como en la figura 8. El gateway transparente tiene soporte completo de globalización.

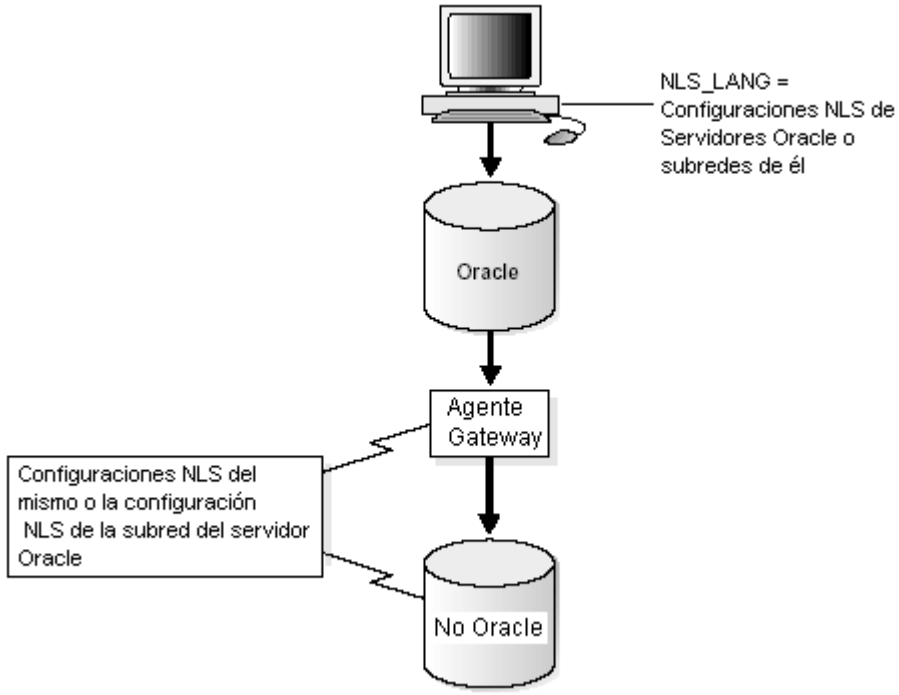


Figura 8. Parámetros de configuración NLS en un ambiente heterogéneo.

En un ambiente heterogéneo, solo el gateway transparente con los servicios heterogéneos SH tienen soporte completo con las capacidades de NCHAR.

CAPITULO III

MANEJO DE UNA BASE DE DATOS DISTRIBUIDA

3.1. Manejar nombres globales en un sistema distribuido

En un sistema de base de datos distribuido, cada base de datos debe tener un único **nombre de base de datos global** que lo identifique. Una tarea primaria de la administración en un sistema distribuido es manejar la creación y la modificación de los nombres de base de datos globales.

3.1.1. Formar nombres de la base de datos global

Un nombre de base de datos global esta formado por dos componentes: un nombre de la base de datos y un dominio. El nombre de la base de datos y el nombre del dominio son determinados por los parámetros de inicialización en la creación de la base de datos como lo muestra la tabla 12.

Componente	Parámetro	Requisitos	Ejemplo
Nombre de la base de datos	DB_NAME	Deben ser ocho caracteres o menos	Ventas
Dominio que contiene la base de datos	DB_DOMAIN	Debe seguir los estándares del Internet. Los niveles en nombres de dominio deben ser separados por puntos y la orden de nombres de dominio es desde la hoja hasta la raíz y de izquierda a derecha.	usa.negocio.com

Tabla 12. Parámetros de inicialización de base de datos

La tabla 13. muestra ejemplos de nombres globales válidos de base de datos:

DB_NAME	DB_DOMAIN	Global Database Name
Ventas	ec.oracle.com	ventas.ec.oracle.com
Ventas	usa.oracle.com	ventas.usa.oracle.com
Marketing	usa.oracle.com	marketing.usa.oracle.com
Nomina	nolucrativo.org	nomina.nolucrativo.org

Tabla 13. Ejemplos de nombres globales válidos

El parámetro de la inicialización DB_DOMAIN es importante en la creación de base de datos y junto con el parámetro DB_NAME, para formar el nombre de la base de datos global. El nombre de la base de datos global se almacena en el diccionario de datos. Para cambiar el nombre global se utiliza la declaración ALTER DATABASE y no se debe alterar el parámetro DB_DOMAIN en el parámetro de inicialización del archivo.

3.1.2. Determinar si el nombre global cumple los requisitos

El nombre que se dé a una conexión de base de datos local depende que la base de datos remota, a la que se desee acceder, cumpla con el nombre global. Si la base de datos remota cumple con el nombre global, entonces se debe usar el nombre de la base de datos global remota como el nombre de la conexión. Por ejemplo, si se conecta al servidor local oficina principal (***op***) y se quiere crear un link a la base de datos remota fábrica (***fa***), y fábrica (***fa***) cumple con el nombre global, entonces se debe utilizar el nombre de la base de datos global fábrica (***fa***) como nombre de la conexión.

Se puede utilizar también los nombres de servicio como parte del nombre de la conexión de base de datos. Por ejemplo, si se utiliza los nombres de servicio ***ns1*** y ***ns2*** para conectarse a la base de datos ***op.negocio.com*** y el nombre global oficina principal (***op***) cumple con el nombre global, entonces se puede crear los siguientes nombres de conexión para ***op***:

- OP.NEGOCIO.COM@NS1
- OP.NEGOCIO.COM@NS2

Para determinar si el nombre de una base de datos global cumple, se examina el archivo de los parámetros de inicialización de la base de datos o se consulta la vista V$PARAMETER. Por ejemplo, para ver si el nombre global cumple en fábrica (***fa***), se

podría empezar una sesión en fábrica (***fa***) y luego crear y ejecutar el siguiente script nombresglobales.sql (el ejemplo incluye la salida):

```
COL NAME FORMAT A12
COL VALUE FORMAT A6
SELECT NAME, VALUE FROM V$PARAMETER
WHERE NAME = 'nombres_globales'
/

SQL> @nombresglobales

NAME                         VALUE
---------------------        ----------
nombres_globales             FALSE
```

3.1.3. Ver el nombre de la base de datos global

Para ver el nombre de la base de datos global, se utiliza la vista NOMBRE_GLOBAL del diccionario de datos. Por ejemplo:

```
SELECT * FROM NOMBRE_GLOBAL;

NOMBRE_GLOBAL
--------------------------------
VENTAS.EC.ORACLE.COM
```

3.1.4. Cambiar el dominio de un nombre de base de datos global

La sentencia ALTER DATABASE permite cambiar el dominio de un nombre de base de datos global. Después que la base de datos es creada, cambiando el parámetro de inicialización DB_DOMAIN no tiene efecto en el nombre de la base de datos global o en la resolución de los nombres de los links de la base de datos.

El siguiente ejemplo muestra la declaración para cambiar el nombre, donde la base de datos es un nombre de base de datos y el dominio es el dominio de la red:

```
ALTER DATABASE RENAME NOMBRE_GLOBAL TO
BaseDeDatos.Dominio;
```

Se utiliza el siguiente procedimiento para cambiar el dominio en un nombre de base de datos global:

1. Determinar el nombre actual de la base de datos global. Por ejemplo:

   SELECT * FROM NOMBRE_GLOBAL;

   NOMBRE_GLOBAL

   VENTAS.EC.ORACLE.COM

2. Cambiar el nombre de la base de datos global usando la declaración ALTER DATABASE. Así:

   ALTER DATABASE RENAME NOMBRE_GLOBAL TO

   ventas.usa.oracle.com;

3. Consultar la tabla NOMBRE_GLOBAL para verificar el nuevo nombre. Por ejemplo:

   SELECT * FROM NOMBRE_GLOBAL;

   NOMBRE_GLOBAL

   VENTAS.USA.ORACLE.COM

3.2. Crear links de base de datos

Para soportar aplicaciones de acceso a datos y objetos de esquema a través de un sistema distribuido de base de datos, se deben crear todas las conexiones necesarias a la base de datos.

3.2.1. Obtener privilegios necesarios para crear links de base de datos

Un link de base de datos es un puntero en la base de datos local que permite tener acceso a objetos dentro de una base de datos remota. La Tabla 14. ilustra que privilegios requiere una base de datos para crear los diferentes tipos de link:

Privilegio	Base de datos	Requerido para
CREATE DATABASE LINK	Local	La creación de un link privado de base de datos.
CREATE PUBLIC DATABASE LINK	Local	La creación de un link público de base de datos.
CREATE SESSION	Remota	La creación de cualquier tipo de link de la base de datos.

Tabla 14. Requerimientos para crear links de base de datos

Para ver los privilegios con los que se cuenta, se puede realizar una consulta a la tabla ROLE_SYS_PRIVS. Por ejemplo, se podría crear y ejecutar el siguiente script privilegios.sql.

```
SELECT DISTINCT PRIVILEGE AS "Database Link Privileges"
FROM ROLE_SYS_PRIVS
WHERE PRIVILEGE IN ( 'CREATE SESSION','CREATE DATABASE
LINK',
'CREATE PUBLIC DATABASE LINK')
/

SQL> @privs

Database Link Privileges
------------------------------------------------
CREATE DATABASE LINK
CREATE PUBLIC DATABASE LINK
CREATE SESSION
```

3.2.2. Especificar los tipos de links

Cuando se crea un link de base de datos, se debe decidir quién tendrá acceso al mismo. Los tres tipos básicos de links son: Privado, Público, y Global.

Crear un link de base de datos privado

Formato:

```
CREATE DATABASE LINK nombre_link...;
```

nombre_link, es el nombre global de la base de datos o un nombre de link arbitrario.

La tabla 15. muestra ejemplos de links de base de datos privados.

Declaración SQL ...	Crea …
CREATE DATABASE LINK sumi.usa.negocio.com;	Un link privado que utiliza el nombre global de la base de datos a la base de datos remota suministros (**sumi**). El link utiliza el nombre de usuario/contraseña del usuario conectado. Así que si scott (identificado por tiger) utiliza el link para una consulta, el link establece una conexión a la base de datos remota como scott/tiger.
CREATE DATABASE LINK link_2 CONNECT TO maría IDENTIFIED BY sol USING 'usa_sumi';	Un link de usuario fijo privado llamado link_2 a la base de datos con el nombre de servicio usa_sumi. El link conecta a la base de datos remota con el nombre de usuario/contraseña de maria/sol a pesar del usuario conectado.
CREATE DATABASE LINK link_1 CONNECT TO CURRENT_USER USING 'usa_sumi';	Un link privado llamado link_1 a la base de datos con el nombre de servicio usa_sumi. El link utiliza el nombre de usuario/contraseña del usuario actual para apuntar a la base de datos remota. **Nota**: El usuario actual no puede ser igual que el usuario conectado, y debe ser un usuario global en ambas bases de datos implicadas en la conexión.

Tabla 15. Ejemplos de links de base de datos privados

Crear un link de base de datos público

Formato:

```
CREATE PUBLIC DATABASE LINK nombre_link...;
```

La tabla 16. muestra ejemplos de links de base de datos públicos.

Declaración SQL ...	Crea …
CREATE PUBLIC DATABASE LINK sumi.usa.negocio.com;	Un link público a la base de datos remota suministros (**sumi**). El link usa el nombre de usuario/contraseña del usuario conectado. Si Scott (identificado por tiger) utiliza el link en una consulta, el link establece una conexión a la base de datos remota como scott/tiger.
CREATE PUBLIC DATABASE LINK pu_link CONNECT TO CURRENT_USER USING 'sumi';	Un link público llamado pu_link a la base de datos con el nombre de servicio suministros (**sumi**). El link utiliza el nombre de usuario/contraseña del usuario actual para apuntar a la base de datos remota. **Nota** : El usuario actual no puede ser igual que el usuario conectado, y debe ser un usuario global en ambas bases de datos implicadas en el link.
CREATE PUBLIC DATABASE LINK ventas.usa.negocio.com CONNECT TO maría IDENTIFIED BY sol;	Un link de usuario fijo público a la base de datos remota **ventas**. El link conecta a la base de datos remota con el nombre de usuario/contraseña de maría/sol.

Tabla 16. Ejemplos de links de base de datos públicos

Crear un link de base de datos global

Los link de base de datos globales se definen en el servidor de nombres de Oracle.

3.2.3. Especificar usuarios de links

Un link de base de datos, define un camino de comunicación desde una base de datos a otra. Cuando una aplicación usa un link de base de datos para acceder a una base de

datos remota, Oracle establece una sesión de base de datos en la base de datos remota a favor de la aplicación local.

Cuando se crea links de base de datos privados o públicos, se puede determinar en que esquema de la base de datos remota, el link establecerá conexiones, para crear un usuario fijo o un actual y conectar al link de base de datos del usuario.

Crear links de base de datos de usuarios fijos

Para crear un link de base de datos de usuario fijo, se debe ingresar la credencial (*nombre de usuario / contraseña*) requerida para tener acceso a la base de datos remota en la definición del link.

Formato:

```
CREATE DATABASE LINK … CONNECT TO nombre_usuario
IDENTIFIED BY contraseña…;
```

La tabla 17. muestra ejemplos de links de base de datos de usuarios fijos.

Declaración SQL...	Crea...
CREATE PUBLIC DATABASE LINK sumi.usa.negocio.com CONNECT TO scott AS tiger;	Un link público usando el nombre global de la base de datos a la base de datos remota suministros (**sumi**). El link se conecta a la base de datos remota con el nombre de usuario/contraseña de scott/tiger.
CREATE DATABASE LINK foo CONNECT TO maría IDENTIFIED BY sol USING 'finanzas';	Un link de usuario fijo llamado **foo** a la base de datos con el nombre de servicio **finanzas**. El link se conecta a la base de datos remota con el nombre de usuario/contraseña de maría/sol.

Tabla 17. Ejemplos de links de base de datos de usuarios fijos

Cuándo una aplicación usa un link de base de datos de usuario fijo, el servidor local siempre establece una conexión a un esquema remoto fijo en la base de datos remota. El servidor local también envía la credencial del usuario fijo a través de la red cuando una aplicación utiliza el link para conseguir acceso a la base de datos remota.

Crear links de base de datos de usuarios actuales y usuarios conectados

Los links de base de datos de usuarios actuales y usuarios conectados no incluyen credenciales en la definición del link. La credencial utilizada para conectarse a la base de datos remota puede cambiar dependiendo del usuario al que se refiere el link y las operaciones ejecutadas por la aplicación.

Crear un link de base de datos de usuarios conectados

Para crear este tipo de link, se omite la cláusula CONNECT TO.

Formato:

```
CREATE [SHARED] [PUBLIC] DATABASE LINK linkbd …
[USING 'nombre_servicio_red'];
```

linkbd, es el nombre del link y nombre_servicio_red es una cadena de conexión opcional. Por ejemplo:

```
CREATE DATABASE LINK ventas.division3.negocio.com
USING 'ventas';
```

Crear un link de base de datos de usuarios actuales

Formato:

```
CREATE [SHARED] [PUBLIC] DATABASE LINK linkbd
CONNECT TO CURRENT_USER [USING 'nombre_servicio_red'];
```

Por ejemplo, para crear un link de base de datos de usuario conectado para la base de datos ***ventas***, su sintaxis sería:

```
CREATE DATABASE LINK ventas CONNECT TO
CURRENT_USER USING 'ventas';
```

Para utilizar un link de base de datos de usuario conectado, el usuario actual debe ser un usuario global de ambas bases de datos implicadas en la conexión.

3.2.4. Usar calificadores de conexión para especificar nombres de servicio dentro de los nombres de link

En algunas situaciones, se quisiera tener varios links de base de datos del mismo tipo (*por ejemplo, públicos*) para una misma base de datos remota y establecer la conexión a esa base de datos remota usando diferentes caminos. Algunos casos donde esta estrategia es útil son:

- Una base de datos remota es parte de una Configuración Real de Aplicaciones Clusters de Oracle, así que se debe definir varios link de base de datos públicos en su nodo local para que los links puedan ser establecidos en instancias específicas de la base de datos remota.
- Algunos clientes se conectan al Servidor de Oracle usando TCP/IP mientras otros usan DECNET (*Protocolo de Comunicación de datos de Digital Equipment Corporation*).

Al crear un link de base de datos, un nombre del servicio es especificado como una porción del nombre del link de la base de datos, separado por un signo @, como en ***@ventas***. Esta cadena es llamada un ***calificador de conexión***.

Por ejemplo, asumir que la base de datos remota ***op.negocio.com*** es manejada en un Ambiente Real de Aplicaciones de Cluster de Oracle. La base de datos oficina principal

(***op***) tiene dos nombres de instancia ***op_1*** y ***op_2***. La base de datos local puede contener el siguiente link de base de datos público para definir los caminos a las instancias remotas de la base de datos oficina principal (***op***):

```
CREATE PUBLIC DATABASE LINK op.negocio.com@op_1
USING 'cadena_de_op_1';
CREATE PUBLIC DATABASE LINK op.negocio.com@op_2
USING 'cadena_de_op_2';
CREATE PUBLIC DATABASE LINK op.negocio.com
USING 'cadena_de_op';
```

En los dos primeros ejemplos el nombre de servicio es simplemente una parte del nombre del link de la base de datos. El texto del nombre del servicio no indica necesariamente cómo un link deberá ser establecido; esta información se especifica en el nombre del servicio de la cláusula USING. En el tercer ejemplo, el nombre del servicio no se especifica como parte del nombre del link. En este caso, así como cuando un nombre del servicio es especificado como parte del nombre del link, la instancia es determinada por la cadena USING.

Para usar un nombre del servicio y especificar un caso particular, se incluye el nombre del servicio al final del nombre del objeto global:

```
SELECT * FROM scott.emp@op.negocio.com@op_1
```

En este ejemplo, hay dos símbolos de @.

3.3. Crear links de base de datos compartidos

Cada aplicación que hace referencia a un servidor remoto, usa un link de base de datos estándar para establecer la conexión entre la base de datos local y la base de datos remota. Muchos usuarios que corren aplicaciones simultáneamente pueden causar un número alto de conexiones entre las bases de datos locales y remotas.

Los links de base de datos compartidos permiten limitar el número de conexiones de red requeridas entre el Servidor local y el Servidor remoto.

3.3.1. Determinar cuando se usa los links de base de datos compartidos

Se debe tener cuidado en la aplicación y la configuración del servidor compartido para determinar si se utiliza links compartidos. Una guía sencilla es utilizar los links de base de datos compartidos, cuando el número de usuarios que acceden al link de base de datos es mucho más grande que el número de procesos del servidor en la base de datos local.

La tabla 18. muestra tres configuraciones posibles de links de base de datos.

Tipo de link	Modo de servidor	Consecuencias
No compartido	Dedicado Compartido	Si la aplicación utiliza un link estándar de base de datos público y 100 usuarios requieren simultáneamente una conexión, entonces se requerirán 100 conexiones directas a la red de la base de datos remota.
Compartido	Compartido	Si 10 procesos del servidor compartido existen en el servidor compartido local, entonces, 100 usuarios que usan el mismo link de base de datos requieren 10 o menos conexiones de la red al servidor remoto. Cada proceso del servidor local compartido sólo necesitará una conexión al servidor remoto.
Compartido	Dedicado	Si 10 clientes se conectan al servidor dedicado local y cada cliente tiene 10 sesiones en la misma conexión y cada sesión hace referencia a la misma base de datos, entonces, solo 10 conexiones son necesarias, mientras que con una conexión de base de datos no compartida se necesitarían 100 conexiones.

Tabla 18. Tres configuraciones posibles de links de base de datos

Los link de base de datos compartidos no son útiles en todas las situaciones. Asumir que sólo un usuario accede al servidor remoto. Si este usuario define un link de base de datos compartido y existen 10 procesos compartidos del servidor en la base de datos local, entonces este usuario puede requerir de hasta 10 conexiones de red al servidor, porque el usuario puede utilizar cada proceso compartido del servidor o cada proceso para establecer una conexión al servidor remoto.

Un link de base de datos no compartido es preferible, en esta situación, porque requiere de una sola conexión a la red. Los links de base de datos compartidos solo se usan cuando muchos usuarios necesitan utilizar la misma conexión. Típicamente, los links

compartidos se utilizan para conexiones públicas de base de datos, pero pueden ser utilizados también para conexiones privadas cuando muchos clientes acceden al mismo esquema local (y por lo tanto a la misma conexión de base de datos privada).

3.3.2. Crear links de base de datos compartidos

Formato:

```
CREATE SHARED DATABASE LINK nombre_linkbd
[CONNECT TO nombre_usuario IDENTIFIED BY
contraseña]|[CONNECT TO CURRENT_USER] AUTHENTICATED
BY nombre_esquema IDENTIFIED BY contraseña
[USING 'nombre_servicio'];
```

El siguiente ejemplo crea un usuario fijo, comparte el link para la base de datos ***ventas***, conectándose como ***scott*** y autentificándose como ***katty***:

```
CREATE SHARED DATABASE LINK link2ventas
CONNECT TO scott IDENTIFIED BY tiger
AUTHENTICATED BY katty IDENTIFIED BY richards
USING 'ventas';
```

Siempre que se utiliza la palabra clave SHARED, la cláusula AUTHENTICATED BY será requerida, ya que de esta manera se previene el ingreso de clientes no autorizados, que se disfrazan como usuarios del link de base de datos para acceder a información privilegiada.

3.4. Manejo de links de base de datos

Para el manejo se requiere de:

3.4.1. Cerrar links de base de datos

Si se accede a un link de base de datos en una sesión (*proceso activo*), entonces el link permanece abierto hasta que se cierre la sesión. Esta situación acarrea las siguientes consecuencias:

- Si 20 usuarios abren las sesiones y acceden al mismo link público en una base de datos local, entonces 20 conexiones al link de base de datos estarán abiertas.
- Si 20 usuarios abren las sesiones y cada usuario accede a un link privado, entonces 20 conexiones del link de base de datos estarán abiertas.
- Si un usuario empieza una sesión y accede a 20 conexiones diferentes, entonces 20 conexiones del link de base de datos estarán abiertas.

Después de que se cierra una sesión, los links que estaban activos en la sesión son cerrados automáticamente, sin embargo, puede existir ocasiones en que se necesite cerrar un link manualmente como por ejemplo:

- La conexión de la red establecida por un link es usada irregularmente en una aplicación.
- La sesión del usuario debe ser terminada.

El formato para el cierre es:

```
ALTER SESSION CLOSE DATABASE LINK nombrelink;
```

Esta declaración sólo cierra los links que son activos en la sesión actual.

3.4.2. Borrar links de base de datos

Se pueden borrar los links de base de datos tal como se pueden borrar tablas o vistas. Si el link es privado se debe hacer en el esquema que se encuentra. Si el link es publico se debe tener el privilegio ***DROP PUBLIC DATABASE LINK*** para borrarlo.

Formato:

```
DROP [PUBLIC] DATABASE LINK linkbd;
```

Procedimiento para borrar un link de base de datos privado

1. Conectarse a la base de datos local usando SQL*PLUS. Ejemplo:

CONNECT scott/tiger@local_bd

2. Consultar USER_DB_LINKS para ver el link que se posee. Ejemplo:

   SELECT BD_LINK FROM USER_DB_LINKS;

   BD_LINK

   VENTAS.USA.ORACLE.COM

   MARKETING.USA.ORACLE.COM

   2 rows selected.

3. Borrar el link deseado utilizando la declaración DROP DATABASE LINK. Ejemplo:

   DROP DATABASE LINK ventas.usa.oracle.com;

Procedimiento para borrar un link de base de datos público

1. Conectarse a la base de datos local como un usuario con el privilegio de DROP PUBLIC DATABASE LINK. Ejemplo:

   CONNECT SYSTEM/contraseña@local_bd AS SYSDBA

2. Consultar DBA_DB_LINKS para ver los links públicos. Ejemplo:

   SELECT DB_LINK FROM USER_DB_LINKS

   WHERE OWNER = 'PUBLIC';

DB_LINK

DBL1.USA.ORACLE.COM

VENTAS.USA.ORACLE.COM

INST2.USA.ORACLE.COM

RMAN2.USA.ORACLE.COM

4 rows selected.

3. Borrar el link deseado. Ejemplo:

   DROP PUBLIC DATABASE LINK ventas.usa.oracle.com;

3.4.3. Limitar el número de conexiones activas del link de base de datos

Se puede limitar el número de conexiones desde un proceso de usuario a bases de datos remotas utilizando el parámetro de inicialización estático OPEN_LINKS. Este parámetro controla el número de conexiones remotas que una sola sesión de usuario puede utilizar concurrentemente en transacciones distribuidas.

Consideraciones para configurar estos parámetros:

- El valor debe ser mayor o igual al número de bases de datos.

- Incrementar el valor si varias bases de datos distribuidas han sido accedidas al mismo tiempo. Por ejemplo: Si se accede regularmente a tres bases de datos, se debe poner en OPEN_LINKS para 3 o más.
- El valor por defecto para OPEN_LINKS es 4. Si OPEN_LINKS es puesto a 0, entonces las transacciones distribuidas no son permitidas.

3.5. Vistas para links de base de datos

El diccionario de datos, de cada base de datos, almacena las definiciones de todos los links de base de datos. Se puede utilizar tablas del diccionario de datos y vistas para aprovechar la información de los links.

3.5.1. Determinar que links están en la base de datos

La tabla 19. muestra las vistas de los links de base de datos que se han definido en la base de datos local y están almacenados en el diccionario de datos:

Vista	Propósito
DBA_DB_LINKS	Listas todos los links de la base de datos.
ALL_DB_LINKS	Lista todos los links de la base de datos accesibles al usuario conectado.
USER_DB_LINKS	Lista todos los links de la base de datos que tiene el usuario conectado.

Tabla 19. Links de base de datos definidos en la base de datos local y almacenados en el diccionario de datos

Las siguientes vistas del diccionario de datos contienen información básica de los links de base de datos, con algunas excepciones (Ver tabla 20).

Columna	¿Qué Vista?	Descripción
OWNER	ALL except USER_*	Es el usuario que creó el link de la base de datos. Si el link es público entonces el usuario es listado como público.
DB_LINK	ALL	Es el nombre del link de la base de datos.
USERNAME	ALL	Si la definición del link incluye a un usuario fijo, entonces esta columna visualiza el nombre de usuario, del usuario fijo, caso contrario, la columna es NULA.
PASSWORD	Only USER_*	La contraseña para apuntar a la base de datos remota.
HOST	ALL	El nombre del servicio usado para conectarse a la base de datos remota.
CREATED	ALL	El tiempo de creación de la conexión de la base de datos.

Tabla 20. Excepciones de la información básica de links de base de datos

Cualquier usuario puede consultar USER_DB_LINK para determinar cuáles links de base de datos están disponibles para el usuario. El siguiente script consulta la vista DBA_DB_LINKS para acceder a la información del link:

```
COL OWNER FORMAT a10
COL USERNAME FORMAT A8 HEADING "USER"
COL DB_LINK FORMAT A30
COL HOST FORMAT A7 HEADING "SERVICE"
SELECT * FROM DBA_DB_LINKS
/

SQL>@link_script

OWNER   DB_LINK                         USER    SERVICE    CREATED
----------  ----------------------------    --------   ------------   -------------
SYS         TARGET.USA.NEGOCIO.COM      SYS       inst1    23-JUN-99
PUBLIC    DBL1.UCRANIA.NEGOCIO.COM BLAKE    ora51   23-JUN-99
PUBLIC    RMAN2.USA.NEGOCIO.COM                 inst2    23-JUN-99
PUBLIC    DEPT.USA.NEGOCIO.COM                  inst2    23-JUN-99
MARIA     DBL.UCRANIA.NEGOCIO.COM BLAKE       ora51   23-JUN-99
SCOTT     EMP.USA.NEGOCIO.COM       SCOTT       inst2    23-JUN-99
6 rows selected.
```

Ver información de contraseñas

Sólo USER_DB_LINKS contiene una columna para la información de contraseñas. Sin embargo, un usuario administrativo (SYS o usuarios que se conecten como SYSDBA), puede ver las contraseñas de todos los links de la base de datos consultando la tabla LINK$, caso contrario se debe:

- Conceder el privilegio específico al objeto para la tabla LINK$.
- Conceder el privilegio del sistema SELECT ANY DICTIONARY

Para obtener información de las contraseñas se debe crear y ejecutar el siguiente script SQL* PLUS:

```
COL USERID FORMAT A10
COL CONTRASEÑA FORMAT A10
SELECT USERID, CONTRASEÑA
FROM SYS.LINK$
WHERE CONTRASEÑA IS NOT NULL
/

SQL>@linkpwd

USERID     CONTRASEÑA
----------     ----------------
SYS          ORACLE
BLAKE      TYGER
SCOTT      TIGER
3 rows selected.
```

Autentificación de contraseñas

Es posible ver la autentificación e identificación (AUTHENTICATED BY... IDENTIFIED BY...) de los nombres de usuarios y contraseñas de todos los links de base de datos consultando la tabla LINK$.

Formato:

```
AUTHENTICATED BY … IDENTIFIED BY …
```

Se debe crear y ejecutar el siguiente script SQL*PLUS:

```
COL AUTEUSR FORMAT A10
COL AUTEPWD FORMAT A10
SELECT AUTEUSR AS userid, AUTEPWD AS contraseña
FROM SYS.LINK$
WHERE CONTRASEÑA IS NOT NULL
/

SQL> @autepwd


USERID     CONTRASEÑA
----------     ----------------
ELLIE                MAY
1 row selected.
```

Se puede ver la información del link y la contraseña juntos, realizando un join entre las tablas involucradas para esto se debe crear el siguiente script.

```
COL OWNER FORMAT A8
COL DB_LINK FORMAT A15
COL NOMBREUSUARIO FORMAT A8 HEADING "CON_USER"
COL CONTRASEÑA FORMAT A8 HEADING "CON_PWD"
COL AUTEUSR FORMAT A8 HEADING "AUTE_USER"
COL AUTEPWD FORMAT A8 HEADING "AUTE_PWD"
COL HOST FORMAT A7 HEADING "SERVICE"
COL CREATED FORMAT A10

SELECT DISTINCT
d.OWNER,d.DB_LINK,d.NOMBREUSUARIO,l.CONTRASEÑA,
l.AUTEUSR,l.AUTEPWD,d.HOST,d.CREATED
FROM DBA_DB_LINKS d, SYS.LINK$ l
WHERE CONTRASEÑA IS NOT NULL
AND d.NOMBREUSUARIO = l.USERID
/
```

SQL> @user_and_pwd

```
OWNER    DB_LINK              CON_USER CON_PWD AUTE_USE AUTE_PWD SERVICE CREATED
-------- -------------------- -------- -------- -------- -------- ------- ----------
MARIA    DBL.NEGOCIO.COM      BLAKE    TYGER    ELLIE    MAY      ora51   23-JUN-99
PUBLIC   DBL1.NEGOCIO.COM     SCOTT    TIGER                      ora51   23-JUN-99
SYS      TARGET.NEGOCIO.COM   SYS      ORACLE                     inst1   23-JUN-99
```

3.5.2. Determinar que conexiones de links están abiertas

Es útil para saber que conexiones del link de base de datos están actualmente abiertas en una sesión. Si se conecta como SYSDBA, no se puede hacer una consulta a una vista para determinar todas los links abiertos de todas las sesiones; solo se puede acceder a la información del link en la sesión en la cual se este trabajando.

La tabla 21. muestra las vistas de las conexiones de los links de base de datos que están actualmente abiertas en la sesión actual.

Vista	Propósito
V$DBLINK	Lista todos los links de base de datos abiertos en la sesión, esto es, todos los links de base de datos que tengan YES en la columna IN_TRANSACTION.
GV$DBLINK	Lista todos los links de base de datos abiertos en la sesión junto con sus correspondientes instancias. Esta vista es útil en una Configuración Real de Aplicaciones de Clusters en Oracle.

Tabla 21. Conexiones de links de base de datos abiertas en la sesión actual

Estas vistas del diccionario de datos contienen la misma información básica acerca de los links de base de datos, con algunas excepciones que se indican en la tabla 22.

Columna	¿Qué vista?	Descripción
DB_LINK	Todas	El nombre del link de la base de datos.
OWNER_ID	Todas	El dueño del link de la base de datos.
LOGGED_ON	Todas	El link de la base de datos esta apuntando actualmente.
HETEROGENEOUS	Todas	Si el link de la base de datos es homogénea (NO) o heterogénea (YES).
PROTOCOL	Todas	El protocolo de comunicaciones del link de la base de datos.
OPEN_CURSORS	Todas	Si los cursores del link de base de datos están abiertos.
IN_TRANSACTION	Todas	Si el link de la base de datos esta accediendo a una transacción que no ha sido hecha un commit o rollback.
UPDATE_SENT	Todas	Si había una actualización en el link de la base de datos.
COMMIT_POINT_STRENGTH	Todas	Es el punto donde se hace el commit de las transacciones usando el link de la base de datos.
INST_ID	Solo GV$DBLINK	La instancia desde la cual la información de la vista se obtuvo.

Tabla 22. Excepciones de las vistas en los links de bases de datos

Por ejemplo, se puede crear y ejecutar el script siguiente para determinar los links que están abiertos:

```
COL DB_LINK FORMAT A25
COL OWNER_ID FORMAT 99999 HEADING "OWNID"
COL LOGGED_ON FORMAT A5 HEADING "LOGON"
COL HETEROGENEOUS FORMAT A5 HEADING "HETER"
COL PROTOCOL FORMAT A8
COL OPEN_CURSORS FORMAT 999 HEADING "OPN_CUR"
COL IN_TRANSACTION FORMAT A3 HEADING "TXN"
COL UPDATE_SENT FORMAT A6 HEADING "UPDATE"
COL COMMIT_POINT_STRENGTH FORMAT 99999 HEADING
"C_P_S"

SELECT * FROM V$DBLINK
/

SQL> @dblink

DB_LINK                   OWNID LOGON HETER PROTOCOL OPN_CUR TXN UPDATE  C_P_S
------------------------- ------ ----- ----- -------- ------- --- ------ ------
INST2.NEGOCIO.COM             0 YES   YES    UNKN          0 YES YES       255
```

3.6. Crear la ubicación de transparencia

Después de configurar los links necesarios de la base de datos, se puede utilizar varias herramientas para ocultar la naturaleza distribuida del sistema de base de datos de los usuarios. En otras palabras, los usuarios pueden tener acceso a objetos remotos como que si estos fueran objetos locales.

3.6.1. Usar vistas para crear la ubicación de transparencia

Las vistas locales pueden proporcionar la ubicación de transparencia para tablas locales y remotas en un sistema distribuido de base de datos.

Por ejemplo, asumir que la tabla empleados (***emp***) está almacenada en la base de datos local y la tabla departamentos (***dept*)** en la base de datos remota. Para hacer estas tablas transparentes a los usuarios del sistema, se debe crear una vista en la base de datos local de tal manera que junte los datos locales y remotos. Así :

```
CREATE VIEW compañía
AS
SELECT a.empnu, a.nombreempleado, b.nombredept
FROM scott.emp a, jose.dept@op.negocio.com b
WHERE a.numdept = b.numdept;
```

La Figura 9. muestra la vista y la ubicación de transparencia.

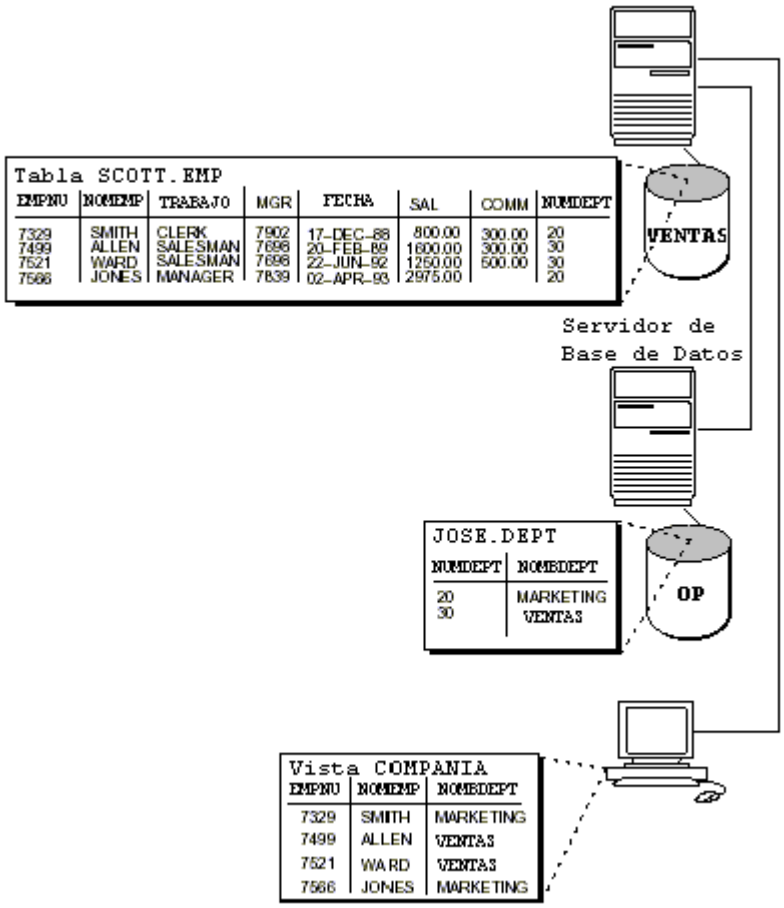


Figura 9. Vistas y ubicación de transparencia

Cuando los usuarios acceden a esta vista, no necesitan saber dónde están almacenados los datos físicamente o si están accediendo a datos de más de una tabla. Esto hace que ellos obtengan la información requerida más fácil. Por ejemplo:

```
SELECT * FROM compañía;
```

El dueño de la vista local puede conceder privilegios sólo a esos objetos de la vista local que han sido concedidos por el usuario remoto.

3.6.2. Usar sinónimos para crear la ubicación de transparencia

Los sinónimos son útiles en ambientes distribuidos y no distribuidos porque ellos ocultan la identidad del objeto, incluyendo su ubicación en un sistema de base de datos distribuido. Si se desea renombrar o mover un objeto sólo se necesita redefinir el sinónimo; las aplicaciones basadas en ese sinónimo continuarán funcionando normalmente.

Crear sinónimos

Se pueden crear sinónimos para: Tablas, Tipos, Vistas, Vistas materializadas, Secuencias, Procedimientos, Funciones, y Paquetes.

Todos los sinónimos son objetos del esquema y se almacenan en el diccionario de datos de la base de datos en la que fueron creados.

Formato:

```
CREATE [PUBLIC] SYNONYM nombre_sinonimo
FOR [esquema.]nombre_objeto[@nombre_link_basedatos]
```

dónde:

PUBLIC: Es una palabra clave y específica que esté sinónimo está disponible para todos los usuarios y si se lo omite, será privado y lo utilizará solo el creador. Los sinónimos públicos pueden ser creados sólo por usuarios con privilegios de CREATE PUBLIC SYNONYM.

nombre_sinonimo: Especifica el nombre del objeto alterno que va a ser referenciado por usuarios y aplicaciones.

esquema: Específica el esquema del objeto especificado en nombre_objecto. Al omitirse este parámetro usará el esquema del creador como el esquema del objeto.

nombre_objeto: Específica una tabla, vista, secuencia, vista materializada, tipo, procedimiento, función o paquete.

nombre_link_basedatos: Específica el link de base de datos identificado en la base de datos remota y el esquema en la cual el objeto especificado en nombre_objeto es localizado.

Ejemplo: Crear un sinónimo público

Asumir que en cada base de datos en un sistema de base de datos distribuido, un sinónimo público se define para la tabla ***scott.emp*** almacenada en la base de datos de oficina principal (***op***):

```
CREATE PUBLIC SYNONYM emp FOR scott.emp@op.negocio.com;
```

Si se puede diseñar una aplicación que administre a los empleados sin considerar donde se utilice la aplicación, porque la ubicación de la tabla scott.emp@op.negocio.com es ocultada por los sinónimos públicos. Las declaraciones de SQL en la aplicación accederán a la tabla referenciada en el sinónimo público ***emp*.**

Además, si se mueve la tabla empleados (***emp***) de la base de datos oficina principal (***op***) a la base de datos de Control (***Cr***), entonces sólo se necesita cambiar los sinónimos públicos en los nodos del sistema. La aplicación que administra a los empleados continuará funcionando apropiadamente en todos los nodos.

Manejar privilegios y sinónimos

Un sinónimo es una referencia al objeto actual, un usuario que tiene acceso a un sinónimo, para un objeto particular del esquema, debe tener privilegios para el esquema de objetos de sí mismo. Por ejemplo, si el usuario quiere acceder a un sinónimo pero no tiene los privilegios sobre una tabla, ocurrirá un error indicando que la tabla o vista no existe.

Asumir que ***scott*** crea el sinónimo local ***emp*** como un alias para el objeto remoto ***scott.emp@ventas.negocio.com***. ***Scott*** no puede conceder privilegios de objetos en el

sinónimo para otro usuario local ya que aumentaría un gran número de privilegios en la tabla remota empleados (***emp***) en la base de datos ***ventas*** lo cual no es permitido. Esta conducta es diferente cuando se administra privilegios para sinónimos que tienen alias para tablas locales o vistas.

Por lo tanto, no se puede manejar privilegios locales cuando los sinónimos utilizan la ubicación de transparencia. La seguridad para el objeto se controla enteramente en el nodo remoto. Por ejemplo, el usuario administrador (***admin***) no puede conceder privilegios de objetos para el sinónimo empleados (***EMP_SYN***).

3.6.3. Usar procedimientos para crear la ubicación de transparencia

Las unidades de programa PL/SQL llamadas procedimientos pueden proporcionar ubicación de transparencia.

Procedimientos locales para referirse a datos remotos

Los procedimientos o funciones (individuales o en paquetes) pueden tener declaraciones SQL que hacen referencia a datos remotos. Por ejemplo, el procedimiento creado por la declaración siguiente:

```
CREATE PROCEDURE emp_aux (numemp NUMBER) AS
BEGIN
DELETE FROM emp@op.negocio.com
WHERE empnu = numemp;
END;
```

Cuándo un usuario o aplicación llame al procedimiento emp_aux, no esta claro que una tabla remota este modificándose.

Una segunda capa de ubicación de transparencia es posible cuando las declaraciones en un procedimiento indirectamente hacen referencia a un dato remoto usando procedimientos locales, vistas o sinónimos. Por ejemplo, la declaración siguiente define un sinónimo local:

```
CREATE SYNONYM emp FOR emp@op.negocio.com;
```

Con este sinónimo, se puede crear el procedimiento emp_aux que utiliza la declaración:

```
CREATE PROCEDURE emp_aux (numemp NUMBER) AS
BEGIN
DELETE FROM emp WHERE empnu = numemp;
END;
```

Si se renombra o mueve la tabla empleados de oficina principal (***emp@op***), entonces solo se necesita modificar el sinónimo local que hace referencia a la tabla.

Procedimientos locales para llamar a procedimientos remotos

Un procedimiento local puede llamar a un procedimiento remoto. El procedimiento remoto debe ejecutar el DML (*Data Manipulation Language, manipula los datos que contienen los objetos de la base de datos utilizando instrucciones insert, select, update y delete*) requerido. Por ejemplo, asumir que ***scott*** se conecta a ***local_bd*** y crea el procedimiento siguiente:

```
CONNECT scott/tiger@local_bd
CREATE PROCEDURE emp_aux (numemp NUMBER)
AS
BEGIN
EXECUTE emp_aux1@op.negocio.com;
END;
```

Ahora, asumir que ***scott*** se conecta a la base de datos remota y crea el procedimiento remoto:

```
CONNECT scott/tiger@op.negocio.com
CREATE PROCEDURE emp_aux1 (numemp NUMBER)
AS
BEGIN
DELETE FROM emp WHERE empnu = numemp;
END;
```

Cuando un usuario o aplicación que este conectado a ***local_bd***, llame al procedimiento ***emp_aux***, este procedimiento en cambio llamará al procedimiento remoto ***emp_aux1*** en ***op.negocio.com***.

Sinónimos locales para referirse a procedimientos remotos

Por ejemplo, ***scott*** se conecta a la base de datos local ***ventas.negocio.com***. y crea el procedimiento siguiente:

```
CREATE PROCEDURE emp_aux (numemp NUMBER) AS
BEGIN
DELETE FROM emp@op.negocio.com
WHERE empnu = numemp;
END;
```

El usuario ***pedro*** entonces se conecta a la base de datos ***sumi.negocio.com*** y crea el siguiente sinónimo para el procedimiento que ***scott*** creó en la base de datos ***ventas***.

```
SQL> CONNECT pedro/paz@sumi
SQL> CREATE PUBLIC SYNONYM emp
       FOR scott.emp_aux@ventas.negocio.com;
```

Un usuario local en suministros (***sumi***) puede usar este sinónimo para ejecutar el procedimiento en ***ventas***.

3.7. Manejar declaraciones de transparencia

Oracle permite las siguiente declaraciones DML para referirse a tablas remotas: SELECT (*consultas*), INSERT, UPDATE, DELETE, SELECT … FOR UPDATE (*no siempre soporta Sistemas Heterogéneos*), LOCK TABLE.

Las consultas incluyen joins, agregaciones, subconsultas, y SELECT … FOR UPDATE para hacer referencia a cualquier tabla local o remota y vistas. Por ejemplo, la siguiente consulta une información de dos tablas remotas (*joins*):

```
SELECT e.empnu, e.nombreempleado, d.nombredept
FROM scott.emp@ventas.division3.negocio.com e,
jose.dept@op.negocio.com d
WHERE e.numdept = d.numdept;
```

Las declaraciones UPDATE, DELETE, INSERT y LOCK TABLE pueden hacer referencia a ambas tablas local y remota. No es necesario programar para actualizar los datos remotos. Por ejemplo, la declaración siguiente inserta nuevas filas dentro de la tabla remota empleados (***emp***) en el esquema de ***scott.ventas*** seleccionando las filas de la tabla empleados (***emp***) en el esquema de ***jose*** en la base de datos local:

```
INSERT INTO scott.emp@ventas.division3.negocio.com
SELECT * FROM jose.emp;
```

3.8. Desarrollo de aplicaciones para sistemas de bases de datos

3.8.1. Manejar la distribución de los datos de una aplicación

En un ambiente distribuido de base de datos se debe coordinar con el administrador de bases de datos para determinar la mejor ubicación de los datos. Los problemas a considerar son:

- El número de transacciones de cada ubicación.
- La cantidad de datos (partes de tablas) usadas por cada nodo.
- Características y fiabilidad de la red.
- La velocidad de varios nodos y la capacidad de los discos.
- La importancia de un nodo o un link cuando estos no están disponibles.
- Necesidad de integridad referencial entre tablas.

3.8.2. Controlar las conexiones establecidas por los links de base de datos

Cuando un nombre del objeto global esta referenciado en una declaración SQL o en una llamada a un procedimiento remoto, los links de base de datos establecen una conexión a una sesión en la base de datos remota para el usuario local. La conexión y sesión remota solo se crean si la conexión no se ha establecido previamente para el usuario local, las mismas que persisten durante la sesión del usuario local, a menos que la aplicación o el usuario explícitamente las termine. Cuando se hace una declaración SELECT a través de un link de base de datos, una transacción bloqueada se coloca en los segmentos rollback. Para re-liberar el segmento, se debe hacer una declaración COMMIT o ROLLBACK.

Para terminar una conexión y sesión remota se usa la sentencia ALTER SESSION con la cláusula CLOSE DATABASE LINK. Por ejemplo:

```
SELECT * FROM emp@ventas;
COMMIT;
```

La declaración siguiente termina la sesión en la base de datos remota que apunta al link de la base de datos ***ventas***:

```
ALTER SESSION CLOSE DATABASE LINK ventas;
```

Para cerrar la conexión de un link de base de datos en una sesión de usuario se debe tener el privilegio del sistema ALTER SESSION.

3.8.3. Mantener la integridad referencial en un sistema distribuido

Si una parte de una declaración distribuida falla, por ejemplo, debido a una violación de integridad, Oracle retornará el número de error ORA-02055 (*Error en operación distribuida de actualización, posibles sitios inconsistentes con otros: requiere un rollback*) mientras que, las declaraciones subsiguientes o llamadas a procedimiento retornarán el número de error ORA-02067 (*Error, no se garantiza la ejecución de operaciones anteriores, por causa de un trigger o un procedimiento almacenado con múltiples actualizaciones remotas, requiere una transacción o un rollback*.) hasta que un rollback se emita.

Oracle no permite declaraciones de integridad referencial para que sean definidas a través de nodos de un sistema distribuido. En otras palabras, una declaración de integridad referencial en una tabla no puede especificar una clave foránea que haga referencia a una clave primaria de una tabla remota. No obstante, se puede mantener las relaciones padre/hijo (parent/child) a través de los nodos usando triggers, pero esto puede causar que la red fracase limitando así la accesibilidad no solo a la tabla padre sino también a la tabla hijo.

2.8.4. Establecer consultas distribuidas

El servidor local de la base de datos Oracle, particiona la consulta distribuida en un número de consultas remotas, las cuales se envían a los nodos remotos para su ejecución. Los nodos remotos ejecutan las consultas y devuelven los resultados al nodo local. El nodo local realiza algún proceso necesario y retorna los resultados al usuario o a la aplicación.

Usar vistas en línea

La manera más efectiva de optimizar consultas distribuidas es accediendo a las bases de datos remotas lo mínimo que sea posible y recuperar solo los datos requeridos.

Por ejemplo, asumir que se hace referencia a cinco tablas remotas de dos bases de datos remotas diferentes para una consulta distribuida y se tiene un filtro complejo (por

ejemplo, WHERE r1.salario + r2.salario > 50000). Se puede mejorar la ejecución de esta consulta reescribiendo la misma, para acceder una vez a las bases de datos remotas y aplicar el filtro al sitio remoto. Esta reordenación causa que haya menos datos a ser transferidos al lugar de la ejecución de la consulta.

Reordenar la consulta para acceder a la base de datos remota se logra una vez que se haya utilizado las vistas en línea. A continuación se definen los siguientes términos:

Collocated.- Significa que dos o mas tablas están localizadas en la misma base de datos.

Inline view.- Una declaración SELECT es sustituida por una tabla en una declaración SELECT padre. La declaración SELECT incluida, dentro de los paréntesis es un ejemplo de una vista en línea:

```
SELECT e.empnu,e.nombreempleado,d.numdept,d.nombredept
FROM (SELECT empnu, nombreempleado
        From emp@orc1.mundo) e, dept d;
```

Collocated inline view.- Una vista en línea que selecciona datos de múltiples tablas de una sola base de datos. Reduce la cantidad de tiempo que la base de datos remota necesita para mejorar una consulta distribuida.

CAPITULO IV

TRANSACCIONES DISTRIBUIDAS

4.1. Transacciones distribuidas

Son aquellas que incluyen una o más declaraciones que, en forma individual o en grupo, permiten actualizar datos en dos o más nodos en una base de datos distribuida.

La figura 10. muestra una transacción distribuida.

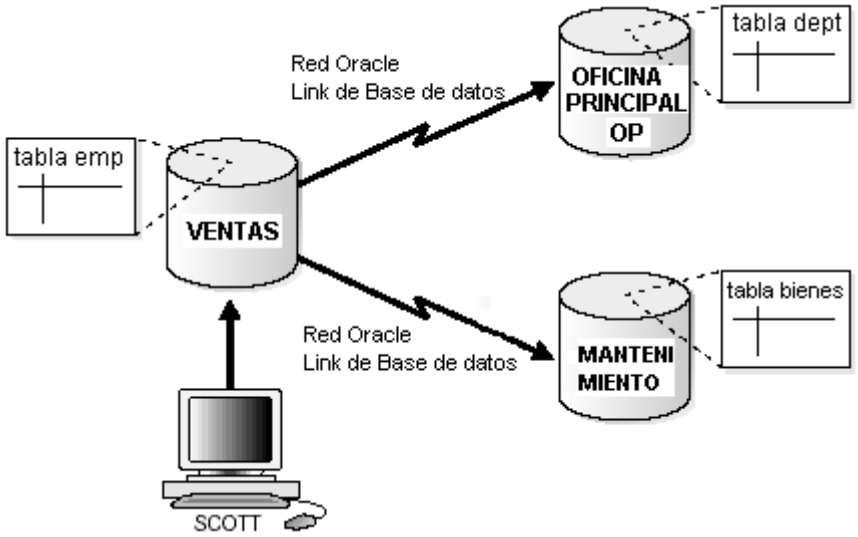


Figura 10. Sistema distribuido – transacciones

La siguiente transacción distribuida ejecutada por ***scott*** actualiza la base de datos local ***ventas*** y las bases de datos remota oficina principal (***op***) y ***mantenimiento***:

```
UPDATE scott.dept@op.usa.negocio.com
SET local = 'ROJO'
WHERE numdept = 10;
UPDATE scott.emp
SET numdept = 11
WHERE numdept = 10;
UPDATE scott.bienes@mantenimiento.usa.negocio.com
SET sala= 1225
WHERE sala = 1163;
COMMIT;
```

Si todas las declaraciones de una transacción hacen referencia solo a un nodo remoto, la transacción será remota y no distribuida.

Hay dos tipos de operaciones en las transacciones distribuidas:

- Transacciones DML y DDL
- Declaraciones de control de transacciones

Transacciones DML y DDL

Entre las operaciones DML y DDL que soporta una transacción distribuida tenemos:

- CREATE TABLE AS SELECT
- DELETE

- INSERT(se cargan por defecto o directamente)
- LOCK TABLE
- SELECT
- SELECT FOR UPDATE

Se puede ejecutar las declaraciones DML y DDL en forma paralela (*Consultas que se hacen a diferentes nodos para mejorar el tiempo de ejecución*), e INSERT para cargar directamente las declaraciones en serie (*Cuando dos o más transacciones se ejecutan simultáneamente el resultado debe ser el mismo. Se ejecutan una a continuación de la otra*), pero tiene las siguientes restricciones:

- Todas las operaciones remotas deben ser declaraciones SELECT.
- Estas declaraciones no deben ser cláusulas en otra transacción distribuida.
- Si la tabla referenciada en tabla_expresión_clausula de una declaración INSERT, UPDATE o DELETE es remota, entonces, la ejecución será en serie y no en paralelo.
- No se pueden hacer operaciones remotas luego de realizar DML/DDL paralelas o cargar directamente INSERT.
- Si la transacción empieza utilizando XA (*Librería Oracle, es una interface externa que permite coordinar las transacciones globales con transacciones administradas*) o OCI (*Oracle Call Interface, proporciona acceso comprensivo*

a toda la funcionalidad de la base de datos Oracle brindado escalabilidad y seguridad), estas se ejecutaran en serie.

- Ninguna operación de ciclos se puede realizar en las transacciones que originan un funcionamiento paralelo. Por ejemplo, no se puede hacer referencia a un objeto remoto que es actualmente un sinónimo de un objeto local.

Declaraciones de control de transacción

Entre las declaraciones para el control de transacciones tenemos:

- COMMIT
- ROLLBACK
- SAVEPOINT

4.2. Arboles de sesión para transacciones distribuidas

Al hacer una declaración en una transacción distribuida, Oracle define un árbol de sesión para todos los nodos que participan en la transacción.

Un árbol de sesión es un modelo jerárquico que describe las relaciones entre sesiones y sus roles; como se puede observar en la figura 11.

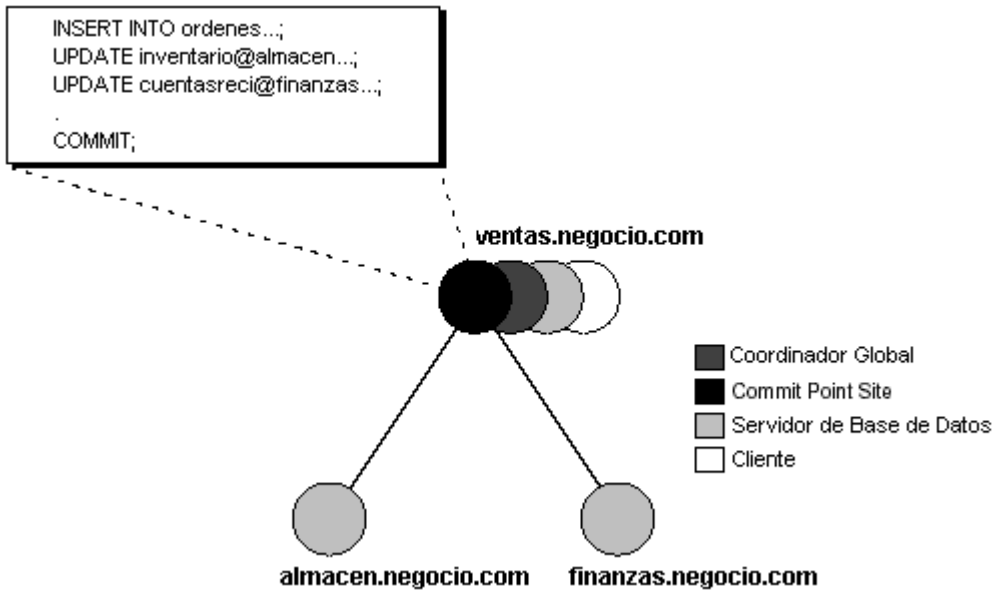


Figura 11. Ejemplo de un árbol de sesión

Todos los nodos que participan en un árbol de una transacción distribuida tienen uno o más de los siguientes roles que se muestran en la tabla 23.

Rol	Descripción
Cliente	Un nodo que hace referencia a información de una base de datos que pertenece a un nodo diferente.
Servidor de Base de Datos	Un nodo recibe una petición de información de otro nodo.
Coordinador Global	El nodo que origina la transacción distribuida.
Coordinador Local	Un nodo hace referencia a datos de otros nodos para terminar su transacción.
Commit point Site	El nodo que actualiza o deshace la transacción según la instrucción del coordinador global.

Tabla 23. Roles de los nodos de un árbol en una transacción distribuida

El rol que un nodo juega en una transacción distribuida se determina por:

- Si la transacción es local o remota.
- El commit point strength (*Prioridad de terminación de la transacción*) del nodo.
- Si todos los datos que se requieren están en un nodo o hay que hacer referencia a otros para completar la transacción.
- Si el nodo es de sólo lectura.

4.2.1. Cliente

Un nodo actúa como un cliente cuando este hace referencia a información de otros nodos de base de datos. El nodo referenciado es un servidor de base de datos. En la figura 11, el nodo ***ventas*** es un cliente de los nodos que se hospedan en las bases de datos ***almacén*** y ***finanzas***.

4.2.2. Servidor de base de datos

Un servidor de base de datos es un nodo que hospeda una base de datos desde la cual un cliente solicita información.

En el Figura 11, se mostró una aplicación del nodo ***ventas*** inicia una transacción distribuida que accede a los datos de los nodos ***almacén*** y ***finanzas***. Por lo tanto, ***ventas.negocio.com*** tiene el rol del nodo cliente, mientras que ***almacén*** y ***finanzas*** son

servidores de base de datos. En este ejemplo, ***ventas*** es un servidor de la base de datos y un cliente porque la aplicación también modifica los datos en la base de datos ***ventas***.

4.2.3. Coordinador local

Un nodo que hace referencia a datos de otros nodos para completar una transacción distribuida se llama coordinador local. En el Figura 11 se mostró, ***ventas*** es un coordinador local porque coordina a los nodos referenciados directamente: ***almacén*** y ***finanzas***, sin embargo, ***ventas*** también es coordinador global porque coordina a todos los nodos implicados en la transacción.

Un coordinador local es responsable de coordinar la transacción entre los nodos que se comunican directamente. Sus tareas son:

- Recibir y retransmitir información del estado de la transacción y de los nodos.
- Enviar las consultas de los nodos.
- Recibir las consultas de los nodos y enviarlos hacia otros nodos.
- Retornar los resultados de las consultas a los nodos que los iniciaron.

4.2.4. Coordinador global

El nodo donde la transacción distribuida se origina se llama coordinador global. La aplicación de la base de datos que publica la transacción distribuida se conecta directamente al nodo que actúa como coordinador global. Por ejemplo, en la figura 11 se mostró, la transacción hecha al nodo ***ventas*** solicita información de los servidores ***almacén*** y ***finanzas***. Por lo tanto, ***ventas.negocio.com*** es el coordinador global de esta transacción distribuida.

El coordinador global viene a ser el padre o la raíz del árbol de la sesión. El coordinador global realiza las operaciones siguientes durante una transacción distribuida:

- Envía todas las declaraciones SQL de las transacciones distribuidas, llamadas a procedimientos remotos, etc, haciendo referencia directamente a los nodos, formando de esta manera el árbol de sesión.
- Instruye directamente a todos los nodos referenciados, mientras que el commit point site (*Nodo de terminación de la transacción, con la prioridad más alta*) prepara la transacción.
- Instruye al Commit Point Site para iniciar un commit global de la transacción si todos los nodos están preparados.
- Instruye a todos los nodos para iniciar un rollback global de la transacción si existe una respuesta de aborto.

4.2.5. Commit point site

El trabajo del commit point site es iniciar una operación de commit o rollback según la orden del coordinador global. El administrador del sistema designa siempre un nodo para que sea el commit point site del árbol de sesión, asignando a todos los nodos un commit point strength. El nodo seleccionado como commit point site debe ser el nodo que almacena los datos más críticos (*Bases de datos que contienen varios cientos de gigabytes y, en muchos casos, varios terabytes de datos. Requieren de mantenimiento de bases de datos grandes conocidas como BDGV. Bases de datos de gran volumen*).

La Figura 12 es un ejemplo de sistema distribuido, con la base de datos ***ventas*** como Commit Point Site.

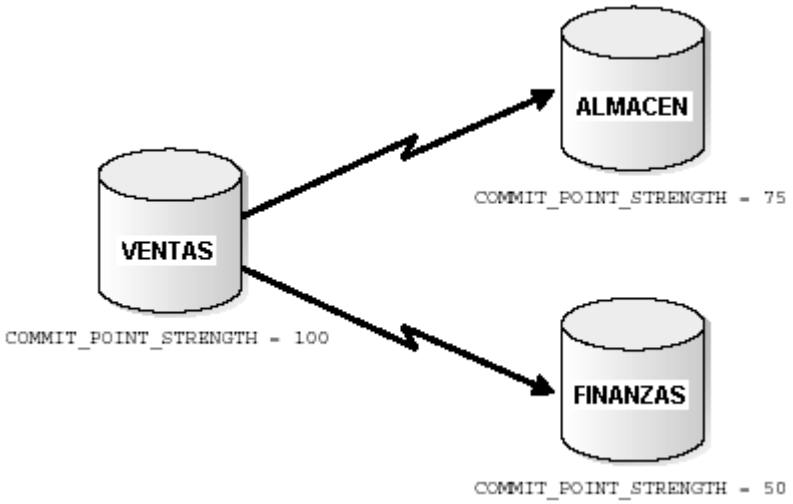


Figura 12. Commit Point Site

El Commit Point Site es distinto de todos los otros nodos involucrados en una transacción distribuida por lo siguiente:

- El Commit Point Site nunca entra a un estado preparado (*Estado donde un nodo puede reconocer o no al coordinador local a través de un mensaje, esto no garantiza que el commit ha sido recibido*). Consecuentemente, si el Commit Point Site almacena los datos más críticos, estos datos nunca se quedan en duda (*Transacciones In-Doubt, fallas por que no se realizan completamente un commit o un rollback*), incluso si ocurre una falla. En situaciones de fallas, los nodos fallidos se quedan en un estado preparado, garantizando de esta manera la seguridad de los datos hasta que las transacciones dudosas se resuelvan.
- El Commit Point Site se actualiza antes que los otros nodos involucrados en la transacción. La salida de una transacción distribuida del Commit Point Site determina si la transacción en todos los nodos hizo un commit o rollback. El coordinador global asegura que todos los nodos completen la transacción de la misma forma que el Commit Point Site.

Actualización de una transacción distribuida

Una transacción distribuida es considerada actualizada después que todos los nodos no Commit Point Site se prepararon y la transacción fue actualizada en el Commit Point Site. Finalmente el archivo Redo Log (*Juego de archivos que graban todos los cambios que se hacen en una base de datos de Oracle. Una base de datos debe tener al menos*

dos archivos redo log) del Commit Point Site se actualiza, así como también el resto de nodos involucrados en la transacción.

Como el commit point log contiene un registro de actualización, la transacción es considerada actualizada, aunque algunos nodos participantes estén solo en estado preparado por lo que, la transacción no se actualizará en estos nodos. De la misma manera, una transacción distribuida se considera no actualizada si la actualización no ha sido ejecutada en el Commit Point Site.

Commit point strength

A cada servidor de base de datos se debe asignar un commit point strength. Si a un servidor de base de datos se hace referencia en una transacción distribuida, el valor del Commit Point Strength determinará que rol juega en el commit de dos fases. Específicamente, el commit point strength determina que nodo es el commit point site en la transacción distribuida. Este valor se especifica usando el parámetro de inicialización COMMIT_POINT_STRENGTH.

El commit point site, el cual se determina al comienzo de la fase de preparación, se lo selecciona de entre los nodos que forman parte de la transacción. Se sigue la siguiente secuencia:

1. De los nodos referenciados directamente por el coordinador global, Oracle selecciona al nodo con mas alto Commit Point Strength como el Commit Point Site.
2. El nodo inicialmente seleccionado determina si cualquier nodo de los que se va a obtener información para una transacción, tiene un más alto commit point strength.
3. O el nodo con el más alto commit point strength referenciado directamente en la transacción o uno de los servidores con el mas alto commit point strength pueden llegar a ser el commit point site.
4. Después de que el commit point strength final ha sido determinado, el coordinador global envía respuestas preparadas a todos los nodos que forman parte de la transacción.

La figura 13. muestra un ejemplo del árbol de sesión del commit point strengths de cada nodo (en paréntesis) y muestra el nodo escogido como el commit point site:

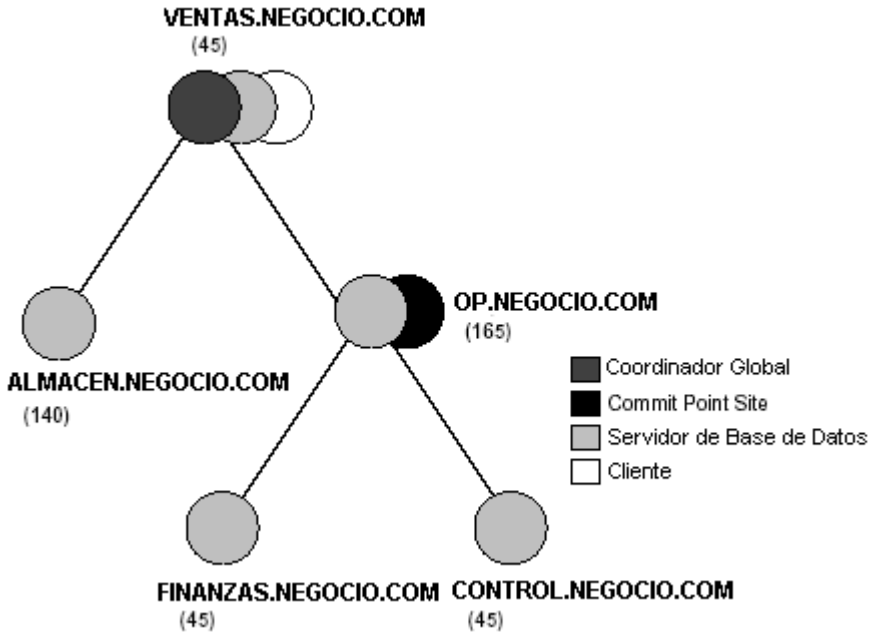


Figura 13. Commit point strengths y determinación del commit point site

Las condiciones siguientes se aplican al determinar el commit point site:

- Un nodo sólo de lectura no puede ser commit point site.
- Si múltiples nodos directamente referenciados por el coordinador global tienen el mismo commit point strength, entonces Oracle designa a uno de estos como el commit point site.
- Si una transacción distribuida termina con un rollback, entonces el estado preparado y el commit de fases no son necesarios, por lo que Oracle no determina un commit point site y el coordinador global envía una declaración rollback a todos los nodos para terminar el proceso de la transacción distribuida.

4.3. Mecanismo commit de dos fases (Two-phase commit)

A diferencia de una transacción de una base de datos local, una transacción distribuida involucra varios datos que se alteran en múltiples bases de datos. Por lo tanto, el procesamiento de una transacción distribuida es más complicado, porque Oracle debe coordinar el commit o rollback de los cambios que hay en una transacción como una unidad autónoma. En otras palabras, la transacción debe realizar un completo commit o un completo rollback.

Oracle asegura la integridad de los datos en una transacción distribuida utilizando el mecanismo commit de dos fases. En la ***fase de preparación***, el nodo que comienza la transacción pregunta a los otros nodos involucrados para realizar el commit o rollback de la transacción. Durante la ***fase de actualización***, el nodo que comienza la transacción pregunta a todos nodos que participan para actualizar la transacción. Si el resultado es posible, todos los nodos involucrados ejecutarán un commit, caso contrario, ejecutarán su respectivo rollback.

Todos los nodos que participan en una transacción distribuida deben realizar la misma acción: o todos hacen un commit o todos hacen un rollback. Oracle automáticamente controla y monitorea el commit o rollback de la transacción y mantiene la integridad de la base de datos global (*todas las bases que participan en la transacción*) utilizando el

mecanismo de commit de dos fases. El mecanismo es completamente transparente y no requiere de ninguna programación o aplicación por parte del usuario.

Este mecanismo tiene fases distintas que Oracle realiza automáticamente siempre que un usuario actualiza una transacción distribuida (Ver tabla 24).

Fase	Descripción
Fase de Preparación (Prepare phase)	El nodo que comienza, llama al coordinador global, pregunta a los otros nodos que el commit point site va hacer un commit o un rollback. Aunque si hay fallas, por que un nodo no puede realizar la transacción esta se detendrá.
Fase de Actualización (Commit phase)	Si todos los nodos participantes responden al coordinador que están preparados, entonces el coordinador pide al commit point site para hacer un commit. Después el coordinador pregunta a todos los otros nodos para actualizar la transacción
Fase de Olvido (Forget phase)	El coordinador global se olvida de la transacción.

Tabla 24. Fases del mecanismo commit

4.4. Transacciones en duda (Transactions in-doubt)

El mecanismo commit de dos fases asegura que todos los nodos ejecuten o un commit o un rollback juntos, sin embargo, puede ocurrir que en una de las tres fases del proceso ocurra un fallo ya sea del sistema o por un error en la red, es entonces cuando la transacción se pone en duda.

Una transacción distribuida puede ser dudosa por los siguientes aspectos:

- Un servidor Oracle sufre una caída de software.
- Una conexión de red entre dos o más bases de datos Oracle implicadas en un proceso distribuido se desconecta.
- Ocurre un error de software inmanejable.

El proceso RECO (*Oracle RECOverer Process proceso de fondo que resuelve transacciones en duda*) automáticamente resuelve las transacciones dudosas cuando la máquina, la red o los problemas de software son resueltos. Hasta que RECO resuelva la transacción, los datos se aseguran ya sea para lectura o escritura. Los bloques de Oracle leen porque no pueden determinar cuál versión de los datos debe mostrarse en la consulta.

4.4.1. Resolución automática de transacciones in-doubt

En la mayoría de casos, Oracle resuelve las transacciones dudosas automáticamente.

4.4.2. Resolución manual de transacciones in-doubt

Se debe solucionar manualmente, si surgen únicamente algunos de los siguientes casos:

- La transacción in-doubt (en duda) tiene seguridades sobre datos críticos o segmentos rollback.

- Por causa de una máquina, de la red o por fallas de software, no puede ser reparado rápidamente el nodo.

La resolución de transacciones dudosas se puede complicar, por lo que se requiere de los siguientes procedimientos:

- Identificar el número de la transacción dudosa.
- Consultar las vistas DBA_2PC_PENDING (*lista de transacciones en duda pendientes*) y DBA_2PC_NEIGHBORS (*lista transacciones entrantes de clientes remotos, y transacciones salientes de servidores remotos, de transacciones en duda*) para determinar si las bases de datos implicadas en la transacción se han actualizado.
- Si es necesario, forzar la actualización utilizando la declaración COMMIT FORCE o un rollback utilizando la declaración ROLLBACK FORCE.

4.4.3. Relevancia del sistema de cambio de números en transacciones in-doubt

Un sistema de cambio de número (SCN) es un tiempo interno que actualiza la versión de la base de datos. El servidor de base de datos Oracle usa el valor del reloj SCN para garantizar la consistencia de la transacción. Por ejemplo, cuando un usuario actualiza una transacción, Oracle registra un SCN para esta operación en el registro online redo log.

Oracle usa los SCNs para coordinar las transacciones distribuidas entre las diferentes bases de datos. Por ejemplo, Oracle utiliza los SCNs de la siguiente manera:

1. Una aplicación establece una conexión utilizando un link de base de datos.
2. La transacción distribuida actualiza con el SCN global más alto, de entre todas las bases de datos involucradas.
3. La actualización SCN global es enviada a todas las bases de datos involucradas en la transacción.

Los SCNs son importantes para las transacciones distribuidas porque estos sincronizan el tiempo de actualización de una transacción, aún si la transacción fallase. Si una transacción es dudosa, el administrador puede usar el SCN para coordinar los cambios que se hagan en la base de datos global. El SCN global de la transacción actualizada también puede ser usado para identificar la transacción más tarde, por ejemplo, en la recuperación distribuida.

4.5. Configuración de los parámetros de inicialización de las transacciones distribuidas

Se puede configurar los parámetros de inicialización que controlan la conducta del procesamiento de transacciones distribuidas. La tabla 25. describe los parámetros de inicialización más relevantes en el procesamiento de transacciones distribuidas:

PARÁMETROS	DESCRIPCION
DISTRIBUTED_TRANSACTIONS	Especifica el número máximo de transacciones distribuidas que la base de datos puede soportar.
COMMIT_POINT_STRENGTH	Especifica el valor usado para determinar el commit point site en una transacción distribuida.

Tabla 25. Parámetros de inicialización en el procesamiento de transacciones distribuidas

4.5.1. Limitar el número de transacciones distribuidas

El parámetro de inicialización DISTRIBUTED_TRANSACTIONS limita el número de transacciones distribuidas, en la que una instancia dada puede participar concurrentemente, como un cliente y como un servidor. El valor por defecto de este parámetro es operado por el sistema dependiente.

Si Oracle alcanza este límite y el subsiguiente usuario emite una declaración SQL que hace referencia a una base de datos remota, entonces el sistema ejecuta un rollback y retorna el siguiente mensaje de error:

ORA-2042: too many global transactions (*Demasiadas transacciones globales, la tabla de la transacción distribuida está llena porque hay demasiadas transacciones distribuidas activas*).

Por ejemplo, asumir que se colocó el siguiente parámetro para una instancia dada:

```
DISTRIBUTED_TRANSACTIONS = 10
```

En este caso, un máximo de 10 sesiones pueden ser procesadas simultáneamente en una transacción distribuida. Si una sesión adicional intenta emitir una declaración DML solicitando acceso distribuido, Oracle retorna un mensaje de error para la sesión y hace un rollback a la declaración.

Incrementar el límite de la transacción

Si el valor de DISTRIBUTED_TRANSACTIONS es superado, cuando una instancia participe en numerosas transacciones distribuidas retornará frecuentemente el mensaje de error ORA-2042, por lo que se debe modificar el límite para permitir que más usuarios procesen sus transacciones distribuidas.

Decrementar el límite de la transacción

Si un sitio experimenta un número anormalmente alto de fallas en la red, se puede temporalmente disminuir el valor de DISTRIBUTED_TRANSACTIONS. Esta operación limita el número de transacciones in-doubt (en duda) del sitio, por lo que el valor del límite, asegura los datos del sitio y el número de transacciones en-duda que podrá resolver.

Deshabilitar el procesamiento de las transacciones distribuidas

Si DISTRIBUTED_TRANSACTIONS se pone a cero, ninguna declaración SQL distribuida se podrá emitir en cualquier sesión. Además el proceso RECO (*Oracle RECOverer Process proceso de fondo que resuelve transacciones en duda*) no

empezará sino empieza la instancia local. Las transacciones en duda que pueden estar presentes, Oracle no las puede resolver automáticamente. Por lo que, solo se coloca el parámetro de inicialización a cero, para prevenir a transacciones distribuidas que una nueva instancia está empezando, y ninguna transacción dudosa permanece después del cierre de la última instancia.

4.5.2. Especificar el commit point strength de un nodo

La base de datos con él más alto commit point strength determina que nodo se actualizará primero en la transacción distribuida. Al especificar un commit point strength para cada nodo, se asegura que el servidor más crítico no se bloquee si ocurre una falla durante una fase de preparación o actualización. El siguiente parámetro de inicialización determina un commit point strength de un nodo.

```
COMMIT_POINT_STRENGTH = Valor
```

El valor por defecto es operado por el sistema dependiente. El rango de valores es cualquier entero comprendido entre 0 y 255. Por ejemplo, para poner al commit point strength de una base de datos a 200, se debe incluir la siguiente línea en el archivo del parámetro de inicialización de la base de datos y solo se debe usar para determinar el commit point site de una transacción distribuida.

```
COMMIT_POINT_STRENGTH = 200
```

Al configurar el commit point strength de una base de datos, se debe considerar los siguientes aspectos:

- El commit point strength almacena información acerca del estado de la transacción, por lo que no debe ser un nodo inestable o indisponible, ya que otros nodos necesitan información acerca del estado de la transacción.
- El commit point strength para una base de datos, se coloca para compartir la cantidad de datos más críticos de una base de datos. Por ejemplo; una base de datos en un computador mainframe usualmente comparte más datos a los usuarios que una base de datos en una PC. Por lo tanto, el commit point strength del mainframe tiene un valor más alto.

4.6. Ver la información de las transacciones distribuidas

El diccionario de datos de cada base de datos almacena información de todas las transacciones distribuidas abiertas. Se puede usar las tablas y vistas del diccionario de datos para aprovechar la información de las transacciones.

4.6.1. Nombrar la transacción

Es muy útil para identificar una transacción distribuida específica y reemplazar el uso de la declaración COMMIT COMMENT.

Para nombrar una transacción, se usa la declaración SET TRANSACTION ... NAME, por ejemplo:

```
SET TRANSACTION NIVEL AISLAMIENTO SERIALIZABLE
NAME 'actualizar inventario en el punto de control  0';
```

Este ejemplo muestra que el usuario empezó una nueva transacción con el nivel de aislamiento igual a SERIALIZABLE y lo denominó 'actualizar inventario en el punto de control 0'.

Para las transacciones distribuidas, el nombre se envía al sitio de participación cuando una transacción se actualiza. Si existe un COMMIT COMMENT, éste es ignorado si existe un nombre de transacción.

El nombre de la transacción se visualiza en la columna NAME de la vista V$TRANSACTION y se coloca en la columna TRAN_COMMENT de la vista cuando la transacción se actualiza.

4.7. Manejo de transacciones in-doubt (en duda)

Se puede forzar hacer manualmente un commit o un rollback local, a una transacción distribuida en duda, pero puede ocasionar problemas de consistencia. Esta actividad únicamente se lo debe realizar en condiciones específicas como las siguientes:

Una aplicación de usuario que actualiza una transacción distribuida, informa de un problema a través de los siguientes mensajes de error:

ORA-02050: transaction ID rolled back, some remote dbs may be in-doubt (*La transacción ID hizo un rollback, algún servidor de base de datos remoto puede estar en duda, debido a fallas en la red o en el servidor remoto durante el commit de dos fases*).

ORA-02051: transaction ID committed, some remote dbs may be in-doubt (*La transacción ID actualizo, algún servidor de base de datos remoto puede estar en duda, debido a fallas en una sesión en el mismo sitio con la misma transacción global ID*).

ORA-02054: transaction ID in-doubt (*La transacción ID en duda, debido a fallas en la red o en el servidor remoto en el commit de dos fases*).

Una aplicación robusta debe guardar información de una transacción que haya recibido cualquiera de los errores anteriores. Esta información puede ser utilizada si se desea recuperar manualmente una transacción distribuida.

Una de las características de Oracle es la recuperación automática de cualquier transacción en duda de forma transparente en todos los nodos del árbol de sesión, luego de que se solucione la falla en la red o en el sistema.

En fallas prolongadas, a menudo, se puede forzar hacer un commit o un rollback a una transacción para liberar algunos datos seguros. Las aplicaciones deben considerar tales posibilidades.

4.7.1. Determinar cuando realizar manualmente una transacción en duda

Se puede forzar manualmente una transacción en duda, cuando una de las siguientes situaciones ocurre:

- La transacción en duda asegura los datos que son requeridos por otras transacciones. Esta situación ocurre cuando el mensaje de error ORA-01591 (*Aseguramiento de datos de una transacción distribuida en duda, se intento acceder a recursos seguros durante el commit de dos fases mientras la*

transacción estaba en estado de preparación) interfiere con las transacciones de usuario.

- Una transacción en duda previene que las extensiones de segmentos rollback puedan ser usados por otras transacciones. La primera parte de una transacción distribuida local en duda (transacción ID) corresponde al ID del segmento rollback, como en el listado del diccionario de datos de la vista DBA_2PC_PENDING y DBA_ ROLLBACK_SEGS.
- El fracaso ocurrido en el commit de dos fases impide que la fase se complete o se corrija en un período de tiempo aceptable. Los ejemplos de estos casos incluyen a la red de telecomunicaciones que se ha dañado o ha una base de datos dañada que requiere un tiempo largo para su recuperación.

Normalmente, se debe tomar una decisión localmente para forzar una transacción distribuida en duda consultando con administradores de otras ubicaciones. Una decisión equivocada puede ocasionar inconsistencias en la base de datos que pueden ser difíciles de arreglar y se deben corregir manualmente.

4.7.2. Analizar los datos de la transacción

Si se decide obligar a que la transacción se complete, se debe analizar la información disponible con las siguientes metas.

Encontrar un nodo para el commit o rollback

La vista DBA_2PC_PENDING se usa para encontrar un nodo que tenga un commit o un rollback. Si se puede encontrar un nodo que ha resuelto la transacción, entonces se puede seguir la acción tomada en ese nodo.

Buscar los comentarios de la transacción

La vista DBA_2PC_PENDING proporciona información de la transacción distribuida dentro de la columna TRAN_COMMENT. Los comentarios se incluyen en la cláusula COMMENT de la declaración COMMIT. El nombre de la transacción se coloca en el campo TRAN_COMMENT cuando la transacción se actualiza.

Por ejemplo, el siguiente comentario de la transacción distribuida en duda, indica el origen de la transacción y que tipo de transacción es.

```
COMMIT COMMENT
'Finanzas/CuentasporPagar/TipodeTransaccion 10B';
```

La declaración SET TRANSACTION … NAME también puede ser usada (de preferencia) para proporcionar información de la transacción en un nombre.

4.8. Realizar manualmente una transacción en duda

Se usa la declaración COMMIT o ROLLBACK junto con la opción FORCE y la cadena de texto que indica el ID de la transacción en duda local o global.

4.8.1. Realizar manualmente un commit a una transacción en duda

Antes de intentar actualizar una transacción, se debe tener privilegios para actualizar manualmente una transacción en duda (Ver tabla 26).

Si la transacción es actualizada por:	Entonces debe tener el privilegio de:
Usted	FORCE TRANSACTION
Otro usuario	FORCE ANY TRANSACTION

Tabla 26. Privilegios para actualizar manualmente una transacción en duda

Actualizar usando solo el ID de la transacción

La declaración SQL siguiente actualiza una transacción en duda:

```
COMMIT FORCE 'transaccion_id';
```

Por ejemplo, al consultar la vista DBA_2PC_PENDING se determina la transacción local ID para la transacción distribuida:

```
LOCAL_TRAN_ID 1.45.13
```

Entonces se emite la siguiente declaración SQL para forzar la actualización de la transacción en duda.

```
COMMIT FORCE '1.45.13';
```

Actualizar usando SCN (Número de Cambio de Sistema)

Opcionalmente, se puede especificar el SCN para forzar a actualizar la transacción. Esta característica permite actualizar una transacción en duda con el SCN que se asignó cuando este actualizo en otros nodos.

Consecuentemente, se puede mantener sincronizado el tiempo de actualización de la transacción distribuida incluso si hay fallas. Solo se debe especificar un SCN cuando se ha determinado el SCN de la misma transacción que ha realizado una actualización en otro nodo.

Por ejemplo, asumir que se desea actualizar manualmente una transacción con el siguiente ID de la transacción global:

```
VENTAS.NEGOCIO.COM.55d1c563.1.93.29
```

Primero, se debe consultar la vista DBA_2PC_PENDING de una base de datos remota también implicada con la transacción en cuestión. Se nota que el SCN fue utilizado para

actualizar la transacción en ese nodo. Específicamente el SCN se usa para actualizar la transacción en un nodo local. Por ejemplo, si el SCN es 829381993, la declaración sería:

```
COMMIT FORCE
'VENTAS.NEGOCIO.COM.55d1c563.1.93.29',829381993;
```

4.8.2. Realizar manualmente un rollback a una transacción en duda

Antes de hacer un rollback a una transacción, se debe tener los siguientes privilegios para hacer manualmente un rollback (Ver tabla 27).

Si la transacción es actualizada por:	Entonces debe tener el privilegio de:
Usted	FORCE TRANSACTION
Otro usuario	FORCE ANY TRANSACTION

Tabla 27. Privilegios para hacer un rollback manualmente una transacción en duda

La declaración SQL siguiente hace un rollback a una transacción en duda:

```
ROLLBACK FORCE 'transaccion_id';
```

Por ejemplo: Si el ID de la transacción en duda es 2.9.4, la declaración del rollback sería:

```
ROLLBACK FORCE '2.9.4';
```

CAPITULO V

REPLICACIÓN DE DATOS EN ORACLE

5.1. Introducción

La replicación es el proceso de copiar y mantener objetos de bases de datos, como tablas; en múltiples bases de datos se constituye en un sistema de base de datos distribuido. Los cambios aplicados en un sitio son capturados y almacenados localmente antes de ser enviados a cada nodo remoto.

La replicación usa tecnología de base de datos distribuida para compartir datos entre múltiples sitios, pero una base de datos replicada y una base de datos distribuida no es lo mismo. En una base de datos distribuida, los datos están disponibles en muchos nodos, pero una tabla en particular reside en un solo sitio. Por ejemplo, la tabla ***empleados*** reside únicamente en la base de datos ***ambato.mundo*** en un sistema de base de datos distribuido que también incluye las bases de datos ***quito.mundo*** y ***cuenca.mundo***. Replicación significa que los mismos datos están disponibles en múltiples sitios. Por ejemplo, la tabla ***empleados*** está disponible en ***ambato.mundo***, ***quito.mundo***, y ***cuenca.mundo***.

5.1.1. Ventajas

Disponibilidad, la replicación mejora la disponibilidad de aplicaciones porque proporciona opciones de acceso a datos. Si un sitio no está disponible, los usuarios pueden continuar con la consulta o incluso actualizar los nodos restantes. En otras palabras, la replicación proporciona excelente protección cuando se presentan fallos.

Funcionabilidad, la replicación proporciona un rápido acceso local a los datos compartidos, por que equilibra las actividades en múltiples sitios. Algunos usuarios pueden acceder a un servidor mientras que otros usuarios acceden a diferentes servidores, reduciendo de esta forma la carga en todos los servidores. Los usuarios pueden acceder a los datos del sitio replicado que tiene el costo más bajo de acceso, el cual es el sitio más cercano a ellos geográficamente.

Operabilidad, una vista materializada es una copia parcial o completa (réplica) de una tabla, en un momento dado. Las vistas materializadas permiten a los usuarios trabajar en un subconjunto de base de datos, cuando se han desconectado del servidor de base de datos central. Luego, cuando una conexión se establece, los usuarios pueden sincronizar (refrescar) las vistas materializadas en demanda. Cuando los usuarios refrescan las vistas materializadas, los usuarios pueden actualizar la base de datos central y recibir cualquier cambio que sucedió mientras ellos estuvieron desconectados.

Reducción de carga en la red, la replicación puede ser usada para distribuir datos sobre múltiples sitios regionales. Las aplicaciones pueden tener acceso a varios servidores regionales en lugar de hacerlo a un solo servidor central. De esta forma se reduce drásticamente la carga en la red.

Despliegue de masa, cada vez más, las organizaciones necesitan desplegar muchas aplicaciones, por lo que se requiere de habilidad para usar y manipular los datos. Con la replicación de Oracle, las plantillas de despliegue permiten crear múltiples vistas materializadas rápidamente. Se puede usar variables para personalizar cada vista materializada según las necesidades. Por ejemplo, se pueden usar plantillas de despliegue para automatizar las ventas. En este caso, las plantillas podrían contener variables de varias regiones de ventas y sus vendedores.

5.1.2. ¿Porqué usar replicación?

Desde el punto de vista básico, la replicación se utiliza para asegurar que los datos estén disponibles cuando y dónde se los necesite. La entrega de información depende de los diferentes ambientes de replicación existentes. Un ambiente de replicación se categoriza de la siguiente forma:

- Distribución de datos a otras ubicaciones
- Consolidación de datos desde otras ubicaciones

- Intercambio bidireccional de datos con otras ubicaciones
- Alguna variante o combinación de los anteriores

5.1.3. Distribución de datos a otras ubicaciones

La distribución de datos involucra el movimiento de todos o un subconjunto de datos de una o más ubicaciones. Los datos pueden ser copiados a un almacén de datos (DataWarehouse, *base de datos separada, dedicada al soporte de decisiones. Los datos que transfiere son de sistemas de transacciones procesadas e integradas*) o a un sistema de toma de decisiones. A menudo esto implica la transformación de datos y desnormalización. Los conjuntos de los datos pueden ser copiados a DataMarts (*Subconjunto de información de un grupo de usuarios, los cuales transfieren por separado a un servidor departamental. La base de datos involucrada puede ser relacional, aunque en un servidor multidimensional OLAP es más apropiado*) para proveer acceso local a grupos de usuarios. Esto permite jerarquizar los datos de la empresa utilizando herramientas inteligentes y mantener la seguridad y la ejecución de las aplicaciones.

La distribución también es usada para proveer datos a aplicaciones desarrolladas en plataformas similares o diferentes. Esto puede ser tan simple como mantener una copia de los datos de producción en otro sistema similar, así como requerir una transformación compleja para adaptarse a los requerimientos de una nueva aplicación.

Los datos que se copien pueden necesitar ser filtrados o transformados para la nueva aplicación. La distribución también puede ser usada para proveer coexistencia a dos sistemas, en la instancia de migración de uno al otro.

La figura 14. muestra la replicación desde un servidor fuente hacia dos servidores destino.

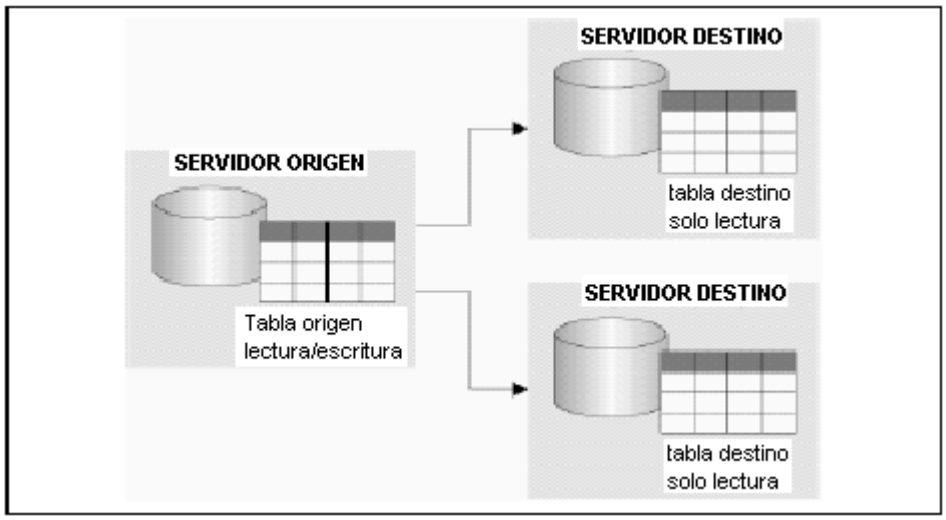


Figura 14. Distribución de datos

Los servidores destino podrían mantener un subconjunto diferente de los datos de acuerdo a las necesidades propias.

5.1.4. Consolidación de datos desde otras ubicaciones

Una empresa puede tener sus datos en distintos sistemas distribuidos. Por ejemplo, una empresa de ventas puede tener sus datos en cada local, una empresa de manufacturación

los puede tener en cada planta de procesamiento, así como una compañía aseguradora podría tenerlos en cada una de sus oficinas o llegar incluso a cada computador personal de sus corredores. La replicación puede copiar los cambios de cada sitio distribuido al sitio central, para analizar, hacer reportes o para el procesamiento central de los datos.

La consolidación de datos puede ser muy útil para aplicaciones de Negocios Inteligentes, como pueden ser aplicaciones OLAP (*Proceso de análisis en línea, sistema que permite la manipulación operacional de los datos, trabajando rápida y flexiblemente*) o DataMining (*Proceso de detección de modelos ocultos, tendencias y datos asociados, que se almacenan en una gran base de datos como warehouse*).

La figura 15. muestra la replicación desde dos servidores fuentes, hacia un servidor destino.

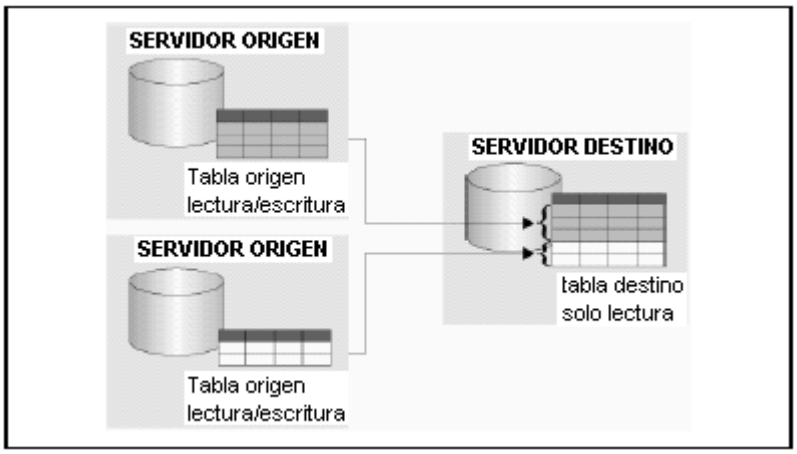


Figura 15. Consolidación de datos

La estructura de los datos en cada origen o fuente es la misma. En términos SQL, los datos del destino son la unión de los datos de los distintos orígenes.

5.1.5. Intercambio bidireccional de datos con otras ubicaciones

Si los datos se pueden modificar en múltiples ubicaciones, entonces la replicación debe procesar los cambios realizados en cada uno de los sitios de forma coordinada. Uno de los servidores, es el servidor maestro, quien se encarga de distribuir los cambios a todos los sitios. Los cambios realizados en los destinos van hacia los otros sitios a través del servidor maestro.

La replicación bidireccional puede ser usada para aplicaciones móviles donde los destinos pueden ser tanto una computadora de una oficina, como una laptop de un camión de entregas. A menudo hay varios destinos que se conectan ocasionalmente al sistema fuente, esta conexión puede ser a través de líneas telefónicas, por lo qué la eficiencia es importante. Esta forma de replicar es conocida como master-slave (maestro-esclavo).

La figura 16. muestra la replicación bidireccional, en la que se tiene un servidor maestro designado.

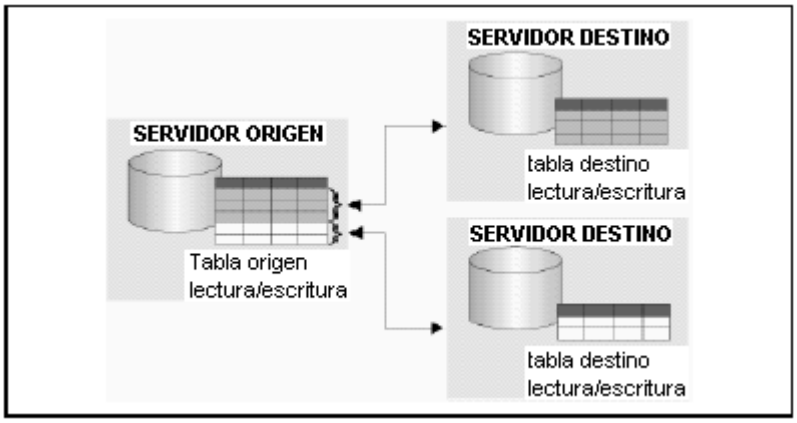


Figura 16. Replicación bidireccional

Otro tipo de replicación bidireccional, es aquella que no tiene designada un servidor maestro. Cada ubicación copia los cambios desde todos los otros sitios directamente.

Esto habitualmente se conoce como replicación “multi-master” o “peer-to-peer”. La replicación “peer-to-peer” puede ser usada para mantener sitios recuperables ante posibles desastres o caídas, así como para proveer sistemas con alta disponibilidad y para balancear la carga de consultas a través de las distintas ubicaciones.

La figura 17. muestra una configuración “peer-to-peer”.

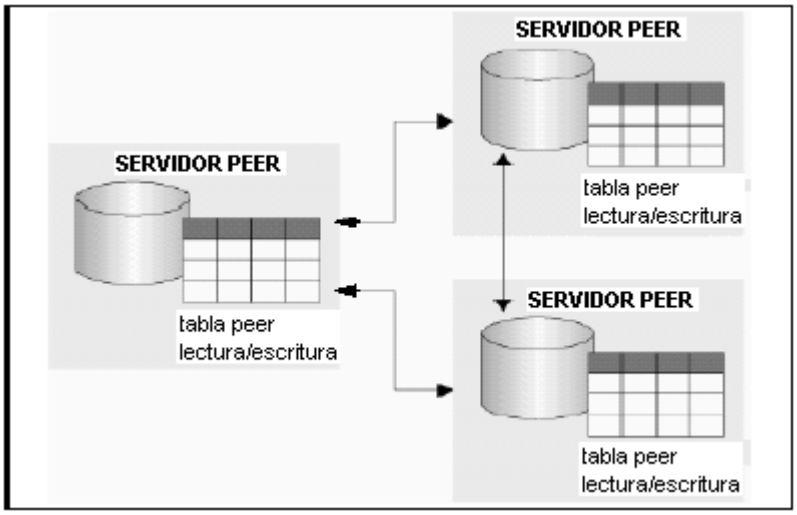


Figura 17. Replicación peer-to-peer

5.1.6. Otros requerimientos

Algunas aplicaciones podrían no requerir una copia completa de los datos pero si necesitar un registro de todos los cambios que han ocurrido. Esto es útil para mantener una auditoria del sistema y también puede ser útil para proveer los cambios para procesos especiales. Por ejemplo, los cambios realizados en un sistema fuente pueden ser almacenados en una tabla intermedia, donde se analicen y se validen, antes de procesarlos y enviarlos a los sitios destino. Esta es una variante de la técnica de distribución de datos, en la cual el destino sólo recibe los cambios.

Los cambios también pueden ser utilizados en escenarios de replicación multi-capas, donde son copiados desde una fuente central hacia un área de almacenamiento en otro sistema y finalmente desde esta área a los sitios destino. Esto es muy útil en caso de existir muchos sitios destino, ya que minimiza el impacto sobre el sistema fuente.

5.2. Conflictos de replicación de datos

Existen varios tipos de conflictos a tener en cuenta en la replicación de datos. Los conflictos pueden ocurrir cuando se esta trabajando en un ambiente de replicación que permite actualizaciones concurrentes sobre los mismos datos en múltiples sitios. Entre estos tenemos:

Conflicto de actualización

Un conflicto de actualización ocurre cuando se produce la replicación de un update sobre un registro con otro update sobre el mismo registro. Este conflicto ocurre cuando dos transacciones originadas desde distintos sitios actualizan el mismo registro, en forma cercana en el tiempo.

Conflicto de unicidad

El conflicto de unicidad sucede cuando la replicación de un registro intenta violar una restricción de integridad, ya sea por Primary Key o Unique. Por ejemplo, cuando dos transacciones originadas de dos sitios diferentes, cada una inserta un registro, en su respectiva tabla replicada, con el mismo valor de clave primaria. En ese caso sucede un conflicto de unicidad.

Conflicto de supresión

Un conflicto de supresión ocurre cuando dos transacciones originadas de sitios diferentes, una de ellas intenta borrar un registro, y la otra actualizar o borrar el mismo registro, ya que en este caso el registro no existe, tanto para ser actualizado como borrado.

Conflicto de orden

Los conflictos de orden pueden ocurrir en ambientes de replicación con tres o más sitios maestros. Si la propagación al sitio maestro X, está bloqueada por alguna razón, entonces la replicación de modificaciones a datos puede seguir siendo propagada a través de los otros sitios maestros; al finalizar la propagación estas modificaciones deben propagarse al sitio X en un orden diferente al que ocurrió en los otros sitios maestros, pudiendo producirse conflictos.

5.3. Resolución de conflictos

Conflicto de actualización

Para estos casos existen tres formas aceptadas de resolverlos:

- Prioridad: Cada servidor tiene una prioridad única, y el servidor de mayor prioridad es el que gana, respecto de aquellos con prioridad menor.

- Timestamp: La más nueva o la más antigua de las modificaciones es la considerada correcta, por defecto, sino se eligió alguno de los criterios gana la más nueva.
- Particionamiento de datos: Se garantiza que cada registro sea manipulado por un único servidor, lo que simplifica la arquitectura.

Conflicto de unicidad

También existen tres formas aceptadas de resolverlos:

- A cada servidor darle un rango distinto de números para los generadores de clave (secuencias).
- Agregar el identificador del servidor a la clave primaria.
- Replicar en tablas separadas, y acceder a los datos a través de una vista formada por la unión de ellas. Para resolver el conflicto de potenciales claves duplicadas en la unión se usará una pseudo columna que representa la base de datos fuente.

Conflicto de supresión

Para evitar este tipo de conflictos, una solución es que los sitios marquen lógicamente los registros a ser borrados y que periódicamente el sitio maestro corra un proceso que realice el borrado físico de los datos, es decir que desde los sitios replicados no se pueda ejecutar una sentencia delete.

Conflicto de orden

Este tipo de conflictos se pueden resolver asignando distintas prioridades a los sitios maestros, de tal forma que ordene las transacciones de acuerdo a su prioridad.

5.4. Replicación en oracle 9i

La replicación en Oracle 9i soporta dos tipos de estrategias:

- **Distribución Masiva**, en la cual los datos deben ser periódicamente sincronizados entre un servidor central y una gran cantidad de clientes en ubicaciones remotas que operan en forma desconectada.
- **Distribución de Servidor a Servidor**, en la cual la información se encuentra distribuida en varios servidores que deben mantenerse sincronizados de manera permanente, casi como si fuera en tiempo real, para asegurar la continua disponibilidad del servicio.

Oracle integra estas estrategias en un ambiente coherente y provee también una interfaz gráfica para controlar desde un punto central todo el ambiente replicado.

5.4.1. Sincronización servidor a servidor

En Oracle 9i se permite la propagación de datos como si fuera en tiempo real entre varios servidores o sitios maestros de igual jerarquía, interconectados. Esta forma de replicación "Multimaster" permite que tablas completas se mantengan casi simultáneamente replicadas entre varios servidores. Oracle 9i extiende el soporte de replicación para los tipos de datos definidos por el usuario como objetos, tablas y tablas con tablas anidadas.

La replicación Multimaster emplea llamadas a procedimientos remotos diferidas, como mecanismos de transporte para propagar y aplicar los cambios.

Un usuario puede elegir entre las varias rutinas preconstruidas para la resolución de conflictos, algunas se basan en la prioridad del sitio, otras en el timestamp; a su vez estas rutinas se pueden elegir para cada tabla o grupos de columnas de una tabla.

También permite que el usuario defina sus propias rutinas para resolución de conflictos.

5.4.2. Capturar y aplicar la replicación de cambios

Se utilizan triggers para la captura y almacenamiento de las transacciones en una cola local. Los triggers y paquetes de aplicación son internos, están programados en C y compilados con el kernel de Oracle, para ganar mayor ejecución.

5.4.3. Propagar los cambios

Se utiliza múltiples sesiones, para paralelizar el flujo de transacciones hacia cada ubicación remota, de manera automática. Para preservar el orden de las transacciones se aplican en la ubicación remota en el mismo orden que se aplicaron localmente, serializando el flujo de transacciones. Se utiliza un pequeño proceso coordinador a nivel local para determinar las dependencias entre transacciones antes de propagarlas.

5.4.4. Minimizar los datos propagados

El usuario puede elegir enviar solo el mínimo de información necesaria para aplicar los cambios en un sitio remoto. Para ello puede enviar los valores de las columnas de la clave primaria, los nuevos valores de las columnas que fueron modificadas, y algún valor necesario para la resolución de conflictos, el "timestamp" por ejemplo.

5.4.5. Aplicaciones de distribución masiva

Oracle provee la replicación masiva debido a que cada vez más organizaciones reconocen la necesidad de automatizar la disponibilidad de datos actualizados en sitios remotos, generalmente desconectados o con conexiones de baja calidad.

Existen cuatro desafíos que enfrenta esta técnica:

- Sincronización a demanda de los sitios destino
- Soporte para configuraciones jerárquicas
- Propagación de la mínima cantidad de información necesaria
- Soporte para administración centralizada de los sitios replicados

Para satisfacer estos requerimientos Oracle proporciona vistas materializadas o snapshots (*instantáneas, copia de una tabla en un sistema remoto*) que pueden definirse para poder contener una copia completa de una tabla maestra, o un subconjunto definido de filas de la tabla maestra basados en un criterio de selección. Las vistas materializadas se actualizan desde las tablas maestras a intervalos fijos de tiempo o a demanda. Se emplean llamadas a procedimientos remotos diferidas como el mecanismo de transporte para propagar y aplicar los cambios.

5.4.6. Soporte para operaciones sin conexión

Decrementar los tiempos de conexión es un requerimiento clave en la mayoría de las operaciones sin conexión. El mejorar la ejecución se centra en dos tipos de optimización:

- Un protocolo que minimiza el número de intercambios requeridos para replicar los cambios necesarios.
- Optimizaciones de mecanismos que permitan detectar aquellas tablas maestras sobre las que no se han producido cambios desde la última sincronización con el sitio remoto.

5.4.7. Vistas materializadas jerárquicas

Oracle soporta una organización jerárquica de los sitios replicados, la estructura de los mismos puede ser representada en forma de árbol, donde los nodos intermedios actúan a la vez como sitios origen y destino.

De esta forma los cambios se propagan de manera transitiva entre vistas materializadas predefinidas. Los snapshots intermedios, pueden efectuar y propagar la resolución de conflictos, para actualizaciones que provienen de otro snapshot.

5.4.8. Particionamiento de la información a replicar

El esquema se basa en vistas lo que permite fácilmente especificar tanto particiones horizontales como verticales basándose en consultas SQL.

El usuario define el subconjunto de datos a replicar con una sentencia SELECT como parte del comando para crear los snapshots. Usando ciertas posiciones en las subconsultas se puede fácilmente definir y mantener subconjuntos únicos de datos para un gran número de ubicaciones remotas. Las subconsultas siguen las referencias establecidas por las claves foráneas para permitir que una tabla hija haga referencia a información mantenida por la tabla padre. Se soportan relaciones de muchos a muchos entre tablas así como también condiciones AND a través de múltiples filas o condiciones OR sin filas. El mecanismo de actualización de Oracle permite el intercambio de información hacia y desde los snapshots a sus tablas maestras asociadas.

5.4.9. Administración central de sitios remotos con snapshots

Dentro del Administrador Empresarial de Oracle (*AEO - Oracle Enterprise Manager OEM*) se incluye el Conjunto de Herramientas de Oracle (*CHO - Oracle Management Replication Tool OMRT*) con la finalidad de mantener, administrar y monitorear varios sitios remotos desconectados. Las plantillas para distribución masiva permiten definir una colección parametrizada de vistas materializadas en cada sitio maestro. Se pueden

reducir los tiempos de conexión pre instanciando las vistas materializadas en la ubicación central.

También almacena la información acerca del ambiente replicado en un catálogo de replicación, el cual se puede consultar por medio de sentencias SQL normales.

5.5. Aplicaciones híbridas

Combinando las técnicas de replicación antes descritas, Servidor a Servidor y Distribución Masiva se pueden definir otras estrategias de replicación que permitan adaptarse a las necesidades.

Específicamente soporta la existencia de un conjunto de sitios master replicados usando la técnica Servidor a Servidor, los que a su vez actúan como sitios master en la técnica de replicación masiva. (Ver Figura 18.)

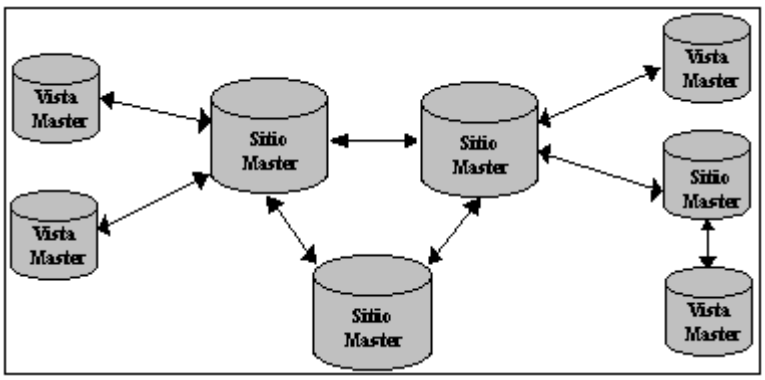


Figura 18. Aplicación híbrida

5.6. Oracle streams

Oracle Streams, más que un flujo de datos, es una tecnología para compartir información. Detecta que información es relevante y quienes la utilizan.

Esta tecnología es utilizada por Oracle para propagar los cambios en un ambiente replicado. Se basa en tres acciones aplicadas a la información: Captura, Almacenamiento y Consumo.

En la replicación la *CAPTURA* se relaciona con un mecanismo que toma los cambios del 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 con replicas. Y el *CONSUMO* es la máquina que se encarga de aplicar los cambios almacenados a la base de datos en cada equipo con replicas.

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.

5.7. Replicación de objetos, grupos y sitios

Replicación de objetos

Un objeto replicado es un objeto de base de datos que existe en múltiples servidores dentro de un sistema de base de datos distribuido. En un ambiente de replicación, cualquier actualización realizada a un objeto replicado en un sitio se aplicará a todas las copias existentes en otros sitios. La replicación permite replicar los siguientes tipos de objetos: Tablas, Indices, Vistas y vistas de objetos, Paquetes y cuerpos de paquetes, Procedimientos y funciones, Tipos definidos por usuarios y tipos de cuerpos, Triggers, Sinónimos, Tipos de Indice, y Operadores definidos por el usuario.

En cuanto a las tablas, la replicación soporta características avanzadas tales como: Tablas particionadas, tablas organizadas por índice, tablas que contienen columnas que se basan en tipos definidos por usuarios y tablas de objeto.

Replicación de grupos

En un ambiente de replicación, Oracle maneja objetos replicados usando grupos replicados. Un grupo replicado es una colección de objetos replicados que lógicamente se encuentran relacionados, logrando así administrar muchos objetos juntos.

Típicamente, se crea y se usa la replicación de un grupo para organizar los objetos de esquema necesarios para soportar una aplicación de base de datos particular. Sin embargo, los grupos replicados y los esquemas no necesitan corresponderse entre ellos.

Un grupo de replicación puede contener objetos de esquemas múltiples y un esquema solo puede tener objetos en múltiples grupos replicados. Sin embargo, cada objeto replicado puede ser miembro de sólo un grupo replicado.

Replicación de sitios

Un grupo replicado puede existir en múltiples sitios replicados. El ambiente replicado soporta básicamente dos tipos de sitios replicados: Sitio Master (maestro) y Sitio de Vistas Materializadas. Un sitio puede ser, un sitio master para un grupo replicado y un sitio de vistas materializadas para un grupo de replicación diferente. Sin embargo, un sitio no puede ser sitio master y sitio de vistas materializadas para el mismo grupo replicado.

Las diferencias entre estos dos sitios son las siguientes:

- Un grupo replicado en un sitio master que esta más específicamente referenciado como un grupo master. Un grupo replicado en un sitio de vista materializada se basa en un grupo master y esta específicamente referenciado como un grupo de vista materializada. Adicionalmente, cada grupo master tiene un sitio master

definido. El sitio master definido de un grupo replicado sirve como centro de control para manejar el grupo replicado y los objetos del grupo.

- Un sitio master mantiene una copia completa de todos los objetos en un grupo replicado, mientras que las vistas materializadas dentro de un sitio de vista materializada pueden contener todo o un subconjunto de los datos de la tabla dentro de un grupo master. Por ejemplo, si el grupo master Control de Representantes (***cr_rep***) contiene las tablas ***empleados*** y ***departamentos***, entonces todos los sitios master que participan en un grupo master deben mantener una copia completa de ***empleados*** y ***departamentos***. Sin embargo, un sitio de vista materializada podría contener sólo una vista materializada de la tabla ***empleados***, mientras que otro sitio de vista materializada podría contener las vistas materializadas de las dos tablas ***empleados*** y ***departamentos***.
- Todos los sitios master en un ambiente de replicación multimaster, se comunican directamente entre ellos para continuamente propagar cambios en los datos del grupo replicado. Los sitios de vista materializadas contienen una imagen o una vista materializada, de los datos de la tabla desde un cierto intervalo de tiempo. Típicamente una vista materializada es refrescada periódicamente para sincronizarse con su sitio master. Se puede organizar las vistas materializadas dentro de un grupo refresh para que puedan pertenecer a uno o más grupos de vistas materializadas de tal forma que puedan ser refrescados al mismo tiempo y asegurar que los datos en todas las vistas materializadas corresponden a la misma transacción.

5.8. Tipos de replicación

5.8.1. Replicación master

En una replicación master, un sitio master soporta un solo sitio de vistas materializadas, sin embargo, su capacidad puede ser incrementada utilizando los mecanismos que hacen que soporte cientos o miles de sitios de vistas materializadas.

Un sitio master que soporta uno o más sitios de vistas materializadas también puede participar en múltiples sitios master, tan solo creando un ambiente de replicación híbrida que se consigue al combinar replicación multimaster y replicación de vistas materializadas.

5.8.2. Replicación multimaster (peer to peer o n-ways)

Es aquella que incluye en sí, múltiples sitios master que participan de igual forma en cualquier modelo de actualización. La actualización hace que un sitio master envíe estos cambios hacia todos los sitos master participantes como se muestra en la figura 19.

Los servidores de base de datos de Oracle operan como sitios master en una replicación multimaster y controlan la convergencia de los datos de todas las tablas replicadas,

asegurando la consistencia de la transacción global y la integridad de los datos. Si surge algún conflicto cada sitio lo resuelve por separado.

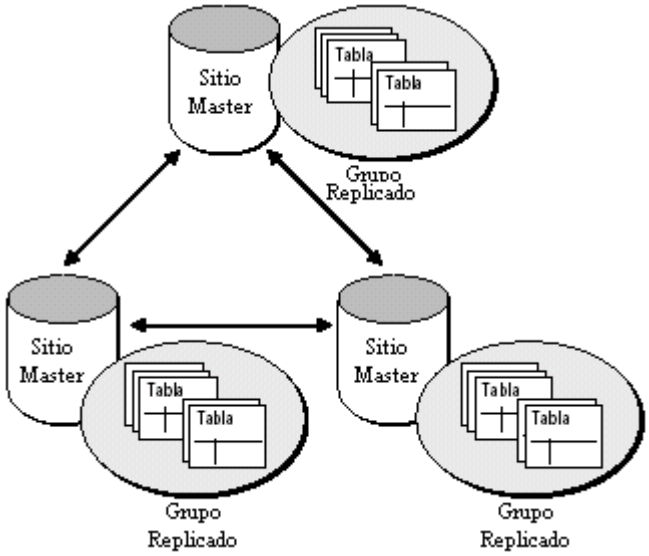


Figura 19. Replicación multimaster

Sitio de definición master

En un ambiente de replicación multimaster, un sitio master opera como el sitio de definición master de un grupo master. Este sitio en particular realiza múltiples tareas administrativas y de mantenimiento para la replicación multimaster.

Cada grupo master puede tener solo un sitio de definición master, aunque este puede ser cualquiera de los sitios master en el ambiente multimaster. Adicionalmente, el sitio de definición master puede ser cambiado hacia diferentes sitios master si es necesario.

Un solo sitio master que soporta replicación de vistas materializada es por defecto el sitio de definición master.

En un ambiente de replicación multimaster se pueden usar tres tipos de replicación: asíncrona, síncrona y procedural.

5.8.3. Replicación asíncrona

Es el camino más común para implantar la replicación multimaster. Cuando se usa replicación asincrónica, se capturan los cambios locales, se los almacena en una cola y luego se los propaga al resto de sitios master remotos a intervalos de tiempos regulares (*transacciones diferidas*).

La utilización de replicación asincrónica implica conflictos en los datos porque el mismo valor de fila podría ser colocado en dos sitios master diferentes casi al mismo tiempo. Sin embargo, se pueden usar técnicas para evitar estos conflictos y si estos ocurren, Oracle proporciona mecanismos preconstruidos que pueden ser implementados para resolver estos problemas, los mismos que se almacena en un log de errores.

La figura 20. muestra el proceso de la replicación de datos asíncrona.

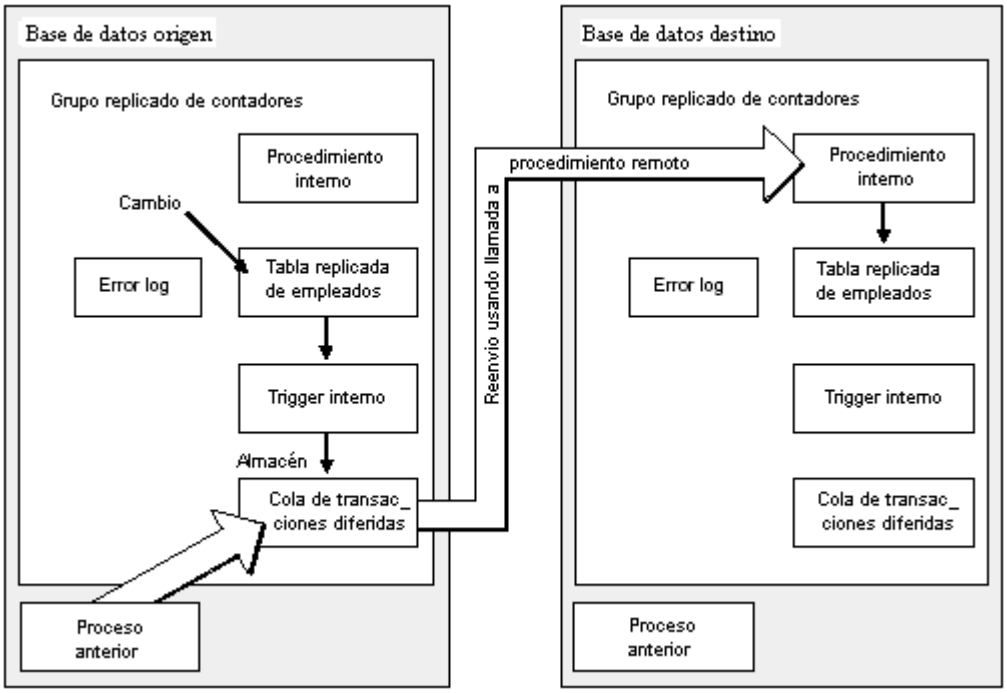


Figura 20. Replicación de datos asíncrona

5.8.4. Replicación síncrona

Conocida también como replicación en tiempo real, es aquella en donde al realizar un cambio en los datos o ejecutar cualquier procedimiento replicado en todos los sitios participantes, se lo hace como una sola transacción.

Cuando se usa replicación síncrona, la actualización de una tabla implica un cambio inmediato en todos los sitios master participantes, sin embargo, si uno de los sitios master no puede procesar una transacción por cualquier razón, se ejecutará un rollback en todos los sitios.

La replicación síncrona requiere de un ambiente muy estable, debido a que si la comunicación a un sitio master no es posible por problemas en la red, los usuarios pueden seguir realizando consultas a las tablas replicadas y como resultado se obtendrá que ninguna de estas transacciones se completará hasta que la red sea reestablecida.

Nota: Es posible configurar la replicación asincrónica para que simule una replicación síncrona.

La figura 21. muestra el proceso de la replicación de datos síncrona.

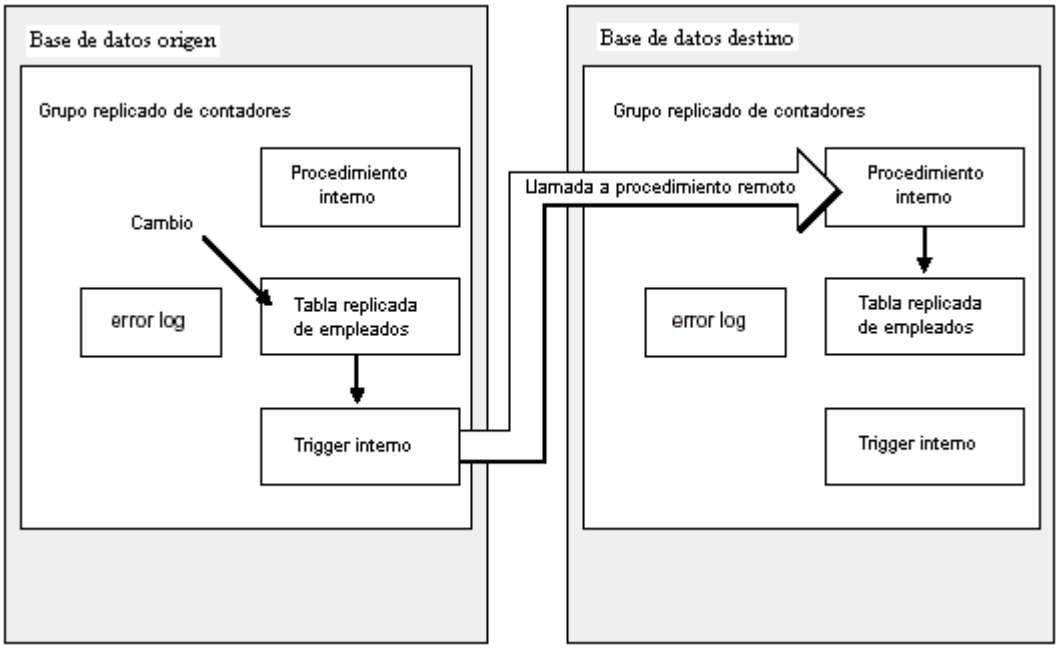


Figura 21. Replicación de datos síncrona

5.8.5. Replicación procedural

Replica sólo la llamada a un procedimiento almacenado que una aplicación usa para actualizar una tabla. No replica las modificaciones de los datos.

En la replicación procedural, se deben replicar los paquetes que modifican los datos en todos los sitios del sistema. Después de replicar un paquete, se debe generar un wrapper para el paquete en cada sitio. Cuando una aplicación llama a un procedimiento empaquetado para modificar los datos en el sitio local, el wrapper asegura que todos los sitios llamen al mismo paquete.

Su objetivo es disminuir el tráfico en la red producido por aplicaciones que realizan actualizaciones por lotes (*replicación de filas*).

La replicación procedural puede ocurrir de manera asíncrona o síncrona.

Grupo Master Quiesce (*temporalmente inactivo o deshabilitado*)

A veces, se debe detener toda actividad de replicación de un grupo master para poder realizar ciertas tareas administrativas en el grupo. Por ejemplo, al agregar un nuevo objeto al grupo master. Esta actividad de detener toda replicación en el grupo master se la conoce como ***quiesce***.

Cuando el grupo master esta en quiesced, los usuarios no pueden emitir declaraciones DML para cualquiera de los objetos del grupo. También todas las transacciones diferidas pueden ser propagadas antes de hacer un quiesced al grupo master. Los usuarios pueden continuar consultando las tablas cuando el grupo master esta en quiesced.

5.8.6. Replicación de vistas materializadas (snapshots)

El término snapshot y vista materializada en la documentación de Oracle son sinónimos.

Una vista materializada contiene una copia completa o parcial de un sitio master designado en un determinado tiempo. Una vista materializada es una vista materializada que funciona como master para otra vista materializada. Una vista materializada multitier (multihilera) es aquella que se basa en otra vista materializada, en lugar de una tabla master.

Las vistas materializadas tienen los siguientes beneficios:

- Habilitan el acceso local, mejorando el tiempo de respuesta y disponibilidad.
- No cargan de consultas al sitio master o al sitio master de vistas materializadas, porque los usuarios pueden consultar la vista materializada local.

- Incrementan la seguridad de los datos, al permitir replicar solo un subconjunto seleccionado de datos del sitio master designado.

Las vistas materializadas pueden ser de solo lectura, actualizable, o escribible.

Vistas materializadas solo de lectura (read only)

En una configuración básica, una vista materializada puede proporcionar acceso de sólo lectura a los datos de una tabla que se originan de un sitio master o de un sitio master de vista materializada. Las aplicaciones pueden consultar datos solo lectura de vistas materializadas para evitar el acceso a la red del sitio master, sin importar si la red esta disponible. Sin embargo, las aplicaciones deben tener acceso a los datos del sitio master para ejecutar cambios en el DML.

Las vistas materializadas de solo lectura tienen las siguientes ventajas:

- Eliminan la posibilidad de conflictos porque no pueden actualizar.
- Soportan vistas materializadas complejas. Por ejemplo, aquellas vistas que contienen operaciones con la cláusula CONNECT BY.

La figura 22. muestra la replicación de vista materializada de solo lectura

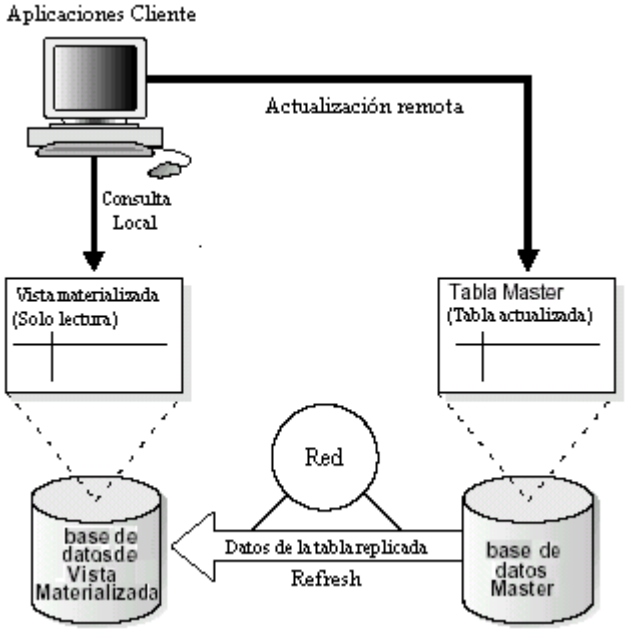


Figura 22. Replicación de vistas materializadas de solo lectura

Ejemplo de una vista materializada solo de lectura:

```
CREATE MATERIALIZED VIEW control.empleados AS
SELECT * FROM control.empleados@orc1.mundo;
```

Vistas materializadas actualizables (updatable)

Este tipo de vista permite a los usuarios insertar, actualizar y borrar filas de la tabla master designada o de las vistas materializadas master. Una vista materializada actualizable puede contener un subconjunto de datos en el sitio master designado.

La figura 23. muestra un ambiente de réplica usando una vista materializada actualizable.

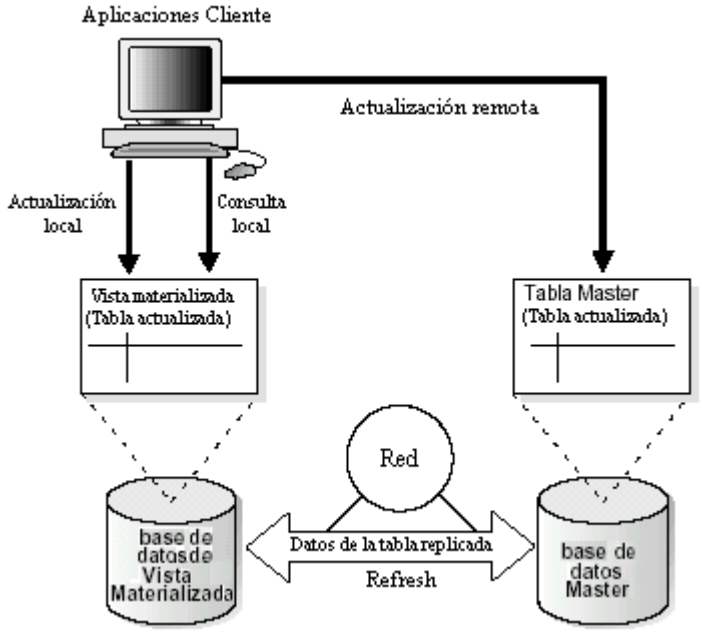


Figura 23. Replicación de una vista materializada actualizable

Una vista materializada actualizable tiene las siguientes propiedades:

- Se basan siempre en una sola tabla, aunque múltiples tablas pueden ser referenciadas en una subconsulta.
- Pueden tener un tiempo rápido para ser refrescados.
- Oracle propaga los cambios hechos por una vista materializada actualizable, a la tabla master remota de la vista materializada o a la vista materializada master. La actualización master se realiza en cascada en todos los otros sitios replicados.

Las ventajas de las vistas materializadas actualizables son:

- Permite a los usuarios consultar y actualizar datos locales replicados, incluso si esta desconectado del sitio master o del sitio master de vistas materializadas.
- Requiere de menos recursos que la replicación multimaster para la actualización de los datos, puesto que se refrescan en demanda, mientras que la replicación multimaster propaga los cambios a intervalos regulares.

Ejemplo de una vista materializada actualizable:

```
CREATE MATERIALIZED VIEW control.departamentos
FOR UPDATE AS
SELECT * FROM control.departamentos@orc1.mundo;
```

La siguiente declaración crea un grupo de vista materializada:

```
BEGIN
DBMS_REPCAT.CREATE_MVIEW_REPGROUP (
gnombre => 'cr_repg',
master => 'orc1.mundo',
propagation_mode => 'ASYNCHRONOUS');
END;
/
```

La siguiente declaración añade cr.departamentos al grupo de vista materializada, haciendo la vista materializada actualizable:

```
BEGIN
DBMS_REPCAT.CREATE_MVIEW_REPOBJECT (
gnombre => 'cr_repg',
snombre => 'cr',
onombre => 'departamentos',
type => 'SNAPSHOT',
min_communication => TRUE);
END;
/
```

Vistas materializadas escribibles (writetable)

Se puede crear una vista materializada usando la cláusula *FOR UPDATE* durante la creación, pero entonces nunca se añadirá la vista materializada a un grupo de vista materializada. En este caso, los usuarios pueden hacer cambios en el DML, pero estos cambios no se realizarán en el sitio master y por lo tanto se perderán al refrescar la vista materializada. Estas vistas materializadas son llamadas vistas materializadas escribibles.

Refresh de vistas materializadas

Para asegurar que una vista materializada es consistente con una tabla master o con una vista materializada master, se debe ejecutar un refresh periódicamente.

Existen tres métodos para realizar un refresh:

- **Refresh rápido,** usa los logs de las vistas materializadas para actualizar solo las filas que han cambiado desde el último refresh.
- **Refresh completo,** actualiza la vista materializada completa.

- **Refresh forzado,** realiza un refresh rápido si es posible, cuando este no es posible, se obliga a ejecutar un refresh completo.

Refresh de grupos

Es importante para las vistas materializadas que las transacciones sean consistentes con todos los demás sitios. Al refrescar el grupo se asegura que los datos en todas las vistas materializadas correspondan a la misma transacción consistente en un determinado tiempo. Las vistas materializadas actualizables y solo de lectura pueden incluirse en un grupo refresh.

Actualmente se usa la vista materializada por consistencia por que se refiere a un objeto de base de datos que contiene los resultados de una consulta de un o más tablas.

Las tablas en la consulta son llamadas tablas maestras (*un término de replicación*) o tablas de detalle (*un término de data warehouse*). Aquí se usa “tablas maestras” por consistencia. Las bases de datos que contienen las tablas maestras son llamadas base de datos maestras.

Para la replicación, los snapshots permiten mantener copias de datos remotos en un nodo local. Estas copias pueden ser actualizadas con características avanzadas para la replicación y si no se usan estas, son solo de lectura. Puede seleccionar datos de un

snapshot provenientes de una tabla o de una vista. En ambientes de replicación, los snapshots tienen una clave primaria, rowid, y vistas materializadas de consultas.

Para propósitos de data warehousing, los snapshots que se crean son vistas materializadas agregadas, vistas materializadas agregadas, basadas en una sola tabla, y vistas materializadas producto de un JOIN. Los tres tipos de snapshot pueden ser usadas para consultas actualizables, en técnicas de optimización que transforman las solicitudes de los usuarios en términos de tablas maestras hacia solicitudes equivalentes, que incluye uno o más snapshots. En un ambiente de data warehouse, todas las tablas maestras deben ser locales.

Los privilegios requeridos para crear un snapshot deben ser dados directamente. Para crear un snapshot en el esquema actual:

- Debe haber sido concedido privilegios para crear un snapshot y tener privilegios de sistema de CREATE TABLE o CREATE ANY TABLE.
- Debe también tener acceso a cualquier tabla maestra del snapshot de la cual no se es propietario, así mismo tener privilegios de SELECT sobre cada tabla o privilegios de sistema de SELECT ANY TABLE.

Para crear un snapshot en un esquema de otro usuario:

- Debe tener privilegios de sistema para CREATE ANY SNAPSHOT o CREATE ANY MATERIALIZED VIEW y acceso a las tablas maestras base del snapshot de las cuales no se es propietario, de igual forma se debe tener privilegios de SELECT sobre cada una de las tablas que usa el snapshot o privilegios del sistema de SELECT ANY TABLE.
- El propietario del snapshot debe tener privilegios de sistema de CREATE TABLE. El propietario debe también tener acceso a cualquier tabla maestra del snapshot de cual no se es propietario y para cualquier log para el snapshot definido previamente para el propósito, por supuesto hace falta privilegios del sistema para SELECT así como de SELECT ANY TABLE.

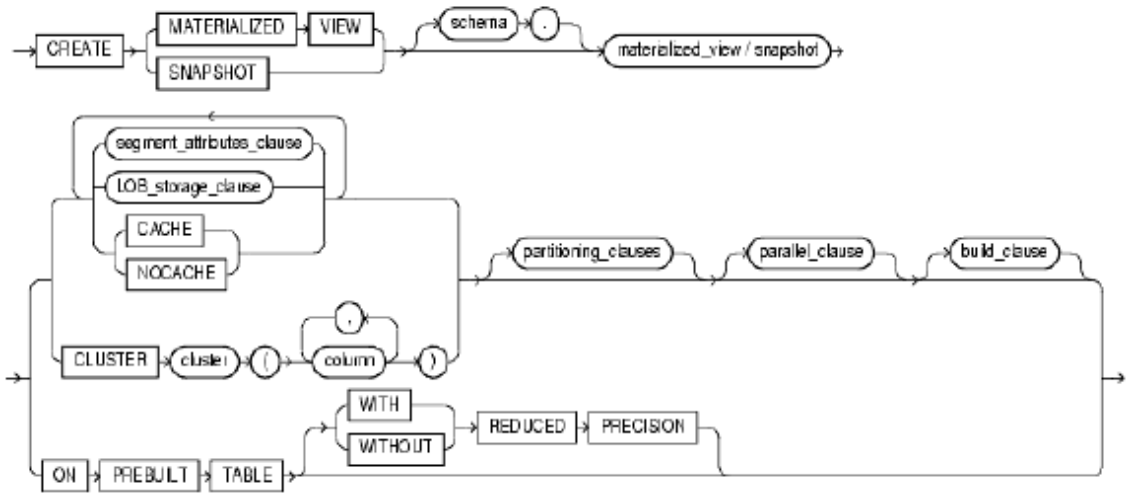
Para crear un snapshots que sea de consulta reescribible, además de los privilegios antes enumerados:

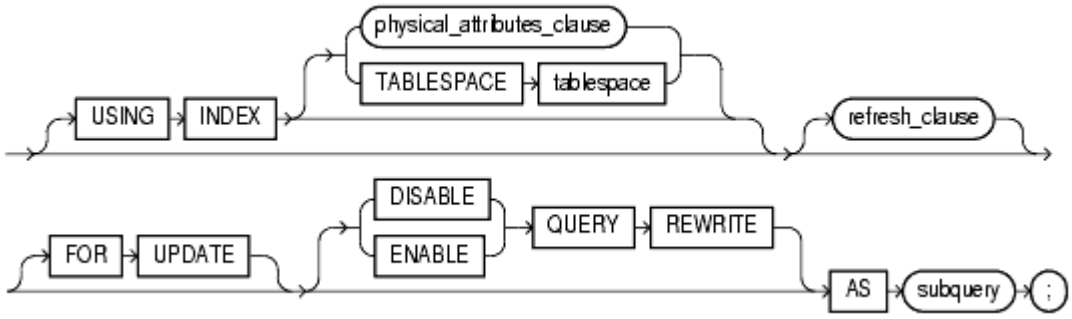
- El propietario de las tablas maestras debe tener privilegios de sistema de QUERY REWRITE.
- Si no se es el propietario de las tablas maestras, es necesario tener privilegios del sistema de GLOBAL QUERY REWRITE.
- Si el propietario de esquema no es propietario no es propietario de las tablas maestras debe tener los privilegios del sistema GLOBAL QUERY REWRITE.

El usuario cuyo esquema contiene el snapshot debe tener una cuota de espacio de tabla suficiente para almacenar la tabla maestra y el índice para el snapshot, o debe tener el privilegio del sistema UNLIMITED TABLESPACE.

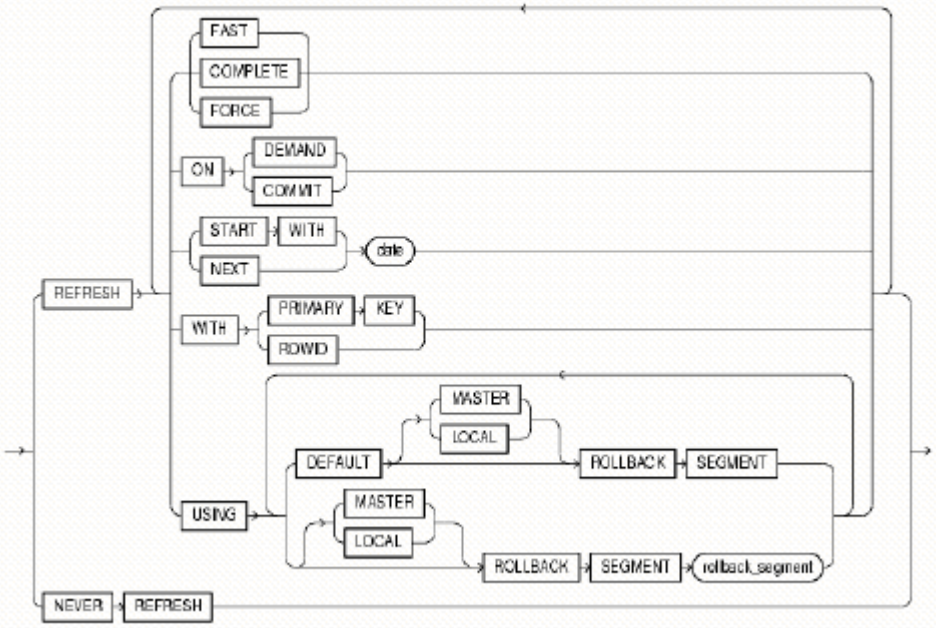
Cuando se crea un snapshot, Oracle crea una tabla interna y por lo menos un índice, y puede crear una vista, todo esto el esquema del snapshot. Oracle usa estos objetos para mantener los datos del snapshot. Debe tener los privilegios necesarios que le permitan crear estos objetos.

Sintaxis:

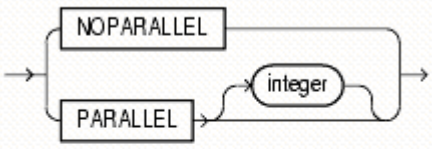




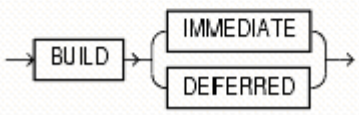
Claúsula Refresh



Claúsula Parallel



Claúsula Build



Palabras Claves y Parámetros

Schema, especifica el esquema que contiene el snapshot, Si se omite schema, Oracle crea el snapshot en el esquema actual.

materializad_view, especifica el nombre de la vista materializada a ser creada. Oracle genera nombres para las tablas e índices usados para mantener la vista materializada, esto lo hace adicionando un prefijo o sufijo al nombre de la vista materializada.

segment_atribute_clause, especifica el segment_atribute_clause para establecer valores para los parámetros PCTFREE, PCTUSED, INITRANS, y MAXTRANS (o, cuando se usa la cláusula USING INDEX, para los parámetros INITRANS y MAXTRANS únicamente), las características de almacenamiento, para asignar a los espacios de tabla, y para especificar si el acceso se va a dar.

TABLESPACE, especifica el espacio de tabla en el cual el snapshot va a ser creado. Si se omite esta cláusula, Oracle crea por defecto el snapshot en el espacio de tabla del propietario del esquema del snapshot.

LOB_storage_clause, permite especificar las características de almacenamiento para un LOB.

LOGGING|NOLOGGING, especifica las características de acceso para el snapshot.

CACHE|NOCACHE, para los datos que son accesados frecuentemente, CACHE especifica que los bloques recuperados por esta tabla son colocados al final de la más recientemente usada en la LRU, listados en el búfer de la caché, cuando se haya realizado un completo escaneo de la tabla. Este atributo resulta útil para búsquedas en pequeñas tablas. NOCACHE especifica que los bloques son colocados al final de la más recientemente usada en la lista de la LRU.

CLUSTER, se usa esta cláusula para crear el snapshot como parte de un clúster específico. Un snapshot de clúster usa el espacio de almacenamiento del clúster. Por consiguiente no use las cláusulas physical_attributes_clause o TABLESPACE con la cláusula CLUSTER.

partitioning_clauses, permite especificar que el snapshot esta particionado en rangos específicos de valores o funciones hash. El particionado de los snapshots es el mismo de las tablas.

parallel_clause, permite indicar si operaciones paralelas serán soportadas por el snapshot, y un conjunto de grados de paralelismo para consultas y DML sobre estas luego de la creación de los mismos.

build_clause, permite especificar cuando rellenar el snapshot.

- **INMEDIATE,** indica que el snapshot es rellenado inmediatamente. Esto es por defecto.
- **DEFERRED**, el snapshot será rellenado cuando se realice el siguiente REFRESH.

ON PREBUILT TABLE, esta cláusula permite registrar una tabla existente como un preiniciliazado snapshot. Esto es útil para registrar snapshot con gran cantidad de datos en ambientes de data warehousing.

USING INDEX, establece los valores para los parámetros de INITRANS, MAXTRANS y STORAGE que los índices de Oracle usa para mantener los datos de un snapshot. Si no se especifica, entonces los valores por defecto son usados por el índice.

refresh_clause, se lo utiliza para especificar métodos por defecto para modos, y tiempos en que Oracle hará un refresco de los datos del snapshot. Si una tabla maestra del snapshot es modificada, los datos del snapshot deben también ser actualizados para hacer que el snapshot muestre la información actual de sus tablas maestras. Esta cláusula permite calendarizar el tiempo y especificar el método y forma como Oracle refrescará el snapshot.

- **FAST**, se usa FAST para indicar un método de refrescamiento incremental, el cual realiza el refrescamiento acorde a los cambios que ocurran en las tablas maestras.
- **COMPLETE**, se utiliza para indicar un método de refrescamiento completo, que es implementado de acuerdo a la definición del snapshot. Oracle realiza un refresco completo, tan rápido como sea posible.
- **FORCE**, indica cuando se va a producir el refresco. Oracle realizará un refresco rápido si no se puede realizar uno completo. Si no se indica el modo de refresco, FORCE es el por defecto.
- **ON COMMIT,** especifica que un refresco rápido esta por ocurrir en el momento en el que Oracle realice un commit a una transacción, que opera sobre una tabla maestra del snapshot.
- **ON DEMMAND,** para indicar que el snapshot será refrescado bajo demanda, mediante el llamado a uno de los tres procedimientos de refrescamiento. Si omite tanto ON COMMIT y ON DEMMAND, este último se hará por defecto.

FOR UPADTE, se utiliza para permitir a una consulta, clave primaria, o snapshot rowid, ser actualizada. Cuando es usado junto con la replicación avanzada, las actualizaciones serán propagadas a la tabla maestra.

QUERY REWRITE, esta cláusula permite especificar si el snapshot es posible de ser usado para consultas de reescritura.

- **ENABLED**, permite que el snapshot sea para una consulta de reescritura.
- **DISABLE**, usado para indicar que el snapshot no es consulta de reescritura. Sin embargo es posible refrescar un snapshot que esta desactivado.

AS subquery, define el snapshot como una consulta. Cuando se crea un snapshot, Oracle ejecuta la consulta y coloca los resultados en el snapshot. Esta puede ser cualquier consulta SQL válida. Sin embargo no todas las consultas se refrescan rápidamente.

Si se quiere que el snapshot sea elegible para un refresco rápido, se debe utilizar logs de snapshot, restricciones adicionales se presentan para esto.

También podemos cambiar un snapshot o borrarlo en forma definitiva, esto lo hacemos con las sentencias:

- ALTER SNAPSHOT
- DROP SNAPSHOT

5.8.7. Configuración híbrida de vistas materializadas y multimaster

La replicación multimaster y de vistas materializadas pueden ser combinadas en configuraciones híbridas o mixtas para encontrar aplicaciones con diferentes requerimientos. Estas configuraciones pueden tener cualquier número de sitios master y sitios de vista materializadas múltiples para cada sitio master.

Por ejemplo, la figura 24. muestra la replicación multimaster entre dos sitios masters, que soportan replicaciones de tablas llenas en dos bases de datos de regiones diferentes.

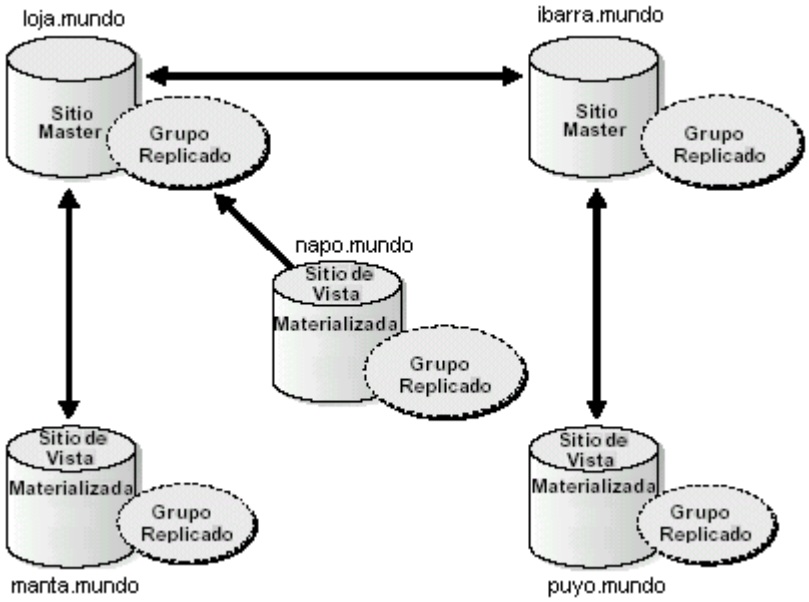


Figura 24. Configuración híbrida

La diferencia de clave entre vistas materializadas y tablas master replicadas son:

- La replicación de tablas master deben tener datos de las tablas a replicarse, mientras que las vistas materializadas pueden replicar subconjuntos de datos de las tablas master.
- La replicación multimaster permite replicar los cambios de cada transacción, es decir, como ocurren. La vista materializada se refresca, propagando los cambios de múltiples transacciones y haciéndolas más eficientes, ya que orienta las operaciones de lote, pero a intervalos menos frecuentes.
- Si los conflictos ocurren debido a los cambios realizados en las múltiples copias de los mismos datos, entonces la detección y resolución de conflictos siempre ocurren en un sitio master o en un sitio de vista materializada master.

5.9. Parámetros de inicialización

La tabla 28. lista los parámetros de inicialización que son importantes para la operación y fiabilidad de un ambiente de replicación.

Parámetros	Valores	Descripción	Recomendación
COMPATIBLE	Por defecto: 8.1.0 Rango: 8.1.0 hasta un número de versión actual.	Permite usar una nueva versión, al mismo tiempo que garantiza la compatibilidad con la versión nueva. Los servidores Oracle que tienen niveles de compatibilidad diferentes pueden interoperar.	Para usar cualquiera de las características nuevas de replicación, se debe poner este parámetro a 9.0.1 o más alto.

Tabla 28. (cont)

Parámetros	Valores	Descripción	Recomendación
GLOBAL_NAMES	Por defecto: falso Rango: verdadero o falso	Especifica si se requiere de un link de base de datos con el mismo nombre que la base de datos a la cual está conectada.	GLOBAL_NAMES debe ser verdadero en todas las bases de datos que participan en la replicación, incluyendo los sitios master y de vistas materializadas.
JOB_QUEUE_PROCESES	Por defecto: 0 Rango: 0 a 1000	Especifica el número de procesos en la cola de trabajo Tn para cada instancia (t000 a t999). Los procesos en la cola de trabajo manejan peticiones creadas por DBMS_JOB. Cuando JOB_QUEUE_PROCESES se pone a 0 en un sitio, las peticiones administrativas se realizan manualmente para todos los grupos del sitio así como también en la cola de las transacciones diferidas. El valor de JOB_QUEUE_PROCESES se lo puede cambiar con la cláusula ALTER SYSTEM	Este parámetro debe ser puesto a 1, y se debe poner hasta el valor del número máximo de trabajos que pueden ejecutarse simultáneamente, más uno.

Tabla 28. (cont)

Parámetros	Valores	Descripción	Recomendación
OPEN_LINKS	Por defecto: 4 Rango: 0 a 255	Especifica el número máximo de conexiones simultáneas abiertas a bases de datos remotas en una sesión. Estas conexiones incluyen los objetos de esquema llamados link de base de datos, así como procedimientos externos. Los cuales usan procesos separados.	Si se usa replicación síncrona, el parámetro OPEN_LINKS debe ser puesto por lo menos al número de sitios master. Por ejemplo en un ambiente con 5 sitios master OPEN_LINKS debe ser puesto por lo menos a 5.
PROCESSES	Por defecto: Se deriva de PARALLEL_MAX_ SERVERS Rango: 6 hasta el límite del sistema de operación	Especifica el número máximo de los procesos de usuarios del sistema de operaciones que simultáneamente pueden conectarse a Oracle.	Asegurar que el valor de este parámetro tenga en cuenta todos los procesos de fondo, tales como seguridades, procesos en la cola de trabajo, y procesos de ejecución paralela.

Tabla 28. Parámetros de inicialización importantes para la replicación

5.9.1. Preparar vistas materializadas

El mayor problema encontrado en la replicación de vistas materializadas, es no preparar apropiadamente el ambiente de replicación. Hay cuatro tareas esenciales que se deben realizar antes de comenzar a crear un ambiente de vistas materializadas:

- Crear los esquemas necesarios.
- Crear los links de base de datos necesarios.
- Asignar los privilegios apropiados.
- Asignar suficientes procesos de trabajo.

Estas tareas son realizadas automáticamente al ejecutar el asistente de las herramientas de replicación.

5.9.2. Crear usuarios en el sitio de vistas materializadas

Cada sitio de vista materializada necesita de varios usuarios para realizar la administración y el refresh, en los sitios de vistas materializadas. Por lo que se deben crear y conceder los privilegios necesarios al administrador de vistas materializadas.

5.9.3. Crear esquemas en sitios de vistas materializadas

Un esquema que contiene una vista materializada en una base de datos remota debe corresponder al esquema que contiene la tabla master en la base de datos master. Por lo tanto, se debe identificar los esquemas que contienen las tablas master que se desean replicar con vistas materializadas. Después de tener identificado los esquemas destino en la base de datos master, crear las correspondientes cuentas con los mismos nombres en la base de datos remota. Por ejemplo, si todas las tablas master están en el esquema

ventas de la base de datos ***ambato.mundo***, entonces crear el correspondiente esquema ***ventas*** en la base de datos de vista materializada ***manta.mundo***.

5.9.4. Crear links de base de datos

Antes de crear las vistas materializadas, se necesita crear varios administradores de links de base de datos. Específicamente, se debe crear un link de base de datos público desde el sitio de vista materializada al sitio master. Los link privados son mas fáciles de crear debido a que no necesitan incluir la cláusula USING en cada link. Se necesita un link de base de datos privado desde el administrador de vista materializada al administrador proxy y desde el propagador al receptor. Pero si se usa el asistente de las herramientas de replicación, los link se crean de forma automática.

La figura 25. muestra un esquema recomendado y la configuración de link de base de datos.

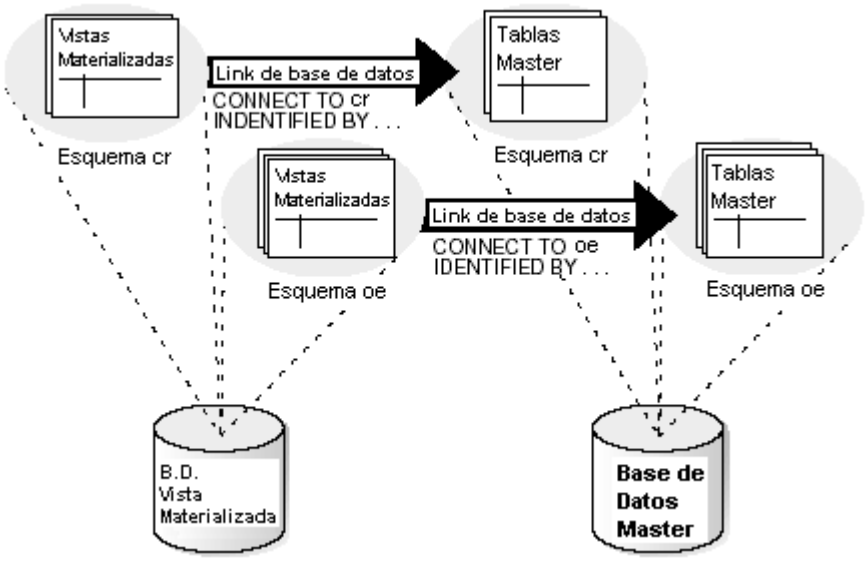


Figura 25. Esquema recomendado y configuración del link de base de datos

5.9.5. Asignar privilegios

El creador o propietario de la vista materializada debe emitir la definición de la declaración SELECT de la vista materializada, es decir, debe tener los privilegios para crear y administrar las vistas materializadas. Los privilegios que debe tener son CREATE MATERIALIZED VIEW y SELECT.

5.9.6. Asignar procesos a la cola de trabajo

Esta asignación es importante para poder manejar la automatización del ambiente de replicación. Los procesos en la cola de trabajo automáticamente se propagan a la cola de

transacciones diferidas, limpian la cola de transacciones diferidas, refrescan las vistas materializadas, y así sucesivamente.

En la replicación multimaster, cada sitio tiene un link fijo para cada uno de los otros sitios master. Por ejemplo, si se tiene seis sitios master, entonces cada sitio tiene un link fijo para los otros cinco sitios. Se necesita de un proceso para cada link fijo. También se puede añadir adicionalmente un proceso de trabajo para limpiar la cola de transacciones diferidas y otros trabajos definidos por el usuario.

5.9.7. Crear log para vistas materializadas (instantáneas)

Antes de crear grupos de vistas materializadas y vistas materializadas para sitios remotos, es necesario crear logs de vistas materializadas en el sitio master o en el sitio de vista materializada. El log de vista materializada es necesario para cada tabla master o vista materializada master, ya que soporta al menos una vista materializada con un refresh rápido.

Para crear un log de vista materializada, se necesitan los siguientes privilegios:

```
CREATE ANY TABLE
CREATE ANY TRIGGER
SELECT (en el log de vista materializada master)
COMMENT ANY TABLE
```

El registro de instantáneas contiene datos transitorios: registros insertados (INSERT) en él, que se utilizan durante los refrescos y luego son borrados (DELETE). Por tanto, debería de reutilizar los bloques de los registros de instantáneas, asignando un valor alto a pctused cuando cree el registro de instantáneas.

Si múltiples instantáneas utilizan la misma tabla maestra, entonces compartirán el mismo registro de instantáneas. Si una de las instantáneas no se refresca durante un largo período de tiempo, entonces puede ocurrir que el registro de instantáneas nunca borre algunos de sus registros. Como resultado de ello, las necesidades de espacio del registro de instantáneas aumentarán.

Un registro para una instantánea (snapshot) se crea con la orden CREATE SNAPSHOT LOG. Este es una tabla asociada con la tabla maestra base del snapshot.

Los términos instantánea (snapshot) y vista materializada (materializad view) son sinónimos. Las dos hacen referencia a una tabla que los resultados de una consulta de una o más tablas, las cuales pueden estar localizadas en la misma base de datos o en una remota.

Cuando se usa lenguaje de manipulación de datos (DML) las cambios son hechos sobre la tabla de datos maestra, Oracle almacena filas describiendo estos cambios en el registro del snapshot, luego esta es utilizado para refrescar el snapshot basado en la

tabla maestra. Esto proceso es llamado "*refrescamiento rápido*". Sin un registro para el snapshot, Oracle debe ejecutar una y otra vez la consulta que esta en el snapshot para refrescar los datos del mismo. Usualmente un refresco rápido toma menos tiempo que un refrescamiento completo.

Un registro de snapshot es localizado en la base de maestra, en el mismo esquema que se encuentra la tabla maestra. Se necesita únicamente un registro para una tabla maestra del snapshot. Oracle puede usar este registro para realizar todos los refrescamientos más rápidos de todos los snapshots que se basen en la tabla maestra.

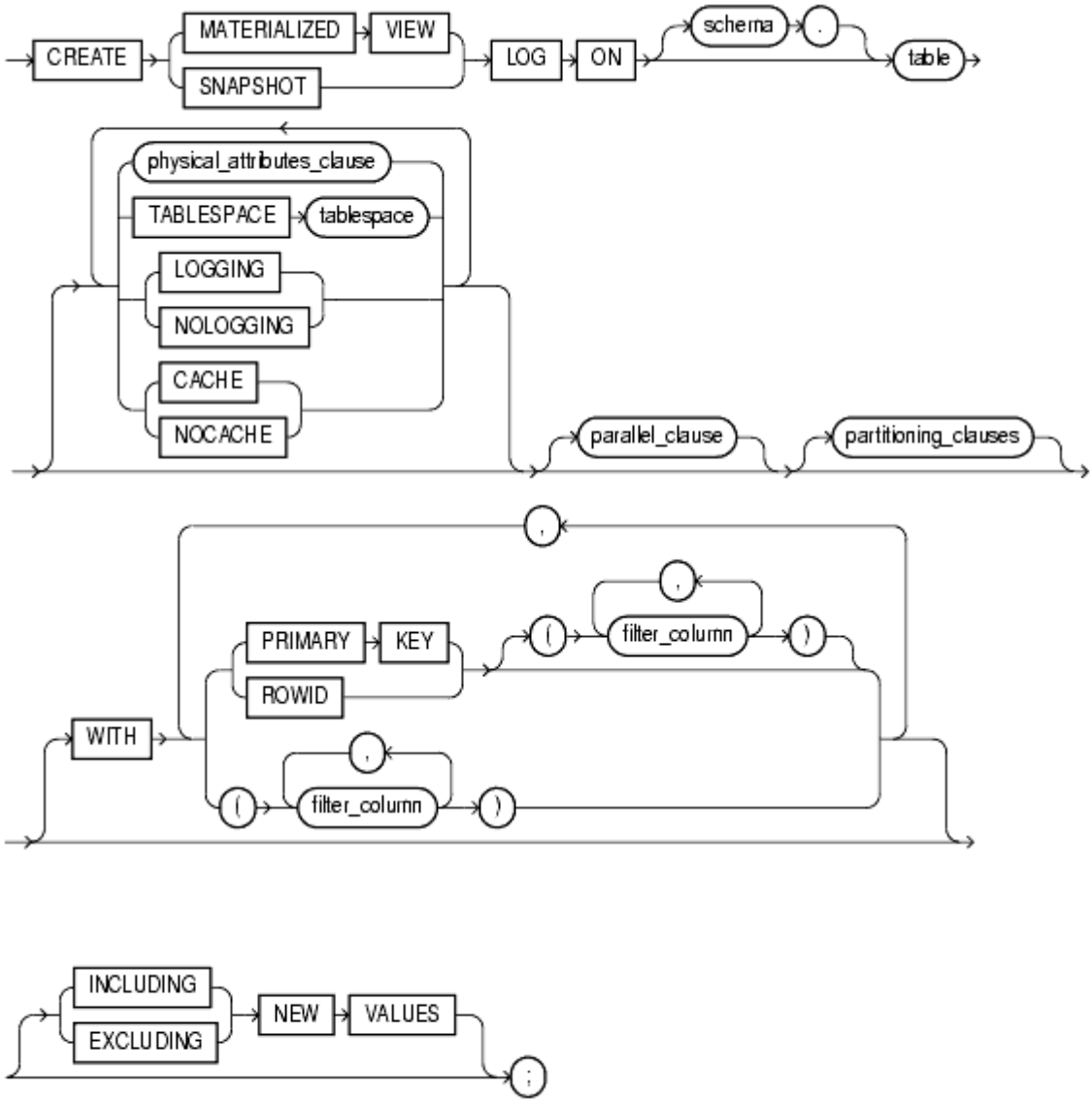
Para realizar un refresco de un snapshot que contiene un JOIN, se debe crear un log por cada tabla maestra que se use en el JOIN para realizar el snapshot.

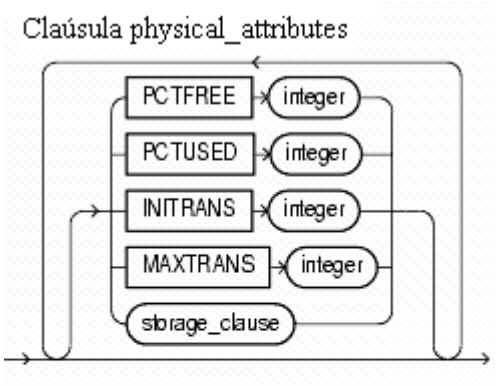
Los privilegios requeridos para crear un registro para un snapshot están directamente relacionados con los objetos que usa el snapshot para realizar su propósito.

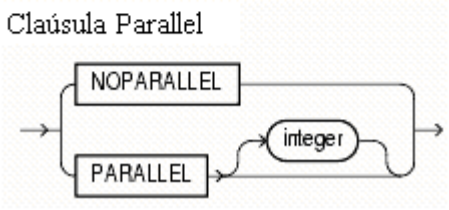
- Si es propietario de la tabla maestra, puede crear un registro para el snapshot.
- Si se esta creando un registro para un snapshot de una tabla que esta en otro esquema, se debe tener los privilegios CREATE ANY TABLE y COMMENT ANY TABLE, así como también privilegios de SELECT y SELECT ANY TABLE sobre la tabla maestra.

En cualquier caso los propietarios del registro deben tener cuota de espacio de tablas suficiente, por lo que debe tener el privilegio del sistema UNLIMITED TABLESPACE.

Sintaxis







Palabras claves y parámetros

Schema, especifica el esquema que contiene el registro de la tabla maestra. Si se omite, Oracle asume que la tabla maestra pertenece al esquema actual. Oracle crea un registro dentro del esquema de la tabla maestra. No esta permitido crear un registro para una tabla dentro del esquema del usuario SYS.

Table, indica el nombre de la tabla maestra para el cual el registro se va a crear. No se puede crear un registro para una vista.

physical_attributes_clause, esta cláusula establece varios valores sobre las características físicas y de almacenamiento del snapshot.

TABLESPACE, indica el espacio de tabla en el cual el registro será creado. Al omitir esta cláusula, Oracle creará el registro en el esquema del usuario actual.

LOGGING|NOLOGGING, especifica un LOGGING o NOLOGGING para establecer las características de ingreso para el registro del snapshot.

CACHE|NOCACHE, para los datos que serán accesados frecuentemente. CACHE especifica los bloques recuperados serán colocados al final del más recientemente usado del registro en el LRU, para hacer más rápido la recuperación de la información. NOCACHE realiza un proceso similar, colocando los bloques de datos al final de los menos recientemente usados espacios del registro en el LRU.

parallel_clause, permite indicar si las operaciones paralelas serán soportadas por el registro del snapshot.

- **NOPARALLEL**, para ejecución serial, este es valor por defecto.
- **PARALLEL**, si se desea que Oracle seleccione un grado de paralelismo igual al número de CPU´s disponibles, use este parámetro.
- **PARALLEL integer**, indica el grado de paralelismo, integer es del número de hilos paralelos en la operación. Normalmente Oracle calcula el grado óptimo de paralelismo.

partitioning_caluses, se utiliza para indicar que el registro del snapshot es particionado en un rango específico de valores o mediante una función de hash. Particionar un registro de un snapshot es igual a hacerlo con un tabla.

WITH , se usa para indicar si el registro del snapshot debe guardarse con una clave primaria, un rowid o ambas. Esta también indica si el registro guardará columnas filtradas, las que no son clave primaria de la consulta contenida en el snapshot.

NEW VALUES, permite indicar si Oracle guardará tanto los antiguos valores como lo nuevos en el registro.

- **INCLUDING**, guarda tanto los antiguos como nuevos valores en el registro.
- **EXCLUDING**, no permite que se guarde los nuevos valores en el registro, este en por defecto el que se realiza si no se especifica.

Además de la orden CREATE SNAPSHOT LOG, también se tiene las siguientes sentencias, que nos permiten el trabajo con los registros para snapshot:

- ALTER SNAPSHOT LOG
- DROP SNAPSHOT LOG

La primera para modificar un registro, y la segunda para borrarla.

5.9.8. Propagación serial y paralela

Cuando se crean links fijos en un ambiente de replicación, cada link puede propagar asincrónicamente cambios a un destino usando propagación serial o paralela. Antes de configurar un ambiente de replicación, se debe decidir que propagación usar:

- Con la **Propagación Serial**, Oracle propaga transacciones replicadas de una en una y en el mismo orden que se actualizaron en el sistema origen. Para configurar un link fijo con propagación serial, el parámetro de paralelismo debe ser cero (0) en el procedimiento DBMS_DEFER_SYS.SCHEDULE_PUSH. O usando las herramientas de replicación, el control de procesos de propagación paralela debe ser cero (0) en el cuadro de diálogo Edit Push Schedule. Se usa propagación serial sólo cuando el destino es un sitio Oracle7.
- Con la **Propagación Paralela**, Oracle propaga transacciones replicadas usando múltiples streams paralelos para un rendimiento más alto. Cuando es necesario, Oracle ordena la ejecución de transacciones dependientes para conservar la integridad de los datos. Para configurar un link fijo con propagación paralela, el parámetro de paralelismo debe ser uno (1) o más en el procedimiento DBMS_DEFER_SYS.SCHEDULE_PUSH. O usando las herramientas de replicación, el control de procesos de propagación paralela debe ser uno (1) o más en el cuadro de diálogo Edit Push Schedule.

5.10. Topología de iconos e imágenes de oracle

 Indica un sitio master.

 Muestra el número actual de requerimientos administrativos que están siendo procesados en el sitio al cual esta asociado.

 Indica un sitio master con errores en transacciones diferidas y/o con errores de requerimientos administrativos.

 Muestra el número actual de transacciones diferidas que van a ser aplicadas en el sitio al cual esta asociado.

 Muestra el número de errores de las transacciones diferidas en el sitio al cual esta asociado.

 Indica un sitio que esta funcionando como un sitio master y como un sitio de vista materializada (es un sitio dual).

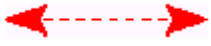
 Indica un sitio dual con errores en las transacciones diferidas y/o con errores en los requerimientos administrativos.

 Cuando esta conectado a un sitio master, indica que el sitio master tiene sitios de vistas materializadas. Las vistas materializadas y/o los grupos de vistas materializadas pueden ser registradas en el sitio master.

 Cuando esta conectado a un sitio de vista materializada, indica el sitio de la vista materializada.

 Indica un sitio de vista materializada con errores.

 Indica que un link de base de datos esta trabajando desde el sitio master actual al sitio master remoto.

 Indica un link de base de datos roto desde el sitio master actual al sitio master remoto.

 Cuando esta conectado a un sitio de vista materializada, indica un link de base de datos entre el sitio de vista materializada y un sitio master. Aquí, puede o no haber conexión a la red actual entre los dos sitios.

---------- Indica uno o más links de base de datos entre los sitios de vistas materializadas y un sitio master. Aquí, puede o no haber conexión a la red actual entre los sitios.

CAPITULO VI

PROTOTIPO DE BASE DE DATOS DISTRIBUIDA

6.1. Pasos para elaborar una base de datos distribuida y su replicación

Para el diseño de la base de datos distribuida y su replicación mediante snapshots se debe seguir los siguientes pasos:

- Diseñar la Base de Datos en una herramienta Case.
- Elaborar el Front End según las necesidades.
- Seleccionar las tablas a replicar
- Creación del enlace de conexión, snapshot, snapshot log, y sinónimos en los servidores por ejemplo:

  Servidor Sucursal01

  1. create snapshot log on clientess1 with primary key;

  Servidor Principal

  2. create database link bdsuc01

     connect to gerente identified by gerente

     using 'bdsuc01';

  3. create snapshot s_clientess1

     refresh fast start with sysdate next sysdate + 1/86400

```
with primary key as
select * from clientess1@bdsuc01;
```

4. create synonym clientess1 for s_clientess1;

6.2. Análisis del prototipo propuesto

Con el fin de lograr la máxima utilidad de la información del presente documento, se ha desarrollado un prototipo de facturación de una farmacia.

El prototipo será distribuido ya que en un nodo cumple las necesidades de ese nodo, pero también es capaz de recibir información de otros nodos y proveer de información a otros nodos dentro de la red.

La aplicación se ha desarrollado en un ambiente gráfico, bajo el Sistema Operativo Windows el mismo que gracias a su característica de red nos permitirá crear una aplicación cliente/servidor distribuida.

La Base de Datos de la aplicación se ha desarrollado en Oracle 9i, la misma que es un Servidor de Bases de Datos Relacional, para la interacción con la base de datos se utilizó Oracle Developer 6i, que es un desarrollador de aplicaciones que permiten construir rápida y productivamente sofisticados sistemas cliente/servidor y sistemas basados en web. Utiliza Oracle Forms, una herramienta no procedural para desarrollar y

mantener aplicaciones de bases de datos rápida y eficientemente, sin una programación real. Oracle Reports y Oracle Graphics herramientas de reportes y generación de gráficos de Oracle Enterprise.

El ambiente de replicación es un intercambio bidireccional. Donde el servidor Principal es master para el servidor Sucursal01 y el servidor Sucursal01 es master para el servidor Principal.

Se utilizo un tipo de replicación master para los servidores Principal y Sucursal01, ya que en este tipo de replicación un sitio master soporta solo un sitio de vista materializada.

Para la distribución de datos entre servidores se utilizo Distribución o Replicación Masiva, la misma que soporta la replicación de tablas enteras, o parte de las mismas. Se basa en las denominadas Materialized Views (*snapshots o instantáneas*), de forma que la información a replicar se especifica a través de una consulta SQL.

Para la propagación de los datos se realizó una replicación síncrona, llamada también replicación en tiempo real, ya que al realizar un cambio en los datos o ejecutar cualquier acción en todos los sitios participantes, se lo hace como una sola transacción.

6.2.1. Diagrama de flujo de datos

- El gráfico 1. muestra el diagrama de contexto.

Gráfico1. Diagrama de contexto

- El gráfico 2. muestra el diagrama de flujo de datos de niveles.

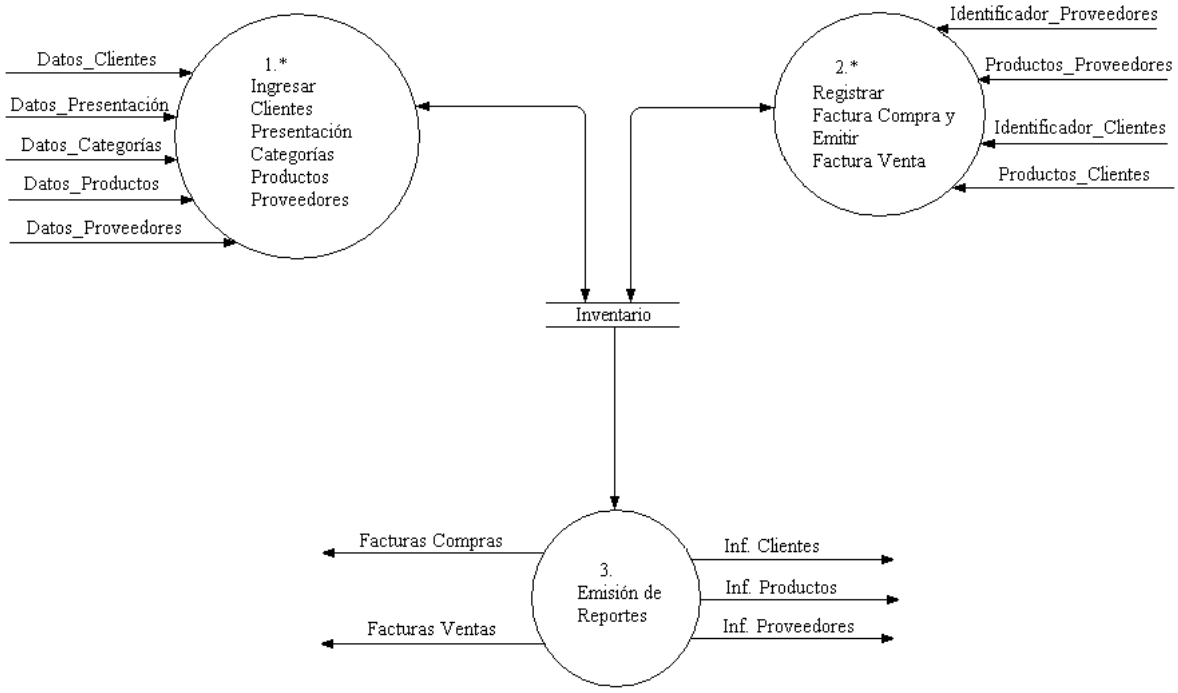


Gráfico 2. Diagrama de flujo de datos de niveles

- El gráfico 3. muestra el diagrama de flujo de datos de nivel 1.

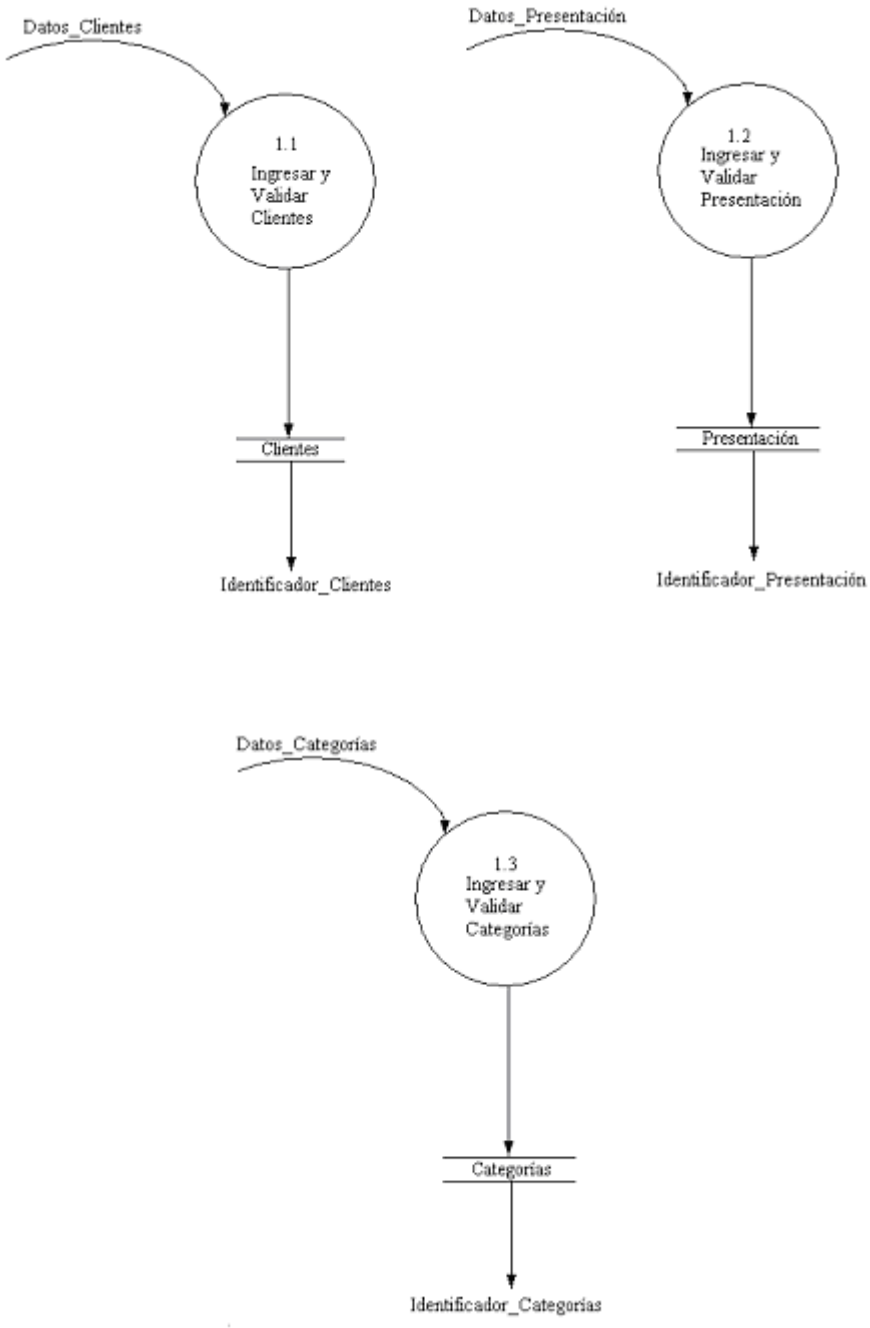


Gráfico 3. (cont)

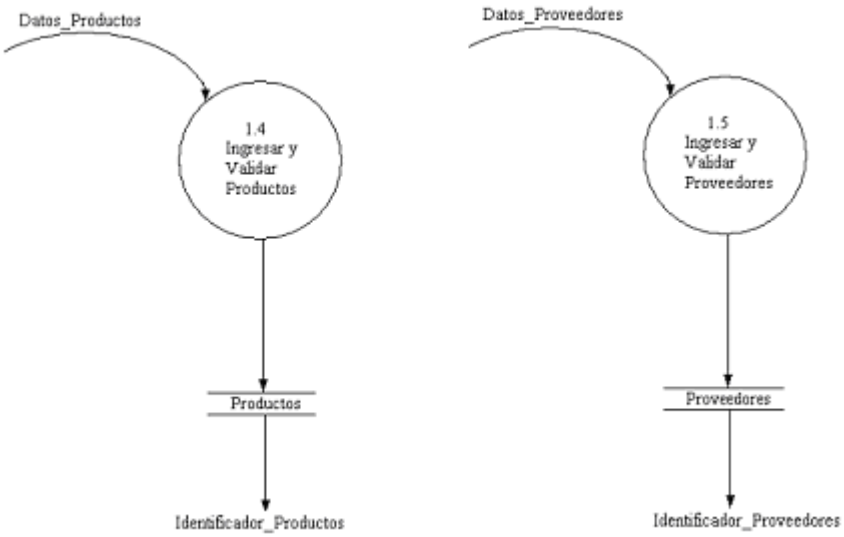


Gráfico 3. Diagrama de flujo de datos de nivel 1

- El gráfico 4. muestra el diagrama de flujo de datos de nivel 2.

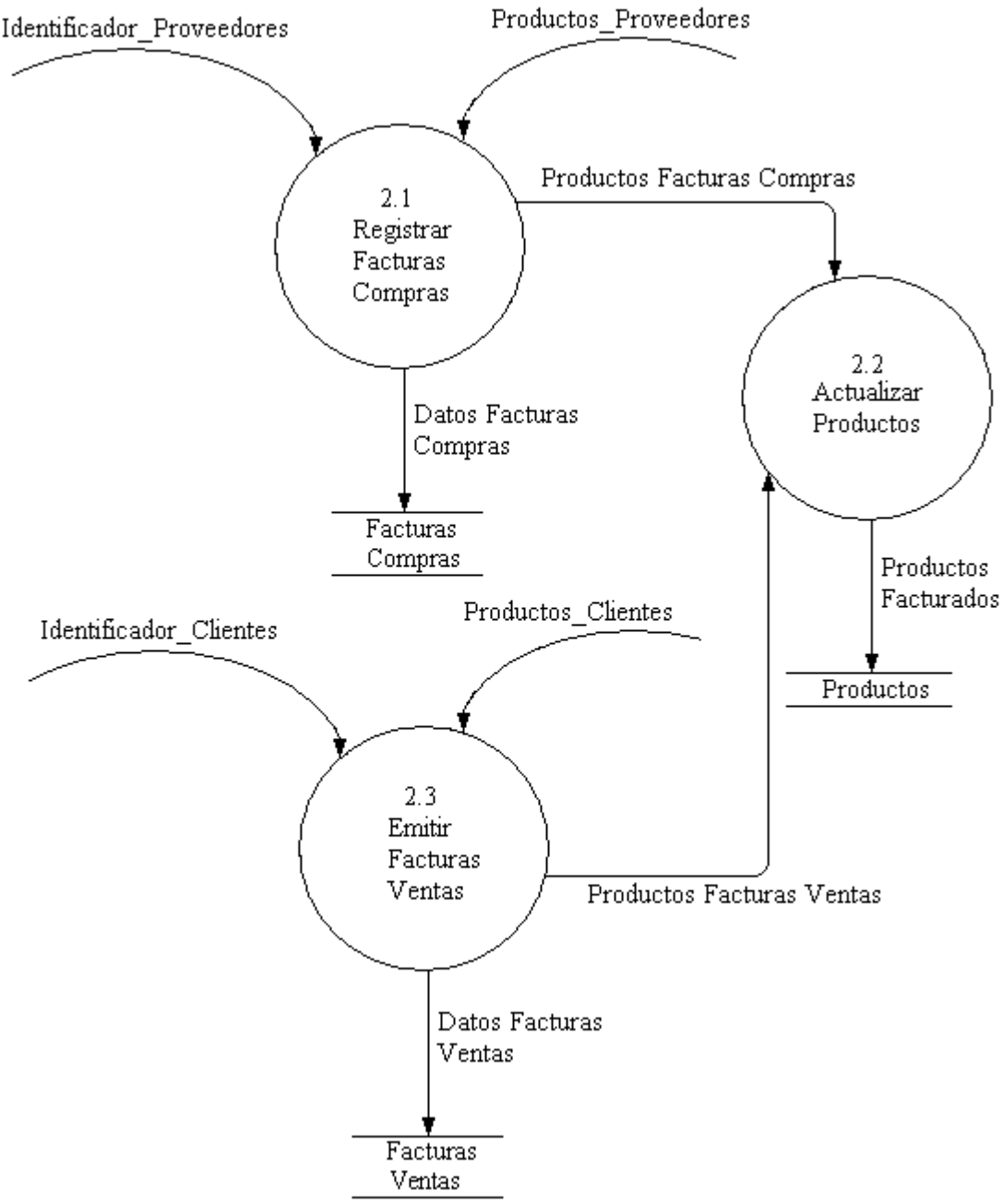


Gráfico 4. Diagrama de flujo de datos de nivel 2

6.2.2. Modelo entidad relación

MODELO LOGICO

- Modelo Entidad Relación – Modelo Lógico (Ver gráfico 5).

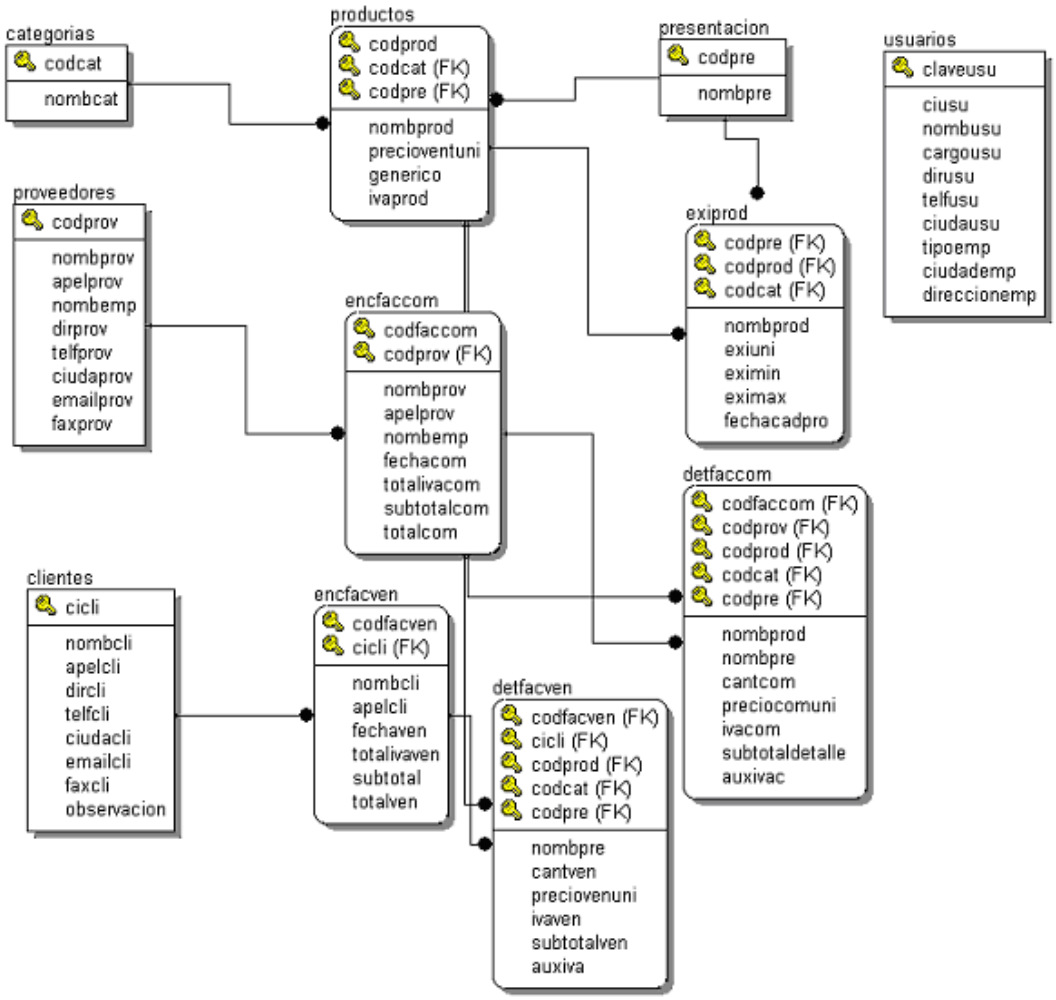


Gráfico 5. MER - Modelo lógico

MODELO FISICO

- Modelo Entidad Relación – Modelo Físico (Ver gráfico 6).

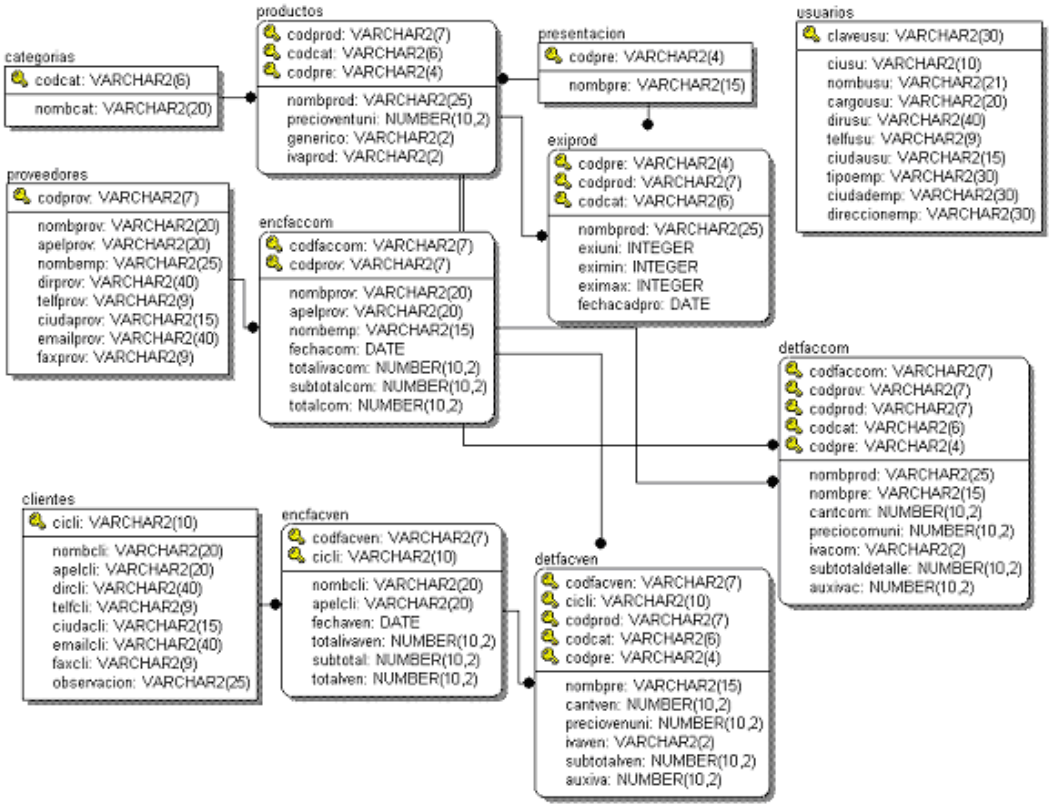


Gráfico 6. MER - Modelo físico

6.2.3. Diccionario de datos

El Diccionario de Datos constituye una referencia de datos relacionados a los datos recopilados durante el análisis y el diseño del prototipo. Para la elaboración de este documento se tomo como punto de partida los Diagramas de Flujos de Datos para recopilar los términos del Diccionario.

Las tablas desde la 29 a la 44 muestran la descripción de los flujos de datos:

Nombre: Datos_Clientes
Alias:
Descripción: cicli + nombcli + apelcli + dircli + telfcli + ciudacli + emailcli + Faxcli + observación

Tabla 29. Descripción del flujo de datos datos_clientes

Nombre: Datos_Proveedores
Alias:
Descripción: codprov + nombprov + apelprov + nombemp + dirprov + telfprov + ciudaprov + emailprov + Faxprov

Tabla 30. Descripción del flujo de datos datos_proveedores

Nombre: Datos_Productos
Alias:
Descripción: codprod + nombprod + codpre + codcat + precioventuni + generico + ivaprod

Tabla 31. Descripción del flujo de datos datos_productos

Nombre: Datos_Presentación
Alias:
Descripción: codpre + nombpre

Tabla 32. Descripción del flujo de datos datos_presentación

Nombre: Datos_Categorias
Alias:
Descripción: codcat + nombcat

Tabla 33. Descripción del flujo de datos datos_categorias

Nombre: Identificador_Proveedores
Alias:
Descripción: codprov + nombprov + apelprov

Tabla 34. Descripción del flujo de datos identificador_proveedores

Nombre: Identificador_Clientes
Alias:
Descripción: cicli + nombcli + apelcli

Tabla 35. Descripción del flujo de datos identificador_clientes

Nombre: Productos_Proveedores
Alias:
Descripción: codfaccom + codprod + nombprod + nombpre + cantcom + preciocomuni + ivacom + subtotalcom

Tabla 36. Descripción del flujo de datos productos_proveedores

Nombre: Productos_Clientes
Alias:
Descripción: codfacven + codprod + nombprod + nombpre + cantven + preciovenuni + ivaven +subtotalven

Tabla 37. Descripción del flujo de datos productos_clientes

FICHEROS

Nombre: Clientes
Alias:
Descripción: cicli + nombcli + apelcli + dircli + telfcli + ciudacli + emailcli + Faxcli + observación

Tabla 38. Descripción del fichero clientes

Nombre: Proveedores
Alias:
Descripción: codprov + nombprov + apelprov + nombemp + dirprov + telfprov + ciudaprov + emailprov + Faxprov

Tabla 39. Descripción del fichero proveedores

Nombre: Productos
Alias:
Descripción: codprod + nombprod + codpre + codcat + precioventuni + generico + ivaprod

Tabla 40. Descripción del fichero productos

Nombre: Presentacion
Alias:
Descripción: codpre + nombpre

Tabla 41. Descripción del fichero presentación

Nombre: Categorías
Alias:
Descripción: codcat + nombcat

Tabla 42. Descripción del fichero categorías

Nombre: Facturas_Compras
Alias:
Descripción: {Encabezado} codfaccom + codprov + nombprov + apelprov + nombemp + fechacom + formapagocom + descformagocom + totalivacom + totalcom num
{Detalle} codfaccom + codprod + nombprod + nombpre + cantcom + preciocomuni + ivacom + subtotalcom

Tabla 43. Descripción del fichero facturas_compras

Nombre: Facturas_Ventas
Alias:
Descripción: {Encabezado} codfacven + codprod + nombprod + nombpre + cantven + preciovenuni + ivaven + subtotalven
{Detalle} codfacven + codprod + nombprod + nombpre + cantven + preciovenuni + ivaven +subtotalven

Tabla 44. Descripción del fichero facturas_ventas

6.3. Diseño de entradas

La calidad de las entradas del prototipo determina la calidad de las salidas del mismo, es por esta razón que durante el diseño de las formas de entrada es necesario tener en mente esta relación. Las entradas han sido diseñadas para satisfacer los siguientes objetivos:

- Eficacia
- Precisión
- Facilidad de uso
- Consistencia
- Sencillez y atracción

- Uniformidad de colores
- Posibilidades de cancelar operaciones

La eficacia significa que las formas de entrada satisfacen propósitos específicos del prototipo, mientras que la precisión hace a un diseño asegurar una realización satisfactoria de determinada operación o proceso dentro de la aplicación. La facilidad de uso implica que las formas son explícitas y no requieren de tiempo adicional para descifrarse, es decir que basta con mirarlas para tener una idea de la tarea que realizan.

La consistencia se refiere a que las formas ordenan los datos de manera similar de una aplicación a otra, mientras que la sencillez mantiene en un mínimo los elementos indispensables que centran la atención del usuario. La uniformidad de colores es por utilización de los mismos colores para el diseño de las diferentes formas que utiliza el prototipo.

6.3.1. Diseño de la pantalla principal

La pantalla principal tiene un menú para acceder a las diferentes opciones del prototipo, además tiene un sistema de ayuda que indica lo que se realiza en cada opción para una mejor comprensión del usuario.

6.3.2. Diseño de las pantallas de captación de datos

Las ventanas de captación de datos han sido diseñadas de manera que tengan un flujo adecuado lo cual minimiza el tiempo invertido y el esfuerzo realizado por los usuarios del prototipo en el llenado de las mismas. Las formas deben seguir un flujo de izquierda a derecha, y de arriba hacia abajo.

La mayoría de las ventanas se caracterizan por su carga modal lo que significa que mientras esté activa una ventana hija no se puede accede a la ventana padre mientras no se cierre la anterior. Otra característica es que la mayoría de ventanas no tienen iconos de minimizar, maximizar, y cerrar.

El prototipo utiliza básicamente las siguientes ventanas:

- Para el ingreso de los codificadores clientes, proveedores, productos, presentación y categorías se tiene el siguiente estilo (Ver figuras 26 - 30).

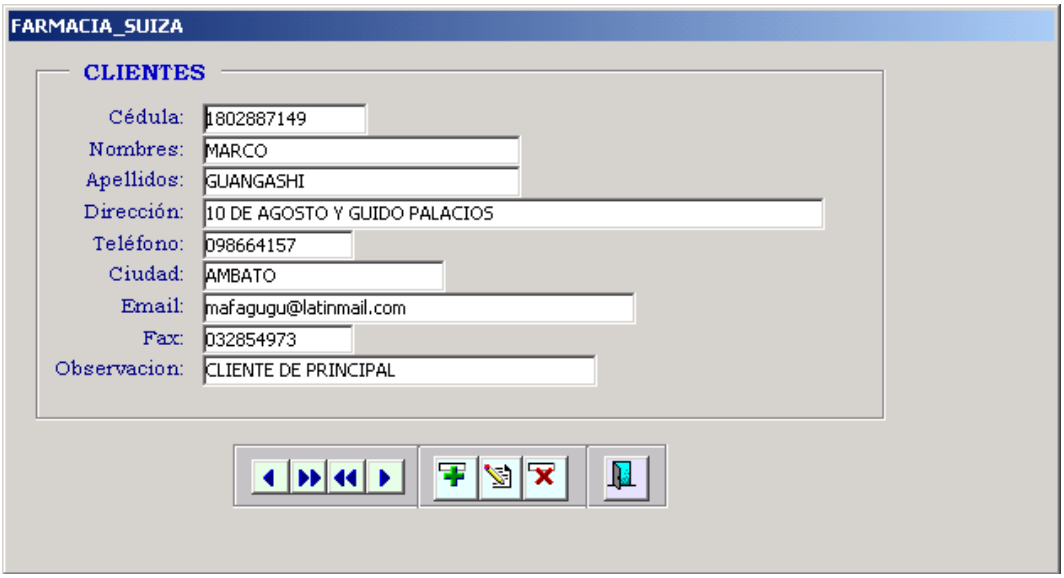


Figura 26. Codificador de clientes

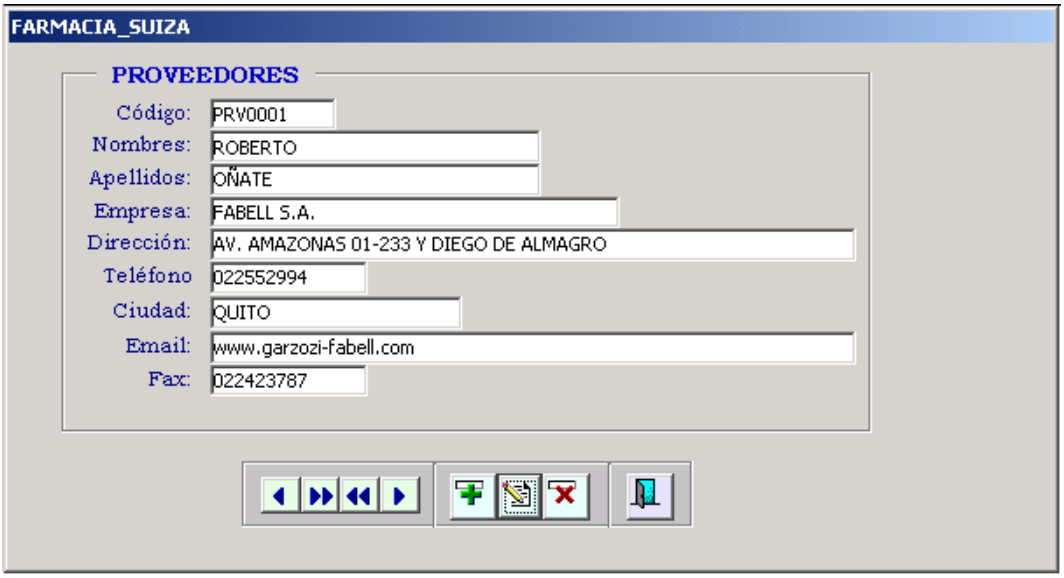


Figura 27. Codificador de proveedores

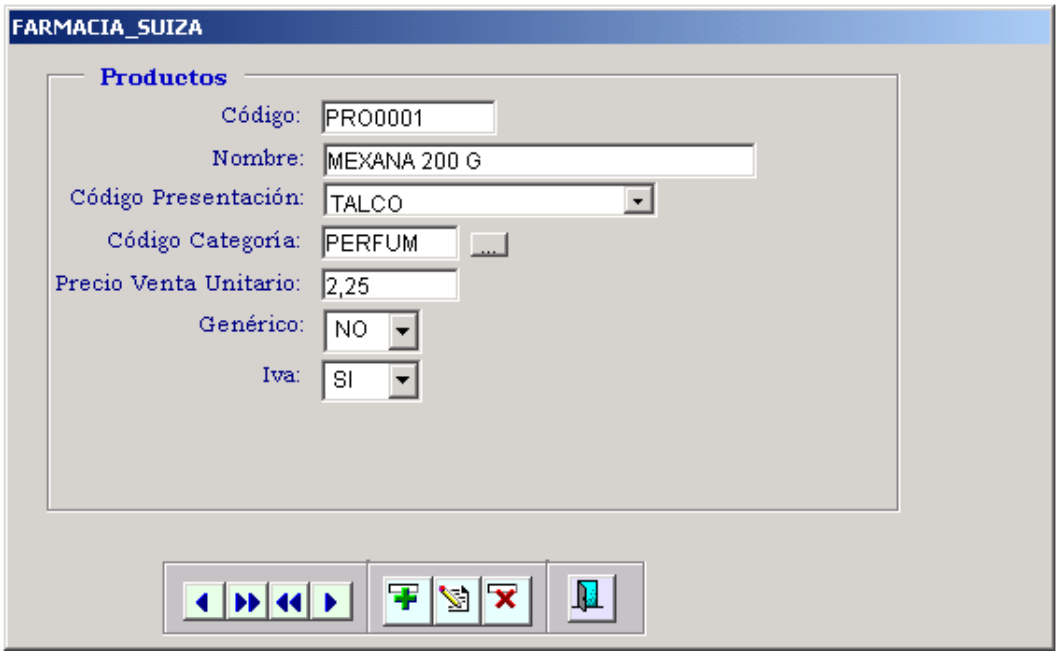


Figura 28. Codificador de productos

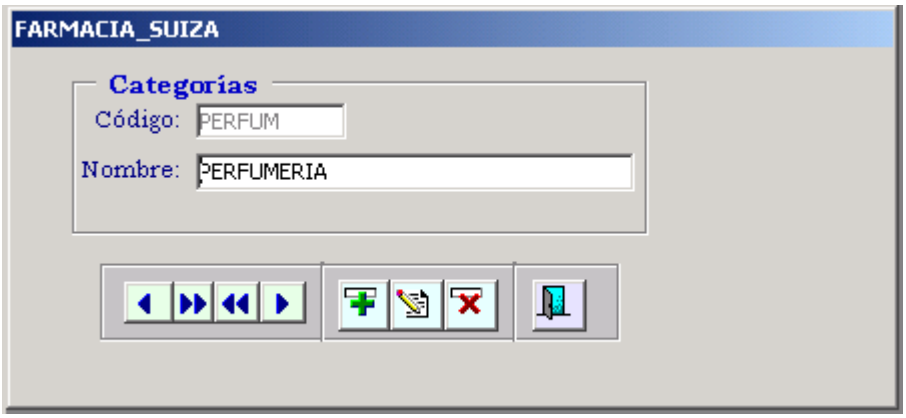


Figura 29. Codificador de categorías

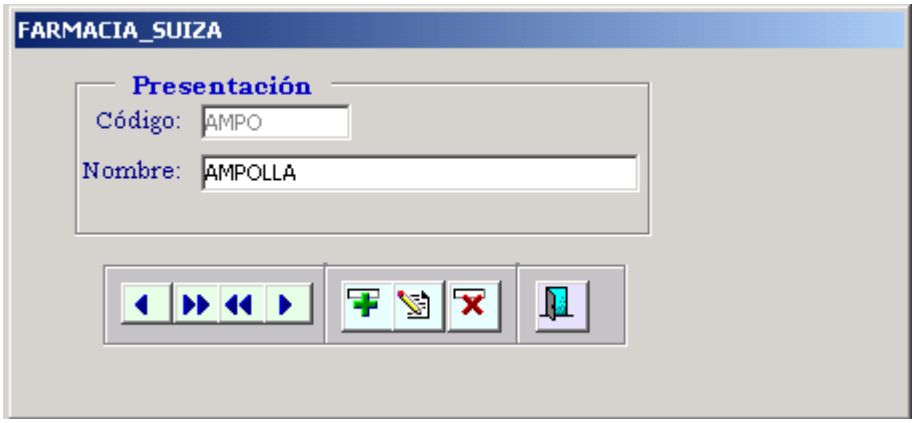


Figura 30. Codificador de presentación

- Para el ingreso del codificador usuarios se tiene el siguiente estilo (Ver figura 31).

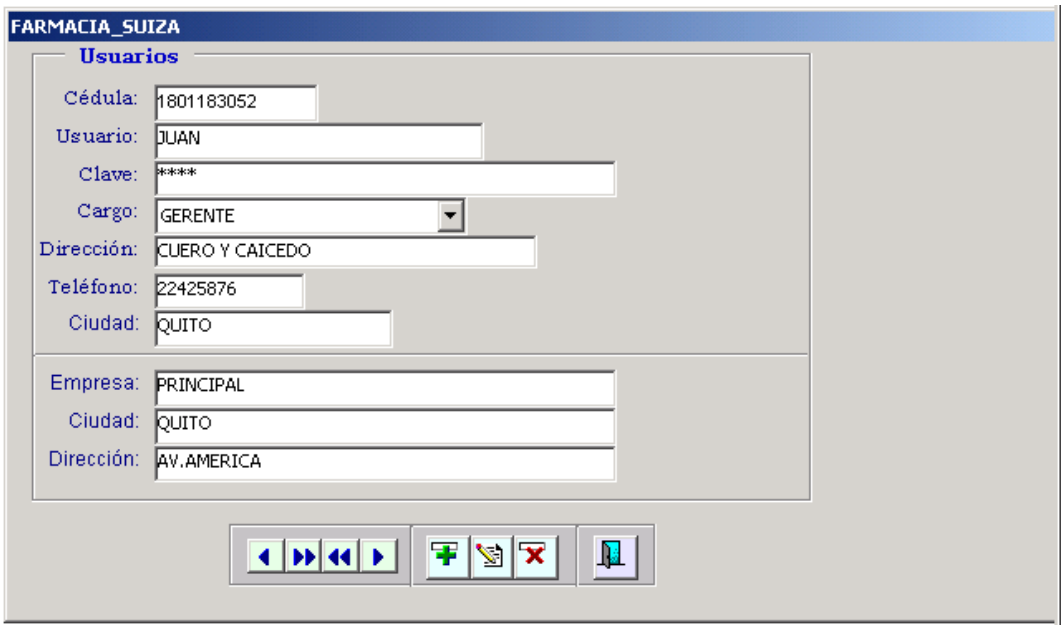


Figura 31. Codificador de usuario

- Para el ingresar al prototipo con privilegios de usuario se tiene el siguiente estilo (Ver figura 32).

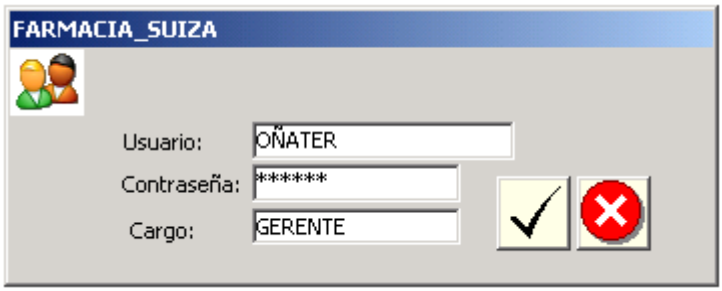


Figura 32. Parámetros de usuario

- Para las facturas de compras y ventas se tiene el siguiente estilo (Ver figuras 33 y 34).

Figura 33. Factura de compra

Figura 34. Factura de venta

- Para las búsquedas de clientes y proveedores se tiene el siguiente estilo (Ver figuras 35 y 36).

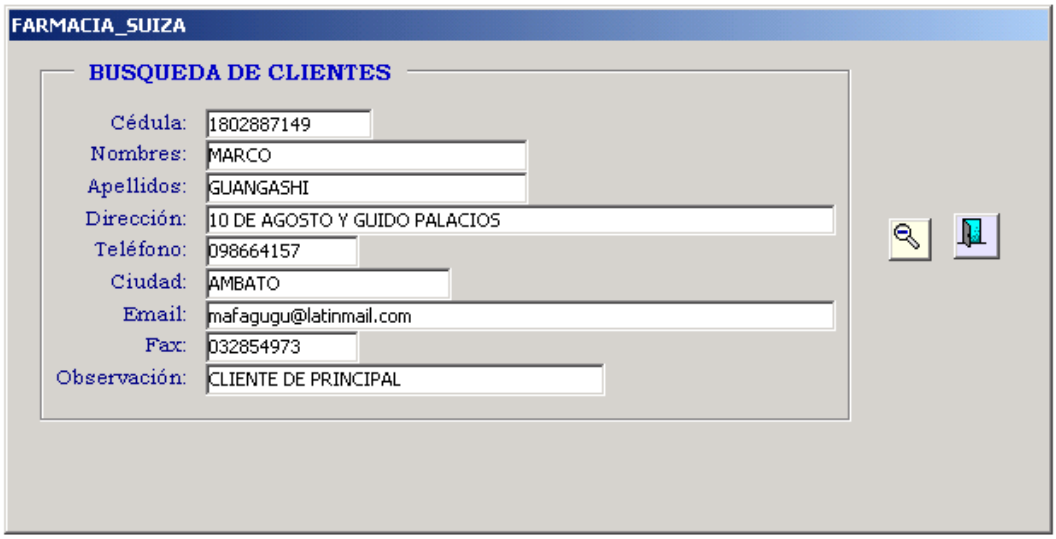


Figura 35. Búsqueda clientes

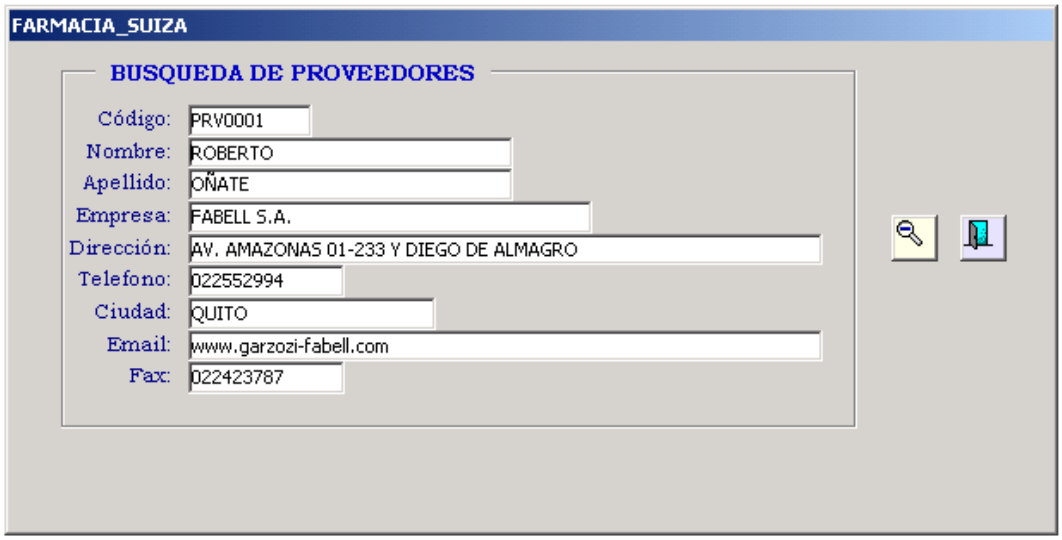


Figura 36. Búsqueda proveedores

6.3.3. Validación de la entrada de datos

Todas las entradas de datos se validan a través de la emisión de un mensaje, el mismo que se encuentra personalizado dependiendo del tipo de error que puede ocurrir durante la interacción del prototipo con el usuario. Este mensaje se muestra en una ventana con las características típicas que poseen las ventanas de Windows (Ver figura 37).

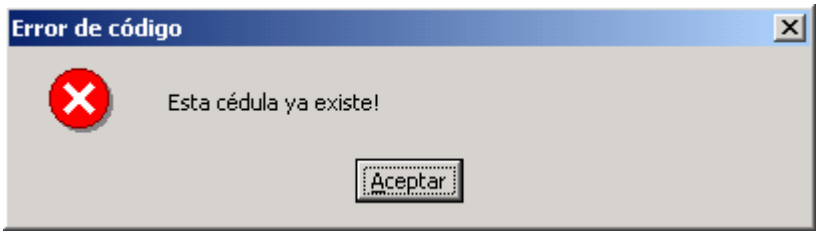


Figura 37. Mensaje de error del prototipo

Es importante capturar los datos con efectividad con el objetivo de asegurar la calidad de los datos que entran en el sistema, por ser la etapa en la cual se obtiene la mayor productividad de los recursos. Por esta razón, para garantizar que la información ingresada sea la adecuada se ha dado un tratamiento cuidadoso a los diferentes tipos de errores que sean detectado:

Errores a nivel de campo, en el cual se verifican longitudes, tipos y algunas otras restricciones que tienen ciertos datos como por ejemplo los tipo fecha se a colocado un botón calendario.

Errores en búsquedas, permite mantener la integridad de los campos claves los mismos que no pueden repetirse, esto es de gran utilidad cuando es el usuario el que ingresa el código correspondiente a este tipo de campo.

6.3.4. Diseño de las pantallas de mensajes

Las pantallas de mensajes se emiten con el fin de darle al usuario la posibilidad de confirmar o cancelar determinada operación, estos mensajes generalmente se visualizan cuando se ejecutan operaciones de edición o eliminación (Ver figura 38).

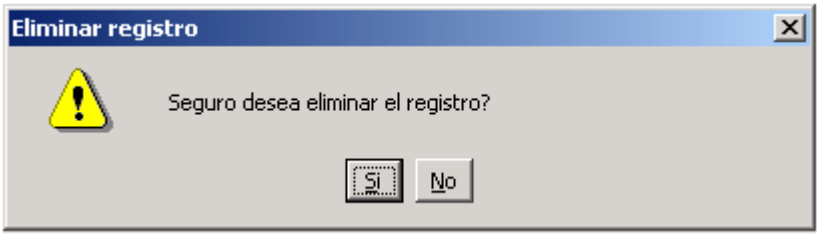


Figura 38. Mensajes emitidos por el prototipo

6.4. Diseño de salidas

Las salidas constituyen la información que reciben los usuarios del sistema, algunas de las cuales antes de convertirse en una salida adecuada, de acuerdo a los requerimientos del usuario, requieren de un proceso y otras que no lo requieren.

El prototipo dará al usuario la posibilidad de escoger el tipo de salida para un determinado reporte, es decir le permitirá elegir entre reportes impresos y salidas en

formatos tal como muestra la pantalla del monitor. Puesto que una buena salida es esencial para lograr la aceptación y uso del sistema, estas han sido diseñadas de tal manera que cumplen con los siguientes objetivos:

- Diseñar una salida que se adapte al usuario.
- Proveer la cantidad adecuada de información.
- Asegurar que la salida esté disponible donde se requiera.
- Proporcionar oportunamente la salida.
- Elegir el método de salida.

El prototipo cuenta con los siguientes reportes en forma de tablas (Ver tablas 45 - 54).

- Reporte de Proveedores.

FARMACIA " SUIZA "

PROVEEDORES

Código	Nombre	Apellido	Empresa	Dirección	Teléfono	Ciudad
PRV0001	BEATRIZ	RIVERA	FABELL S.A.	AV. AMAZONAS 01-233 Y DIEGO DE ALMAGRO	022552994	QUITO
PRV0021	ANITA	ARBOLEDA	BAYER	10 DE AGOSTO	228546558	QUITO

Tabla 45. Lista de proveedores

- Reporte de Productos.

FARMACIA " SUIZA "

PRODUCTOS

Código	Nombre	Categoria	Presentación	Precio	Genérico	IVA
PRO0001	MEXANA 200 G	PERFUMERIA	TALCO	2,25	NO	SI
PRO0002	MEXANA 100 G	PERFUMERIA	TALCO	1,3	NO	SI
PRO0003	FUNGIREX	PERFUMERIA	TALCO	1	NO	SI
PRO0004	VOLTAREN	ANTIINFLAMATORIO	LIQUIDO	3,5	SI	NO
PRO0005	ALKASELZER	ANTIACIDOS	SPRAY	3	SI	SI
PRO0006	VOLTAREN	ANTIINFLAMATORIO	SPRAY	2	SI	SI

Tabla 46. Lista de productos

- Reporte de Clientes.

FARMACIA " SUIZA "

CLIENTES

Cédula	Nombre	Apellido	Dirección	Telefono	Ciudad
1802887149	MARCO	GUANGASHI	10 DE AGOSTO Y GUIDO PALACIOS	098664157	AMBATO
1802648095	ROBERTO	OÑATE	BOLIVAR 01-233 Y ESPEJO	098552994	AMBATO
1802777951	JUAN	CONDO	MERA 01148 Y CUENCA	2845793	AMBATO
1800227595	SIMON	PEREZ	CEVALLOS Y ESPEJO	000000000	AMBATO
0000000000	Cliente	Cliente	XXXXXXXXXXXX	000000000	AMBATO
1801183052	NESTOR	LOPEZ	5 DE JUNIO 345 Y BOLIVAR	2429787	AMBATO
1709254831	MARIA	LASCANO	PEREZ DE ANDA 748 Y CASTILLO	099854677	AMBATO

Tabla 47. Lista de clientes

- Reporte de Existencia de Productos.

FARMACIA " SUIZA "

EXISTENCIA DE PRODUCTOS

Producto	Presentación	Existencia	Mínima	Máxima	Caducidad
MEXANA 200 G	TALCO	35	5	40	12/08/08
MEXANA 100 G	TALCO	25	5	45	01/09/05
FUNGIREX	TALCO	10	3	15	04/08/05

Tabla 48. Lista de existencia de productos

- Reporte de Productos en Caducidad.

FARMACIA " SUIZA "

PRODUCTOS EN CADUCIDAD

Código	Producto	Presentación	Fecha Caduca
PRO0001	MEXANA 200 G	TALCO	12/01/05
PRO0002	MEXANA 100 G	TALCO	01/01/05
PRO0003	FUNGIREX	TALCO	04/01/05

Tabla 49. Lista de productos en caducidad

- Reporte de Factura de Venta.

FARMACIA " SUIZA "

FACTURA DE VENTA

Cédula: 1802887149

Cliente: MARCO GUANGASHI

 # Factura: 41

Fecha: 20/02/05

Producto	Presentación	Cantidad	Precio U	IVA	Subtotal
MEXANA 200 G	TALCO	12	2,25	SI	27
				Subtotal:	**27**
				IVA:	**3,24**
				Total:	**30,24**

Tabla 50. Factura de venta

- Reporte de Factura de Compra.

FARMACIA " SUIZA "

FACTURA DE COMPRA

Factura: FAC0101

Proveedor: PRV0001 BEATRIZ RIVERA

Fecha: 20/02/05

Codigo	Nombre	Presentación	Precio	Cantidad	Iva	Subtotal
PRO0002	MEXANA 100 G	TALCO	1	10	SI	10
					Subtotal:	**10**
					iva:	**1,2**
					Total:	**11,2**

Tabla 51. Factura de compra

- Reportes Distribuidos de Sucursal 01.

FARMACIA " SUIZA "

Productos Sucursal 01

Código	Nombre	Precio	Genérico	Exi. Unid.	Exi.Mini.	Exi. Maxi
PRO0001	MEXANA 200 G	2,25	NO	30	5	40
PRO0003	FUNGIREX	1	NO	10	3	15

Tabla 52. Lista de productos sucursal 01

FARMACIA " SUIZA "

Clientes Sucursal 01

Cédula	Nombre	Apellido	Dirección	Teléfono	Ciudad	Observación
1802648095	ROBERTO	OÑATE	BOLIVAR 01-233 Y ESPEJO	098552994	AMBATO	CLIENTE DE SUCURSAL01
1801183052	JUAN	PEREZ	CUENCA Y MONTALVO	000000000	AMBATO	CLIENTE DE SUCURSAL01
1802115053	MAGDALENA	ISLAS	MIRAFLORES	032827463	AMBATO	CLIENTE DE SUCURSAL01

Tabla 53. Lista de clientes sucursal 01

FARMACIA " SUIZA "

Proveedores Sucursal 01

Código	Nombre	Apellido	Dirección	Teléfono
PRV0001	BEATRIZ	RIVERA	AV. AMAZONAS 01-233 Y DIEGO DE ALMAGRO	022552994
PRV0002	ROBERTO	OÑATE	AV. AMAZONAS 5-25	022546382

Tabla 54. Lista de proveedores sucursal 01

6.5. Seguridades y pruebas

Para mantener la integridad de los datos, la aplicación permite el ingreso de usuarios y claves correspondientes a cada uno de ellos, los mismos que son ingresados y validados para asignarles los permisos respectivos, con el objetivo de que determinada información no este disponible para usuarios no autorizados o personas ajenas a la empresa.

Durante el proceso de pruebas se llevo a cabo una evaluación total de todos los elementos del prototipo, lo que permitió identificar todos o por lo menos la mayoría de problemas que pudieron haber sido pasados por alto por parte de los desarrolladores, gracias a lo cual se puede garantizar el prototipo.

Las pruebas del prototipo se hicieron para verificar la manera en que trabaja el mismo, se desarrollaron evaluaciones con datos de prueba válidos y no válidos, los mismos que posteriormente se usaron para ver si las rutinas trabajan como se esperaba y también para la generación de errores.

Se hizo también una prueba del prototipo completo, es decir que se actuó como usuarios y operadores del sistema para lo cual se utilizaron varios datos de prueba.

CAPITULO VII

CONCLUSIONES Y RECOMENDACIONES

7.1. Conclusiones

Una vez concluido el trabajo investigativo para el Diseño de Bases de Datos Distribuidas mediante la arquitectura de Replicación Oracle se ha concluido que:

- Oracle tiene dos tipos de ambientes de replicación: Replicación Master y Replicación Multimaster (peer to peer o n ways). Soporta dos configuraciones básicas de replicación: Distribución Masiva y Replicación Servidor a Servidor, sobre tres mecanismos de propagación de cambios: sincrónica, asincrónica y procedural.
- Una estrategia para replicar bases de datos es determinar los datos a ser replicados y la forma que éstos van a ser usados en cada una de las replicas.
- En la aplicación se realizó una replicación bidireccional en tablas diferentes para solucionar el conflicto de unicidad ya que pueden haber dos transacciones con la misma clave primaria, generando de esta manera un error de DML. La solución para este caso es replicar en tablas separadas y generar sinónimos para las tablas replicadas.

- En la aplicación los servidores principal y sucursal01 son sitios master entre sí, por esta razón se utilizó vistas materializadas de solo lectura, ya que estas permiten actualizar el sitio master, dejando el sitio de vista materializada como solo lectura.
- Los sistemas de bases de datos distribuidas están formados por sitios de bases de datos independientes y la caída de uno de estos no afecta a todo el sistema como es el caso de los sistemas centralizados.
- La replicación proporciona acceso y procesamiento rápido a los datos debido a que la información se encuentra distribuida y cada nodo que interviene en la transacción ejecuta una carga de trabajo determinada, eliminando la sobrecarga en la red.

7.2. Recomendaciones

- Se recomienda considerar la replicación como una estrategia para disminuir el tráfico de la red y garantizar la tolerancia a fallas de los sistemas de bases de datos.
- Antes de implementar la replicación se recomienda establecer que tablas se van a replicar y sobre que usuarios, así como la disponibilidad de los recursos informáticos, sobre todo en lo que se refiere al sistema de comunicación que se va a emplear.
- Se recomienda usar Distribución Masiva para replicar tablas enteras, o parte de las mismas, pero mediante esta técnica no se puede replicar una base de datos entera.
- En la replicación avanzada los parámetros de inicialización se configuran por defecto, sin embargo, se recomienda verificar si estos parámetros satisfacen los requerimientos para la replicación.

GLOSARIO DE TERMINOS

La siguiente es una lista de los términos que se han utilizado a lo largo de los diferentes capítulos. Las definiciones ayudarán a comprender con mayor claridad algunos conceptos.

Administrador de Base de Datos

El administrador o DBA es el principal responsable de la operación, configuración y rendimiento de una base de datos. Su principal tarea consiste en resguardar la integridad de los datos almacenados en la base, proveyendo para esto mecanismos de respaldo, efectuando monitorizaciones periódicas al sistema, implementando medidas de seguridad, etc.

API

Application Program Interface. Juego de rutinas, proporcionadas en librerías que extienden la funcionalidad de un lenguaje.

ARCHIVELOG

Modo de base de datos que contiene archivos Redo Log. Este modo garantiza la completa recuperabilidad de la base de datos.

Atributo

Cada atributo de una relación almacena información de una parte de un objeto. Los atributos se representan como columnas en una tabla. Cada atributo en una relación es único y contiene valores atómicos. El número de atributos en una relación es llamado grado de la relación.

Base de datos Distribuida

Base de datos cuyos objetos (tablas, vistas, columnas y/o archivos) residen en más de un sistema en una red, y se puede acceder o actualizar desde cualquier nodo en la red.

Bloque

Un bloque es la unidad más pequeña de almacenamiento en una base de datos Oracle. El tamaño mínimo es de 2 KB y el máximo no debiera superar los 16 KB.

Buffer

Este término se refiere a una cantidad de memoria utilizada para almacenar información. Un buffer comúnmente almacena datos que están a punto de ser usados o se acaban de utilizar recientemente. En la mayoría de los casos son copias exactas de datos que se encuentran almacenados en el disco y se mantienen en memoria con el fin de lograr un acceso más rápido y ayudar de esa manera a mejorar el rendimiento de un sistema.

Caché

Es un área de almacenamiento implementada en la memoria RAM del computador que permite accesos más rápidos a la información ya que es mucho más veloz que la memoria.

Canvas

Es la superficie en Oracle Forms, en la cual los items y prompts son dibujados. Los canvas se muestran en una ventana.

Catálogo (Catalog)

Ver Diccionario de datos

Checkpoint

Un checkpoint es una operación que fuerza a que todos los cambios registrados en bloques de datos en memoria, sean escritos en el disco.

Clean buffer

Un buffer de este tipo es aquel que no ha sido modificado y que por lo tanto el proceso DBWR (lectura-escritura de la base de datos) no utilizará para confirmar los cambios en el disco (porque no ha sufrido cambios).

Commit de dos fases

Estrategia en la que se aplican cambios a una base de datos temporalmente. Una vez que todos los cambios se han hecho con éxito, los cambios se anuncian permanentemente a la base de datos.

Concurrencia

Este término se refiere a la capacidad de permitir muchas funciones al mismo tiempo. Oracle provee a muchos usuarios el acceso simultáneo a sus servicios, implementando de esta forma la concurrencia.

DBA

Ver Administrador de Base de datos

DBMS

El database management system o DBMS corresponde al software y grupo de herramientas que permiten manejar la base de datos. Un RDBMS es un DBMS relacional, es decir, cuya naturaleza es la formación de relaciones al interior del mismo.

DDL (comandos DDL)

Los comandos DDL (data definition language) son utilizados en la creación y modificación de objetos del esquema. Proveen la habilidad de crear, alterar e incluso eliminar objetos de un esquema, otorgar y revocar privilegios y roles a los usuarios,

establecer opciones de auditoria e incluso agregar comentarios al diccionario de datos del sistema. Estos comandos están estrechamente relacionados con las labores de administración de la base de datos.

Diccionario de Datos

Repositorio de metadata (información de datos). El diccionario de datos es un grupo de tablas de Oracle que se utilizan para almacenar información sobre el resto de las tablas, índices, clusters y otros objetos de la base de datos.

DML (comandos DML)

Los comandos DML (data manipulation language) son menos poderosos que los comandos DDL en cuanto a administración se refiere, de hecho, implementan modificaciones sobre la información que se guarda en los objetos de una base de datos. Estas sentencias son del tipo DELETE, INSERT, SELECT y UPDATE, principalmente.

Equi Join

Un Equi Join (Inner Join o Simple Join) es una declaración que usa una operación de equivalencia (colA=colB) para emparejar filas de diferentes tablas.

Esquema

Un esquema es una colección de objetos asociados dentro de una base de datos.

Función

Una función es un grupo de sentencias SQL, escritas generalmente en PL/SQL que implementan una serie de rutinas que devuelven un valor. Son casi idénticas a los procedimientos y sólo se diferencian en esa última condición. Implementando funciones en el servidor de base de datos se reduce el tráfico de comunicaciones en la red, ya que sólo se envían a la función los parámetros de entrada y ésta sólo devuelve el valor al final de todo el proceso, el que es ejecutado en la misma máquina donde reside la base de datos mejorando así el rendimiento general del sistema.

Memoria Virtual

Indica la memoria que puede ser utilizada por programas que corren en un sistema operativo y que está implementada físicamente en sectores del disco y no en la RAM. El proceso de copiar datos de la RAM al disco (o memoria virtual) se llama paginación (paging, en inglés). El archivo resultante es llamado el “swap file” y cada vez que un programa accede a esta memoria virtual disminuye el rendimiento del mismo debido a que realmente está accediendo al disco y no a la RAM.

Oracle Developer

Es una solución de desarrollo de aplicaciones que permiten construir rápida y productivamente sofisticados sistemas cliente/servidor y sistemas basados en web.

Oracle Forms

Es una herramienta no procedural para desarrollar y mantener aplicaciones de bases de datos rápida y eficientemente, sin una programación real.

Oracle Reports

Herramienta de reportes de Oracle Enterprise, puede manejar diferentes formatos de archivos (HTML, PDF, XML, RTF, etc) y métodos de distribución (E-mail, Web, Archivos, etc).

PL/SQL

Es una extensión del lenguaje procedural SQL de Oracle, tiene sintaxis, estructura y tipos de datos definidos. El lenguaje incluye técnicas de programación orientada a objetos tales como encapsulación, función de sobrecarga, ocultación de información (menos herencia), y trae una programación innovadora para el servidor de base de datos Oracle con una variedad de herramientas.

Procedimiento

Un Procedimiento almacenado es un grupo de sentencias SQL o PL/SQL que implementan un programa que se ejecuta en el servidor de base de datos, pero que a diferencia de las funciones, no devuelve un valor. Al igual que las funciones su implementación permite reducir el tráfico en la red, potenciando el rendimiento del sistema.

Pseudo-columna

Es una columna de base de datos Oracle que no se guarda en el disco. Los ejemplos de pseudo columna son: SYSDATE, SYSTIMESTAMP, ROWID, ROWNUM, LEVEL, CURRVAL, NEXTVAL, etc.

Query (consulta)

Es una consulta efectuada contra la base de datos en lenguaje SQL. Se genera utilizando la sentencia SELECT. Su principal característica es que no efectúa cambios en la base de datos; por este motivo es llamada también una transacción de sólo lectura.

Queue (Cola)

Primero en entrar primero en salir es la estructura usada para procesar múltiples demandas de los datos para un recurso. Los objetos se añaden a la cola y se procesan desde la cabeza.

Quiesce

Temporalmente inactivo o deshabilitado, una base de datos puede ser detenida con el comando ALTER SYSTEM QUIESCE RESTRICTED y se puede activar con el comando ALTER SYSTEM UNQUIESCE.

RECO (Oracle RECOverer Process)

Es un proceso de recuperación, creado cuando empieza una instancia con DISTRIBUTED_TRANSACTIONS = en el archivo del parámetro de inicialización. Este proceso intenta resolver transacciones in-doubt (en duda) en bases de datos distribuidas.

Redo Log

Juego de archivos que graban todos los cambios hechos en una base de datos Oracle. Un base de datos debe tener por lo menos dos archivos redo log. Se pueden hacer copias de los archivos log para asegurar que la información no se pierda.

Replicación

Se puede definir como una copia similar de los datos en la misma o en diferente plataforma. El proceso de reproducir los archivos de los datos a una o más bases de datos en tiempo cercano a la reliadad, es conocido como replicación. Los datos pueden presentarse en diferentes formatos.

Savepoint

Punto de referencia en el cual se puede hacer después un commit o un rollback

SCN (System Change Number)

Sistema de cambio de número, un número interno que Oracle incrementa con el tiempo como vectores de cambio. Se aplican y se guardan en el archivo redo log.

Scott

Es un usuario de la base de datos usado para propósitos de demostración que contienen las famosas tablas EMP y DEPT. Se crea el usuario scott/tiger ejecutando el script /rdbms/admin/utlsampl.sql.

Sistema Distribuido

Se refiere a los sistemas de computación en múltiples ubicaciones a lo largo de una organización, el sistema en una ubicación cumple las necesidades de esa ubicación, pero también es capaz de recibir información de otros sistemas y proveer de información a otros sistemas dentro de la red.

Snapshot

La copia de una tabla en un sistema remoto. Ver Vistas Materializadas.

SQL*Plus

Es una utilidad de la línea de comandos de Oracle usada para ejecutar comandos SQL y PL/SQL.

System Global Area (SGA)

El SGA es un área compartida de memoria que utiliza Oracle para guardar información de control en una instancia. Se asigna un espacio a esta área cuando la instancia se levanta (startup) y se elimina cuando ésta se baja (shutdown). Cada instancia de Oracle maneja su propia SGA y guarda información de los buffers y la shared pool.

Tabla Externa

Una tabla que no se guarda en una base de datos Oracle. Los datos se cargan por medio de un manejador de acceso (normalmente ORACLE_LOADER) cuando se accede a la tabla. Una tabla externa es como una vista que permite hacer consultas SQL.

Tablas de rendimiento dinámicas

Estas tablas son creadas cuando se levanta una instancia y se usan para guardar información acerca del rendimiento de ésta.. Esta información incluye notas acerca de la conexión, datos que manejan los procesos de entrada/salida, valores de los parámetros de inicialización , entre otros.

Timestamp

Una extensión del tipo de dato DATE que puede almacenar fechas y horas (incluyendo fracciones de segundo). Los timestamp ocupan 11 bytes.

Transacción

Una transacción es una unidad lógica de trabajo que consiste de una o más sentencias SQL, que pueden finalizar con un commit o un rollback. Las métricas de rendimiento utilizan comúnmente las unidades “transacciones por segundo” o “transacciones por minuto”.

Trigger

Un trigger es un mecanismo que permite escribir procedimientos que son ejecutados en forma automática (sin una orden explícita del usuario o programador) cuando ocurre un evento de INSERT, UPDATE o DELTE sobre una tabla o vista. Generalmente se utilizan los triggers para forzar las restricciones de integridad entre las tablas o automatizar alguna otra función específica.

Vista Materializada (Materialized View)

Una vista materializada es similar a una vista pero los datos realmente se guardan en el disco. Las Vistas Materializadas se usan a menudo para sumarios, y pre joins de tablas, o simplemente para hacer una instantánea (snapshots) de una tabla disponible en un sistema remoto. Una vista materializada puede hacer un refresh cuando los datos en las tablas subyacentes han cambiado.

Wrapper

Un objeto, función o procedimiento que encapsula y delega llamadas a otros objetos para alterar su interface o conducta de alguna manera.

BIBLIOGRAFÍA

KEN, Chu - Oracle Developer 2000 Advanced Techniques Manual, Madrid, editorial Oracle Corporation, año 1994, 2da Edición.

PÉREZ, César - Oracle 9i Administración y Análisis de Bases de Datos, Madrid, editorial Omega, año 2003, 1era edición.

PRESSMAN, Rogger S. - Ingeniería del Software un Enfoque Práctico, Madrid, editorial Mc Graw Hill, año 1998.

SENN, James A. - Análisis y Diseño de Sistemas de Información, México, editorial Mc Graw Hill, año 1977.

Direcciones WEB

- Oracle9i Advanced Replication.

http://www.cise.ufl.edu/help/database/oracle_docs/server.920/a96567/toc.htm

- Oracle 9i Database Replication.

http://www.oracle.com/ultrasearch/wwws/searchoc.jsp?p_Action=Search&p_Group=1&p_Group=4&p_Query=Replicationry=Replication

- Distributed Database Management.

http://www.cise.ufl.edu/help/database/oracle_docs/server.920/a96521/part6.htm#435958

- Fundamentos de Base de datos Distribuidas

http://www.geocities.com/ddbqp

- Diseño de Sistema de Bases de Datos

http://www.ucab.edu.ve/ingenieria/informatica/bd2/programa.pdf

- Introducción a las Bases de Datos Distribuidas

http://mailweb.udlap.mx/~tesis/msp/alvarez_c_g

- Sistema de Base de Datos Distribuida

http://www.cs.cinvestav.mx/SC/prof_personal/adiaz/Disdb/temario.html

- Distributed Database

http://www.cise.ufl.edu/help/database/oracle-docs/server.920/a96521/ds_concepts.htm

- Bases de Datos Distribuidas

http://sacbeob.8m.com/tutoriales/bddistribuidas/cap1.html

- Oracle Glossary of Terms

http://www.orafaq.com/glossary/

- Oracle Glossary

http://www.rdtex.ru/docs/glossary/G.html

A N E X O S

ANEXO I

a. INSTALACION Y CONFIGURACION DE SERVIDORES

El Sistema Operativo instalado en los servidores es Windows 2000 Advanced Server en español, el primer servidor se llama **Principal** con dirección IP 100.100.100.100 y el segundo servidor se llama **Sucursal01** con dirección IP 100.100.100.102**,** estos servidores cuentan con Service Pack 4.

La figura 39. muestra los servidores principal y sucursal01.

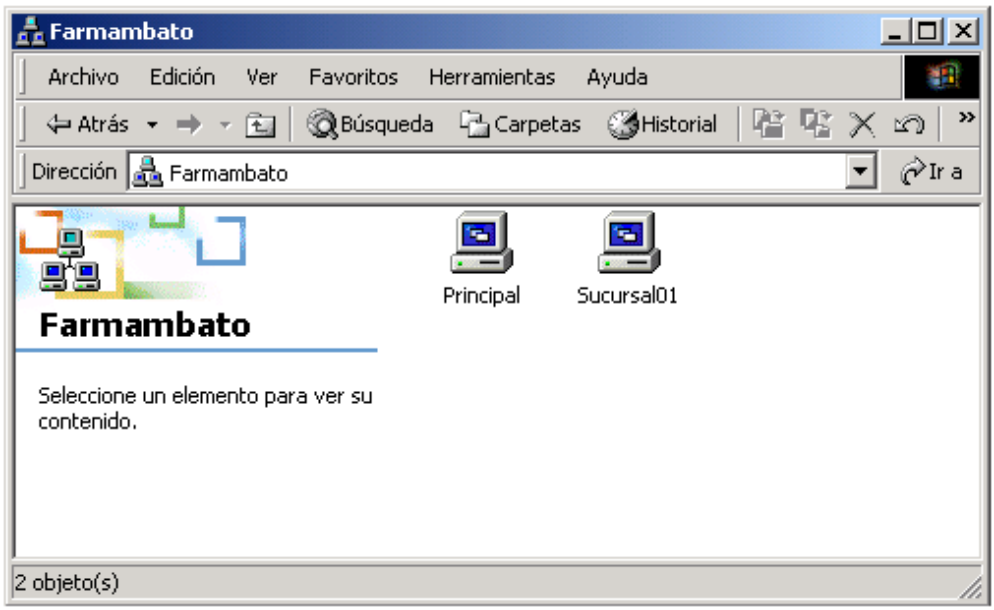


Figura 39. Servidores principal y sucursal01

Para la configuración de los servidores se deben seguir los siguientes pasos en orden:

1. Configuración de Active Directory en el Servidor Principal

(Ver figuras 40 - 43).

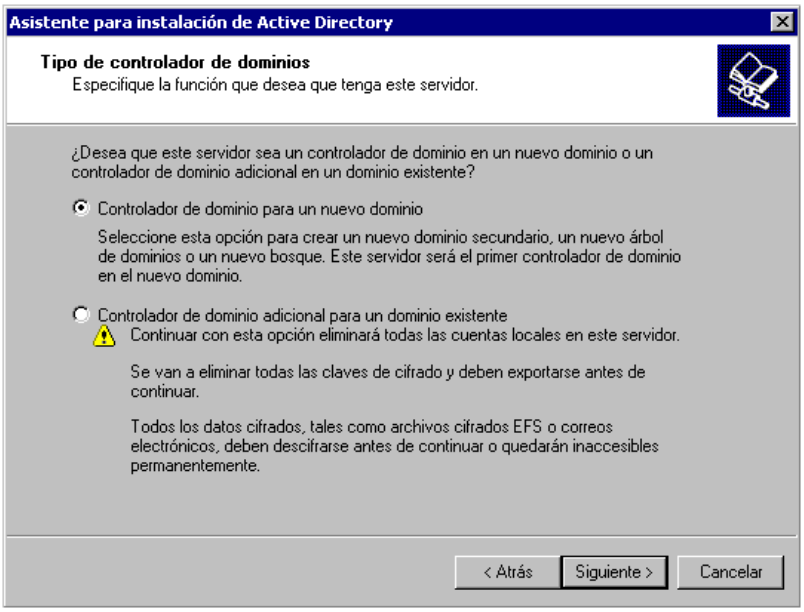


Figura 40. Asistente para instalación de active directory (servidor principal)

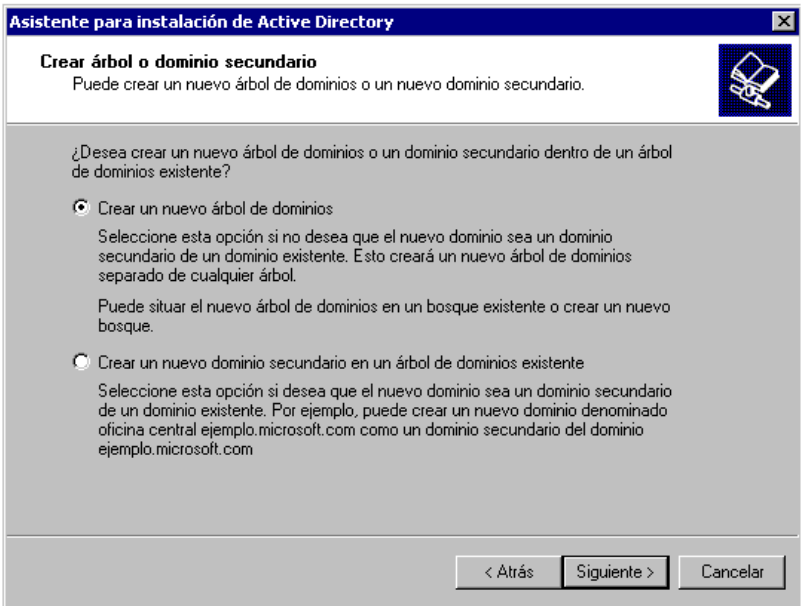


Figura 41. Asistente para instalación de active directory (servidor principal)

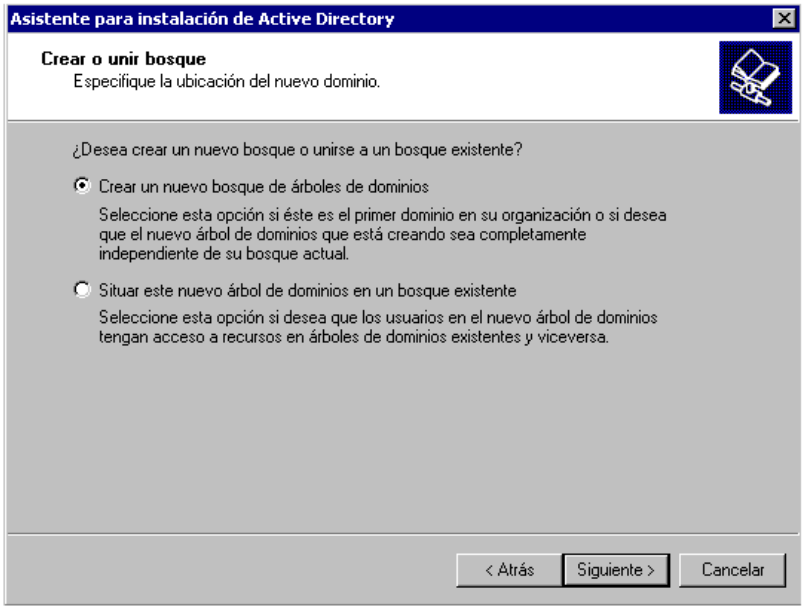


Figura 42. Asistente para instalación de active directory (servidor principal)

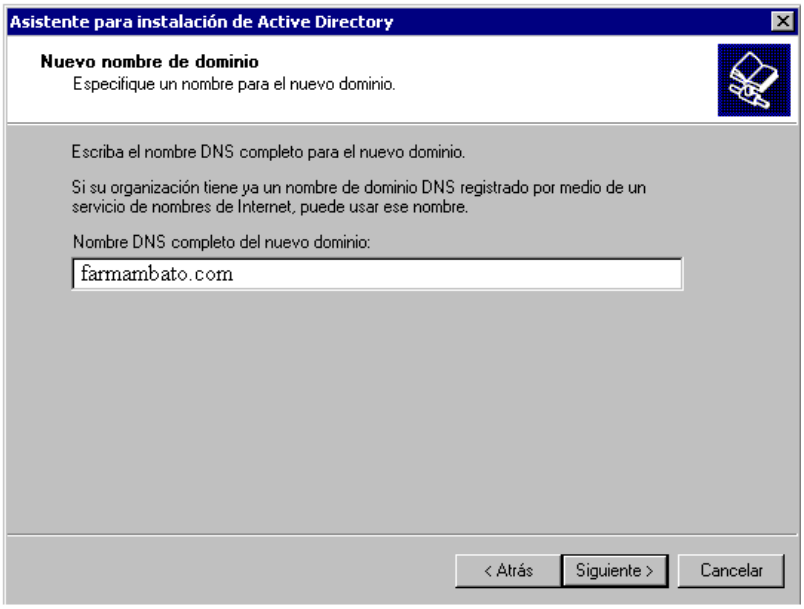


Figura 43. Asistente para instalación de active directory (servidor principal)

Luego pedirá el Nombre del NetBios del Dominio que será también **farmambato**, clic en Siguiente, y Aceptar para configurar el DNS.

Por último se escoge la opción, Permisos compatibles solo con servidores Windows 2000.

2. Configuración del Servidor de Nombres de Dominio (DNS) en el Servidor Principal

- Ir a Inicio, Programas, Herramientas Administrativas, escoger DNS.
- Ir a la carpeta Zona de Búsqueda Directa, ahí se encontrara el dominio que se creo con Active Directory (**farmambato.com**).

- Clic derecho en farmambato.com, escoger Host Nuevo, pedirá el Nombre que será **principal,** la IP que será 100.100.100.100, clic en Agregar Host, en Aceptar y en Realizado.
- Clic derecho en farmambato.com, clic en Alias Nuevo, pedirá el Nombre de alias que será **www**, clic en Examinar, en Principal, en Zona de Búsqueda Directa, en farmambato.com, en principal, en Aceptar y en Aceptar.

Quedando de la siguiente forma (Ver figura 44):

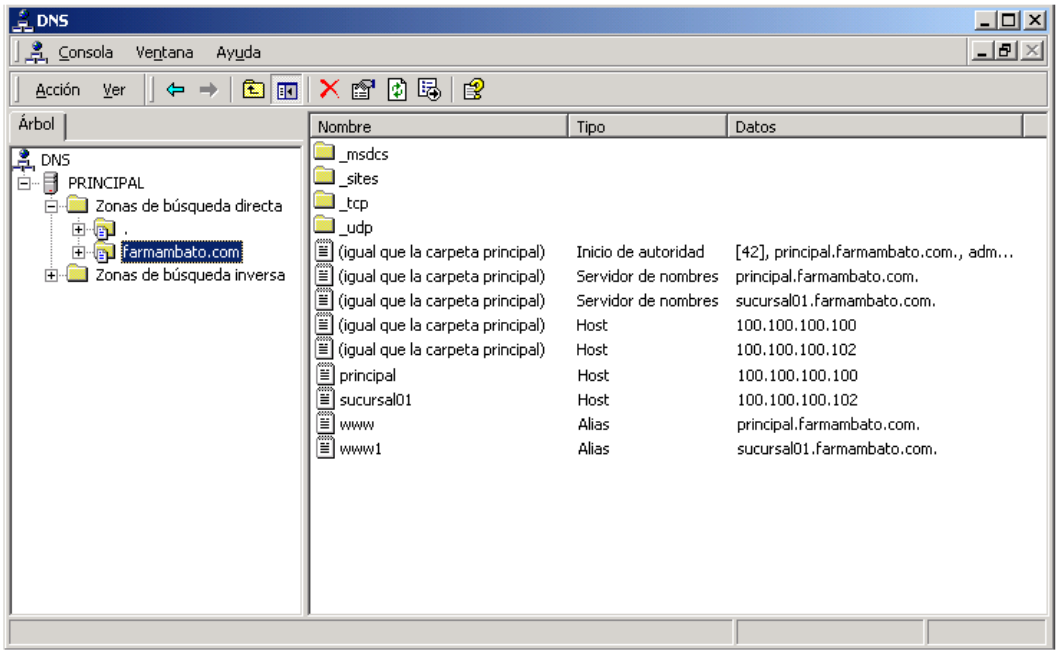


Figura 44. Configuración del servidor de nombres de dominio (servidor principal)

3. Configuración de Active Directory en el Servidor Sucursal01

(Ver figuras 45 y 46).

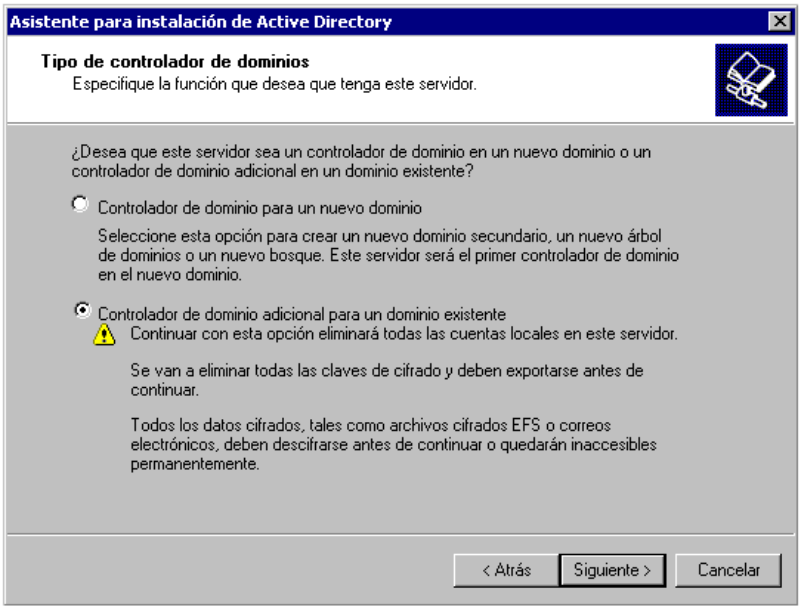


Figura 45. Asistente para instalación de active directory (servidor sucursal01)

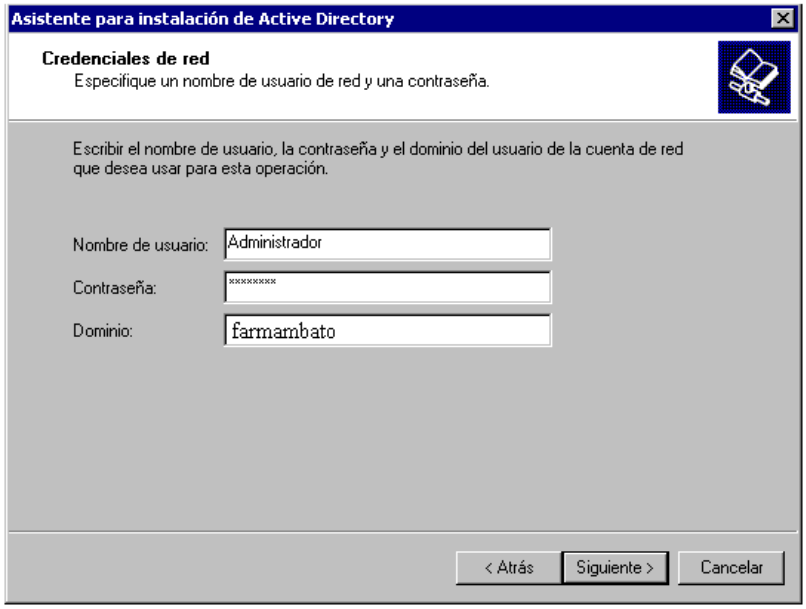


Figura 46. Asistente para instalación de active directory (servidor sucursal01)

Luego pedirá el Dominio Principal que será: **farmambato.com** y el Dominio Secundario: **sucursal01**, y por último clic en Siguiente.

4. **Configuración del Servidor de Nombres de Dominio (DNS) en el Servidor Sucursal01**

- Ir a Inicio, Programas, Herramientas Administrativas, escoger DNS.
- Ir a la carpeta Zona de Búsqueda Directa, ahí se encontrara el dominio que se creo con Active Directory en el servidor **principal**.
- Clic derecho en farmambato.com, escoger Host Nuevo, pedirá el Nombre que será **sucursal01,** la IP que será 100.100.100.102, clic en Agregar Host, en Aceptar y en Realizado.

- Clic derecho en farmambato.com, clic en Alias Nuevo, pedirá el Nombre de alias que será **www1**, clic en Examinar, en Sucursal01, en Zona de Búsqueda Directa, en farmambato.com, en sucursal01, en Aceptar y en Aceptar.

Quedando de la siguiente forma (Ver figura 47):

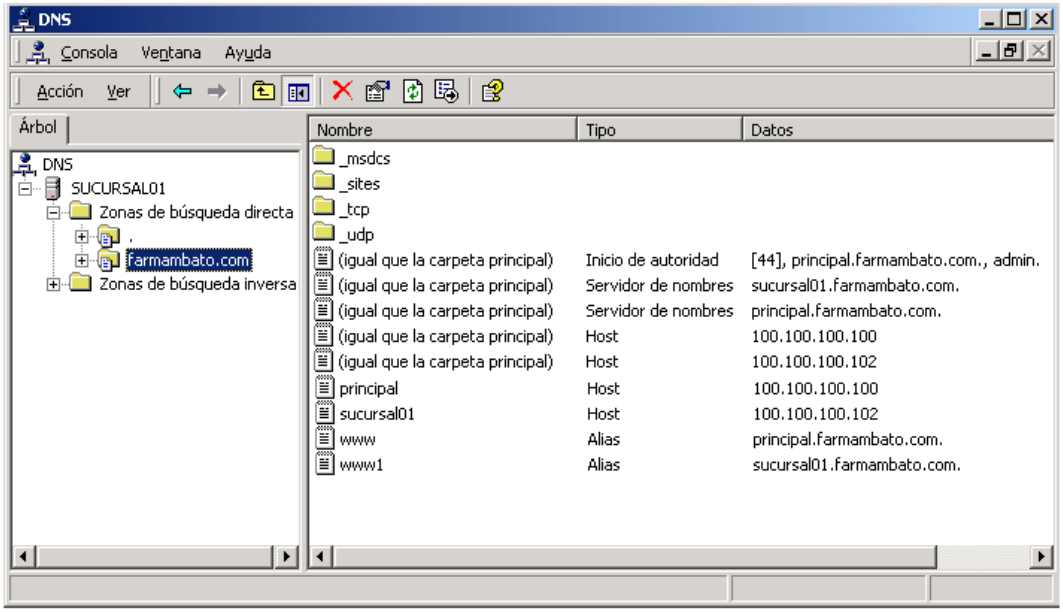


Figura 47. Configuración del servidor de nombres de dominio (servidor sucursal01)

5. Creación de Usuarios

Para la creación de usuarios se ha creado una Unidad Organizativa (Objeto contenedor utilizado para crear agrupaciones lógicas de objetos equipo, usuario y grupo) de nombre **Personal**.

Pulsar el botón Crear un nuevo usuario de la barra de herramientas de Usuarios y equipos de Active Directory, lo que produce un cuadro de dialogo.

En el cuadro de dialogo Nuevo objeto, hay que especificar el nombre y apellidos del usuario y el nombre de inicio de sesión que proporcionara el usuario cuando se conecte a la red. El nombre de inicio de sesión de nivel inferior para el usuario (esto es, el nombre con el que iniciará sesión el usuario en las estaciones de trabajo Windows NT o Windows 9.x) aparece entonces automáticamente. El cuadro de dialogo que proporciona un campo para la contraseña del objeto usuario y permite establecer opciones básicas para la contraseña y la cuenta para el usuario.

Después de que una pantalla de resumen confirme la información introducida, Usuarios y equipos de Active Directory crea el objeto usuario en el contenedor seleccionado en este caso la unidad organizativa **Personal.**

Quedando de la siguiente forma ya que se actualiza en los dos servidores (Ver figuras 48 y49):

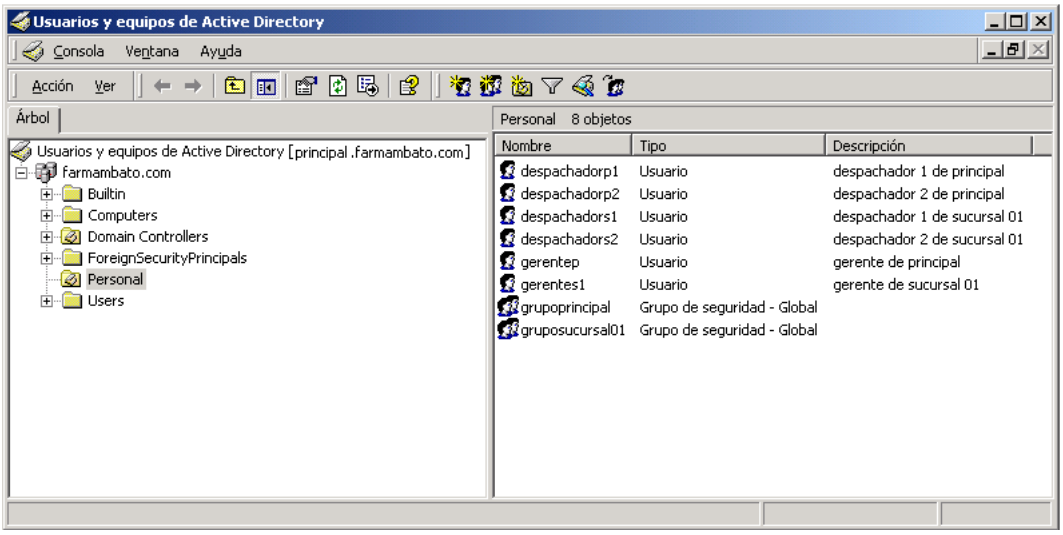


Figura 48. Usuarios y equipos de active directory (servidor principal)

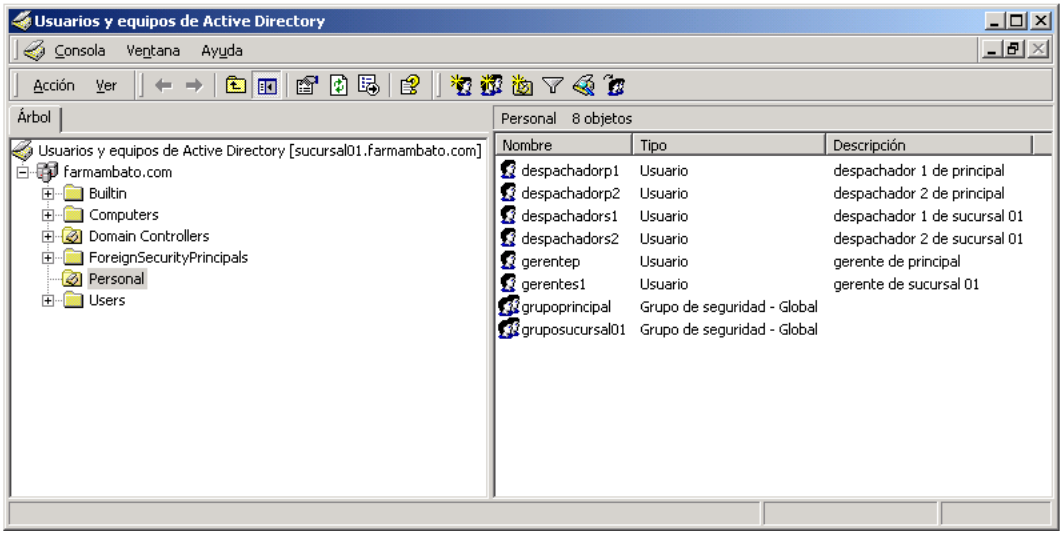


Figura 49. Usuarios y equipos de active directory (servidor sucursal01)

Quedando la red con dominio **farmambato.com** constituida de la siguiente forma (Ver figura 50):

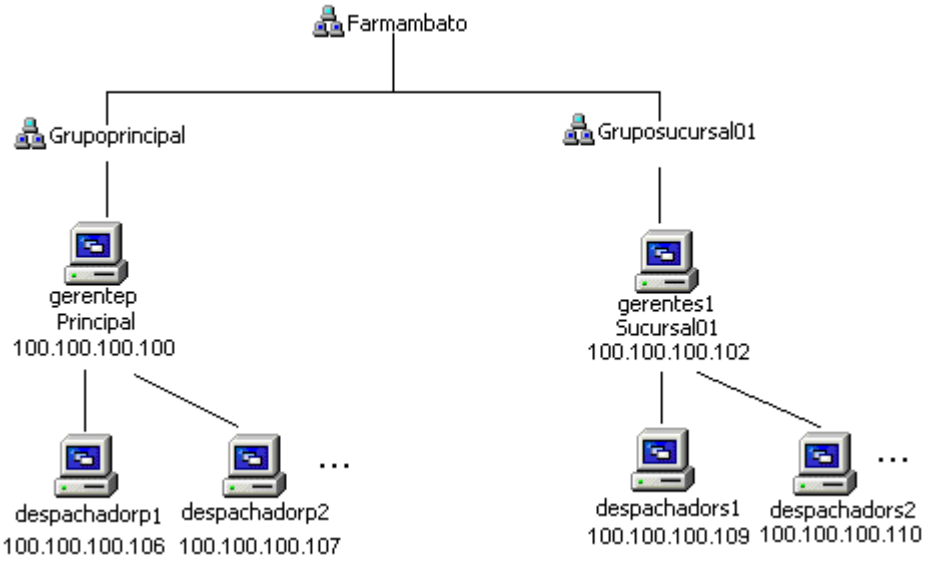


Figura 50. Red del dominio farmambato.com

La replicación se hará entre los servidores Principal y Sucursal01.

b. CREACIÓN DEL NOMBRE DEL SERVICIO PARA LA CONEXIÓN A LA BASE DE DATOS EN ORACLE DEVELOPER

Servidor Sucursal 01

Se siguen los siguientes pasos (Ver figuras 51 - 58):

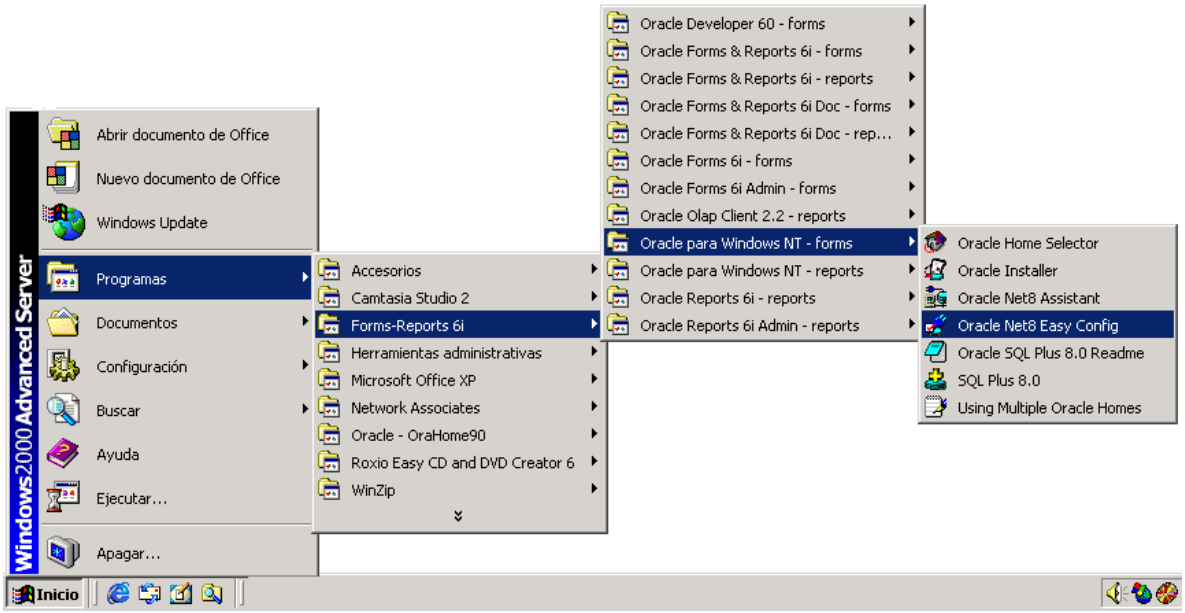


Figura 51. Pasos para la creación del nombre del servicio para la conexión con oracle developer

Añadir el nombre del servicio para este caso será **farmacia**, clic en Siguiente.

Figura 52. Pasos para la creación del nombre del servicio para la conexión con oracle developer

Por omisión se coge TCP/IP (Protocolo Internet), clic en Siguiente.

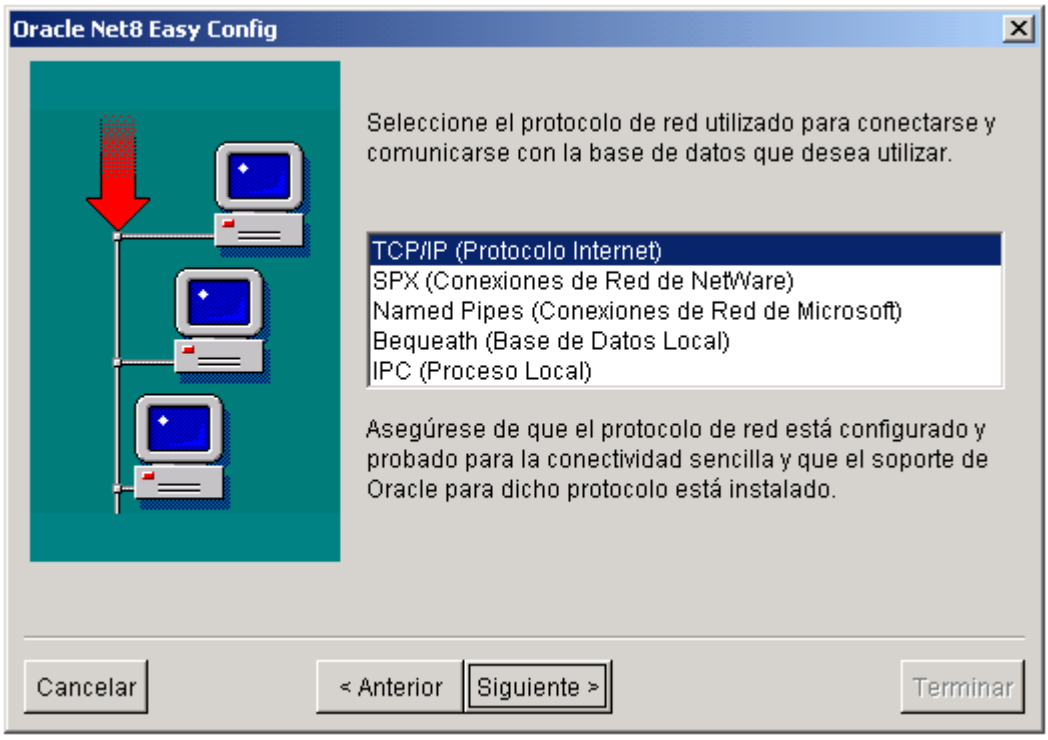


Figura 53. Pasos para la creación del nombre del servicio para la conexión con oracle developer

El Nombre del Host que es el nombre del servidor, el número de puerto es el valor por omisión, clic en Siguiente.

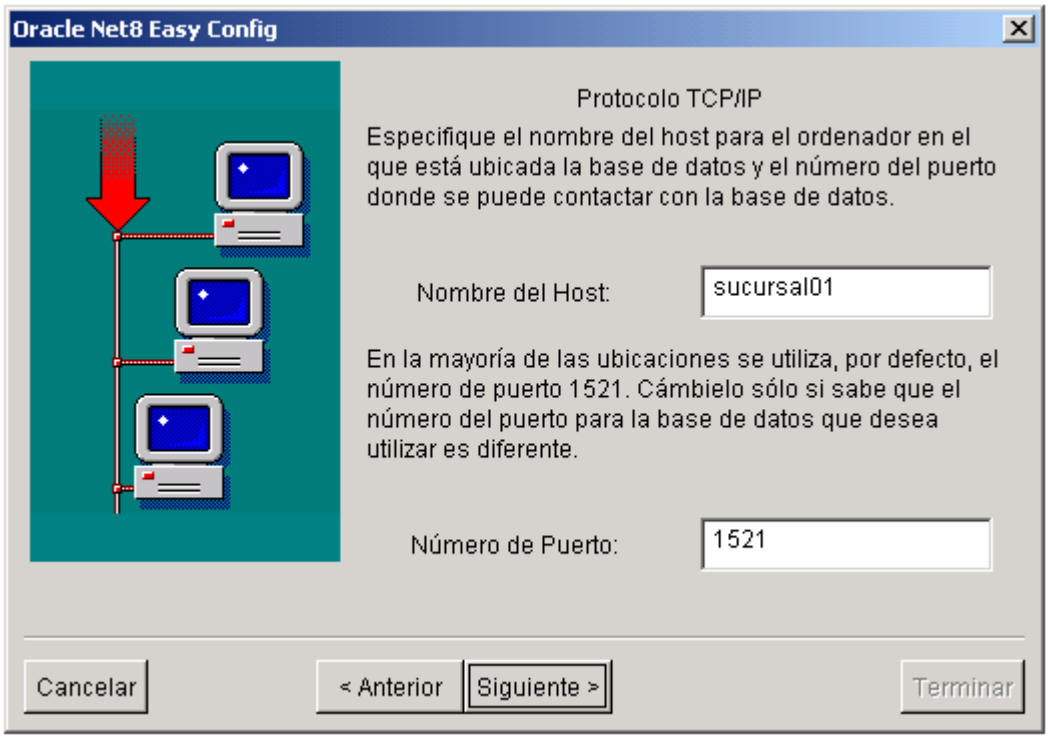


Figura 54. Pasos para la creación del nombre del servicio para la conexión con oracle developer

El SID de Base de Datos es el nombre de la Base de Datos Global de Oracle, clic en Siguiente.

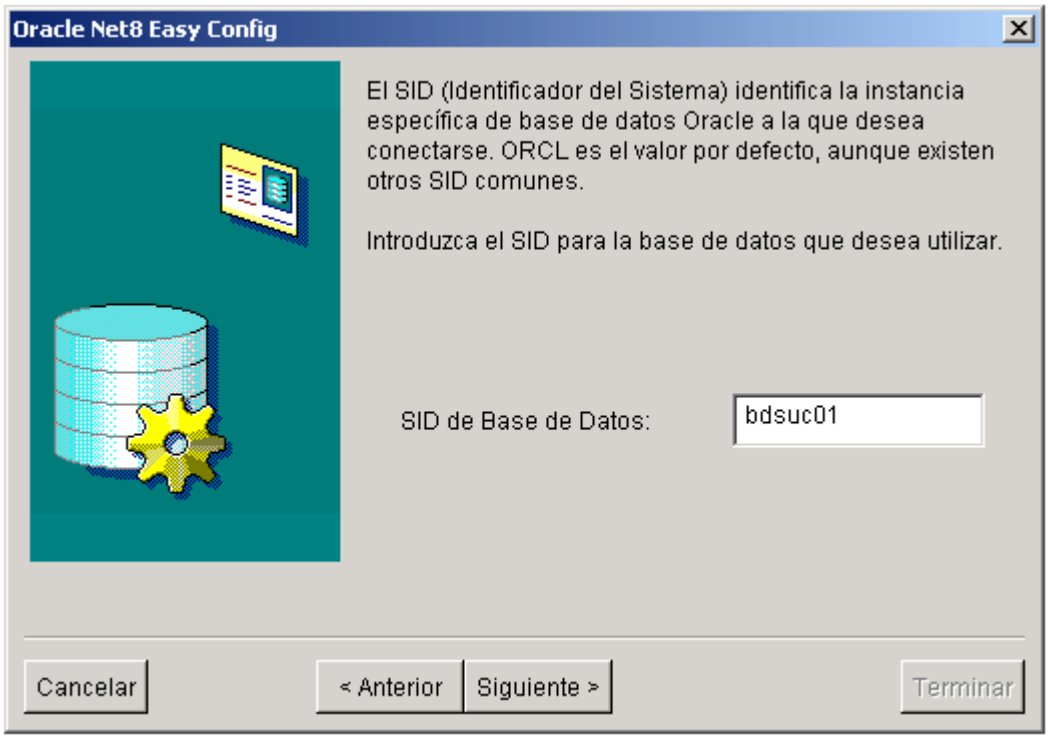


Figura 55. Pasos para la creación del nombre del servicio para la conexión con oracle developer

Clic en probar el servicio.

Figura 56. Pasos para la creación del nombre del servicio para la conexión con oracle developer

Ingresar el Nombre de Usuario y la Clave, clic en Probar, si la conexión se ha realizado correctamente hacer clic en Cerrar.

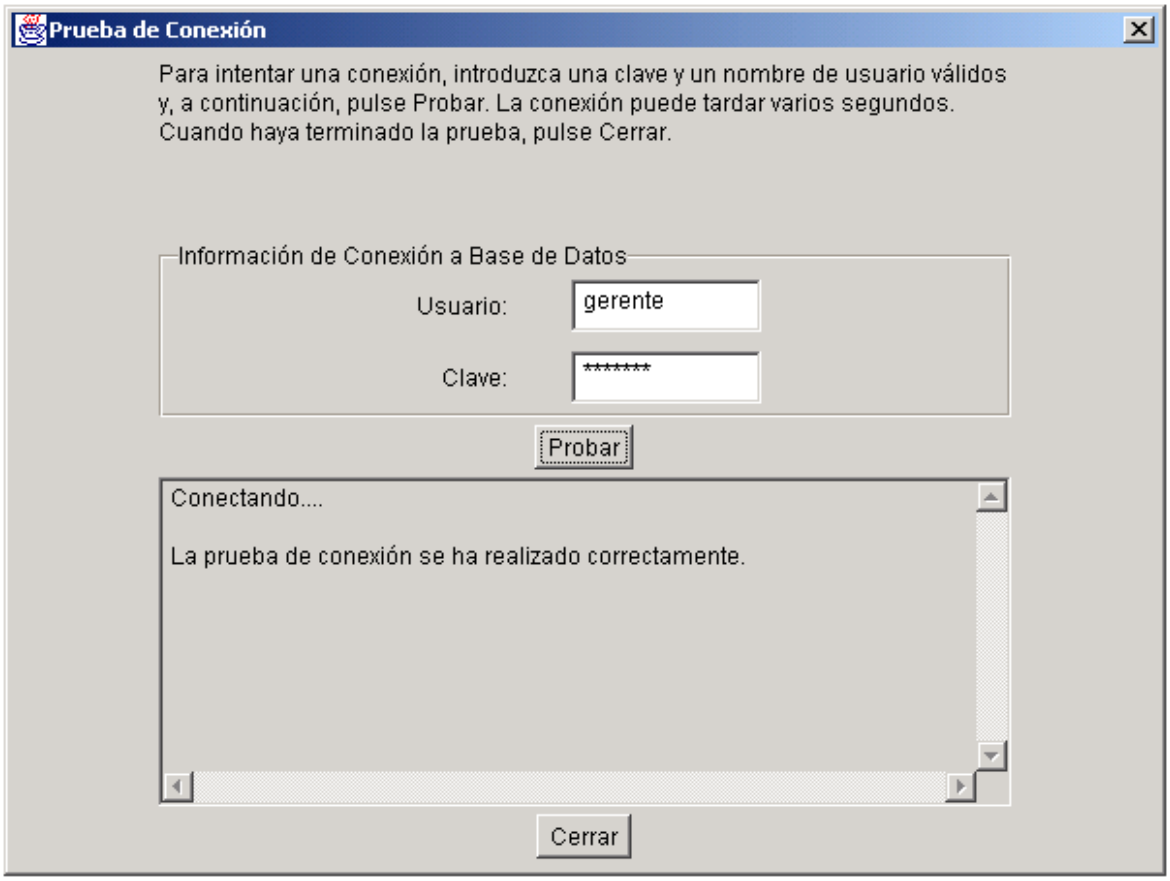


Figura 57. Pasos para la creación del nombre del servicio para la conexión con oracle developer

Finalmente clic en Terminar.

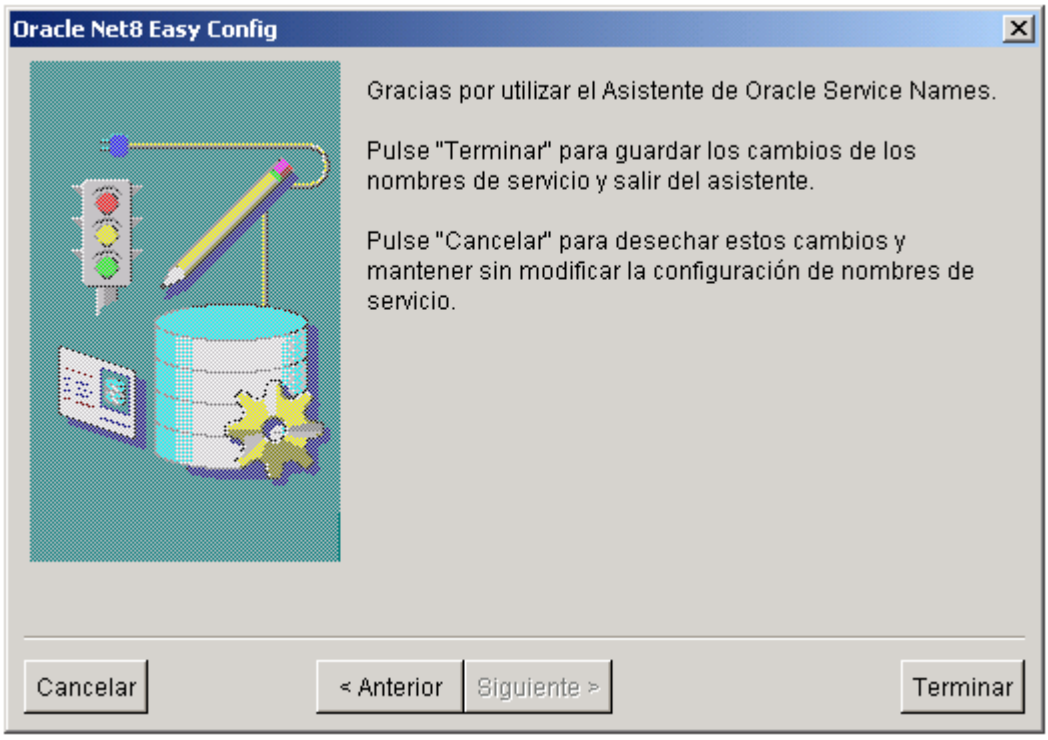


Figura 58. Pasos para la creación del nombre del servicio para la conexión con oracle developer

Servidor Principal

En el Servidor Principal por ser Pentium IV la creación del nombre del servicio se lo realiza de la siguiente forma:

- Ir a la carpeta donde se encuentra instalado Oracle Developer y acceder a:

  \ORA90\NET80\ADMIN\

- Abrir el archivo de nombre TNSNAMES.ORA e ir a:

  EXAMPLE1.WORLD =

  (DESCRIPTION =

(ADDRESS = (COMMUNITY = tcp.world)(PROTOCOL = TCP)(Host = Production1)(Port = 1521))

(CONNECT_DATA = (SID = SID1)))

- Cambiar lo anterior por:

FARMACIA =

(DESCRIPTION =

(ADDRESS = (PROTOCOL = TCP)(HOST = principal)(PORT = 1521))

(CONNECT_DATA = (SID = bdprin)))

De la misma forma se realiza la creación del nombre del servicio de conexión para la Base de Datos en Oracle Developer, para las máquinas clientes.

c. CONFIGURACION DEL NOMBRE DE SERVICIO DE RED

Servidor Sucursal 01

Se siguen los siguientes pasos (Ver figuras 59 - 71):

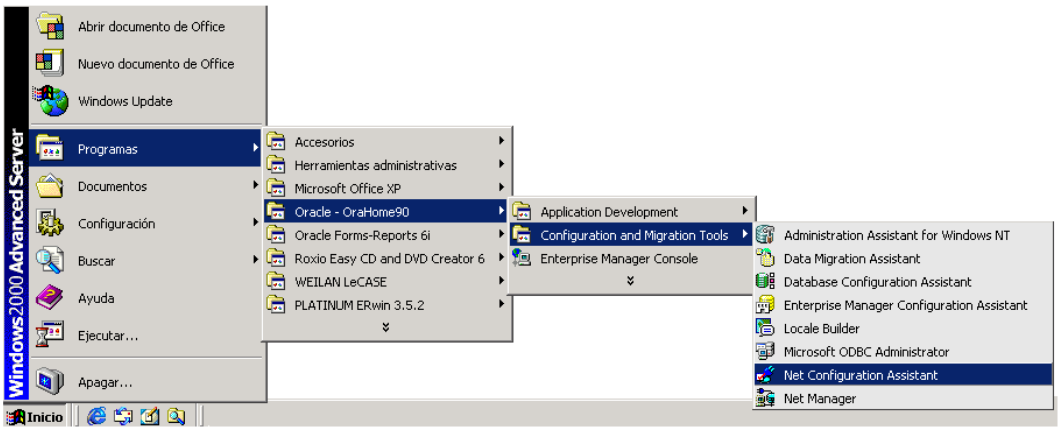


Figura 59. Configuración del nombre de servicio de red (sucursal01)

Se escoge Net Configuration Assistant.

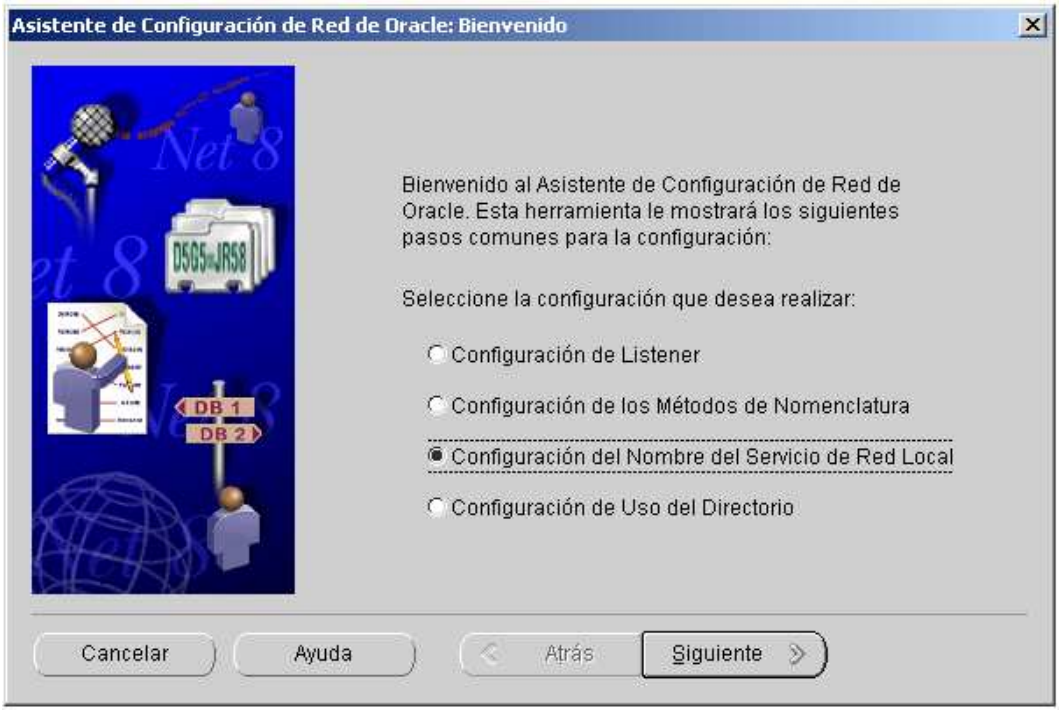


Figura 60. Configuración del nombre de servicio de red (sucursal01)

Clic en Configuración del Nombre del Servicio de Red Local, clic en Siguiente.

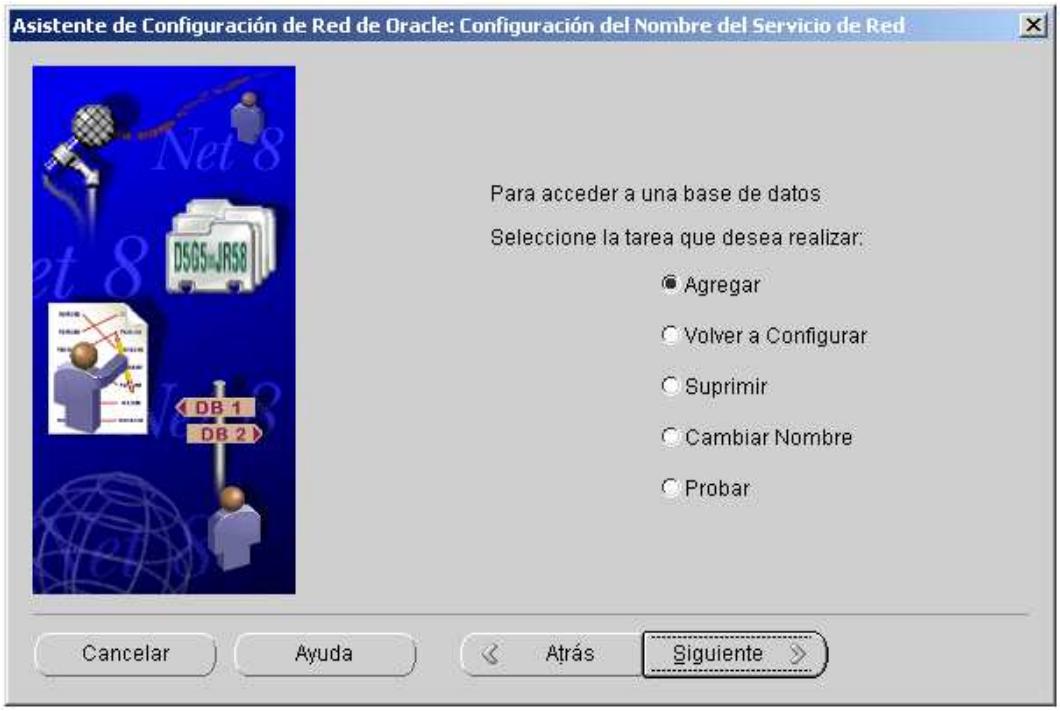


Figura 61. Configuración del nombre de servicio de red (sucursal01)

Clic en Agregar y en Siguiente.

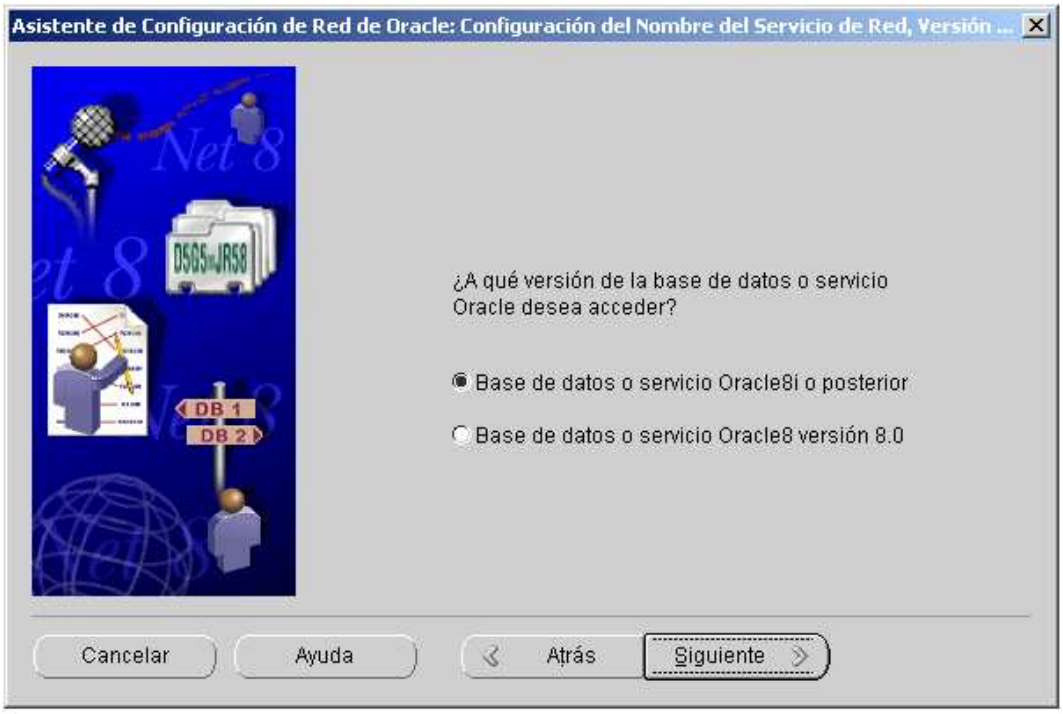


Figura 62. Configuración del nombre de servicio de red (sucursal01)

Escoger Base de datos o servicio Oracle8i o posterior, clic en Siguiente.

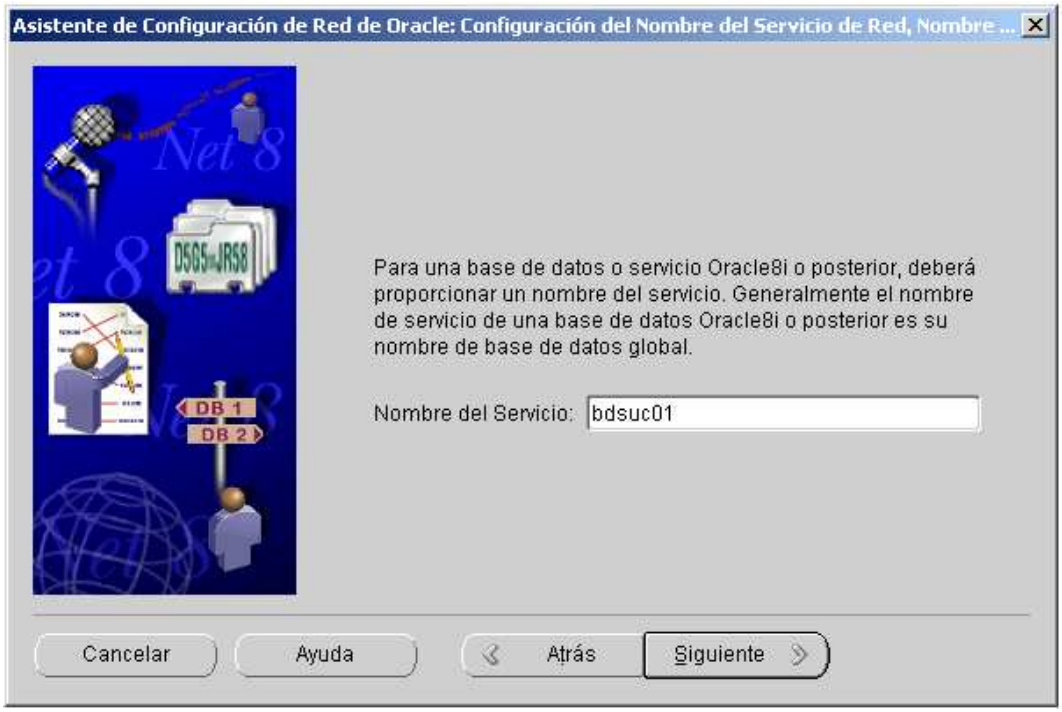


Figura 63. Configuración del nombre de servicio de red (sucursal01)

Poner el nombre de la base de datos global de Oracle, clic en Siguiente.

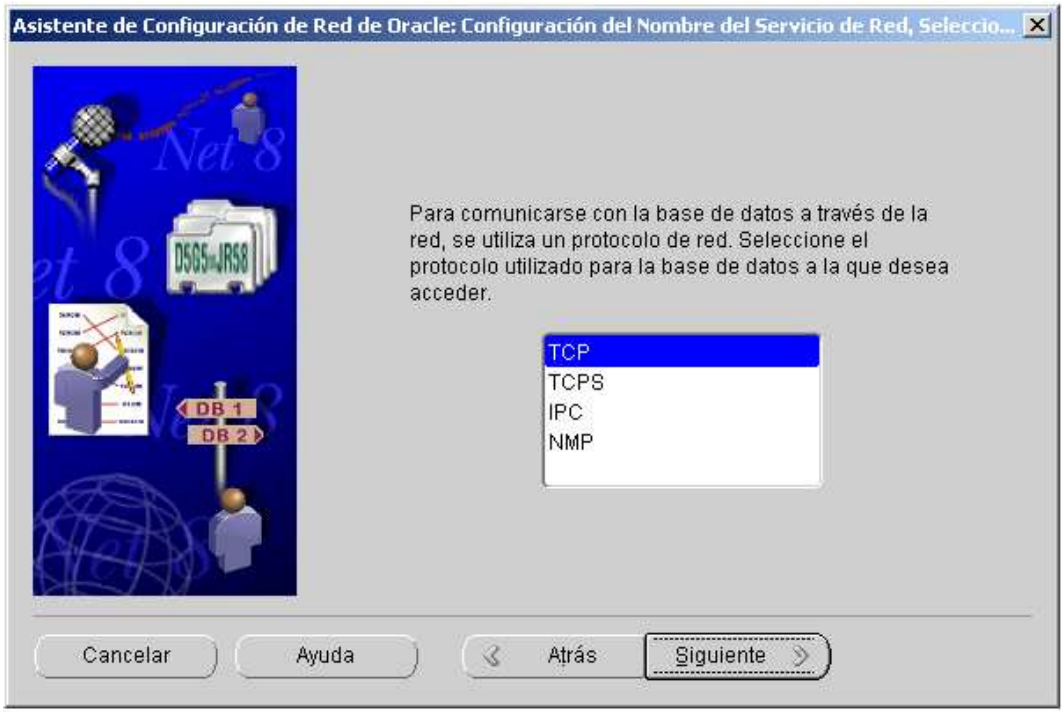


Figura 64. Configuración del nombre de servicio de red (sucursal01)

Por defecto coge TCP, clic en Siguiente.

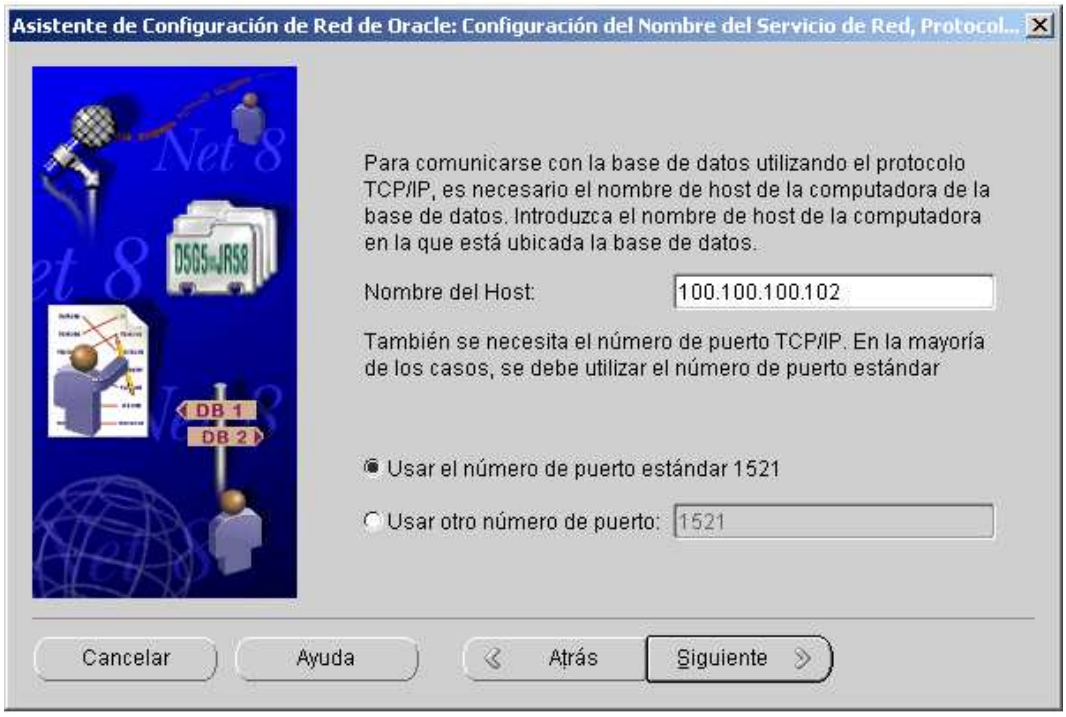


Figura 65. Configuración del nombre de servicio de red (sucursal01)

Nombre del host es el nombre del servidor o la dirección IP, clic en Siguiente.

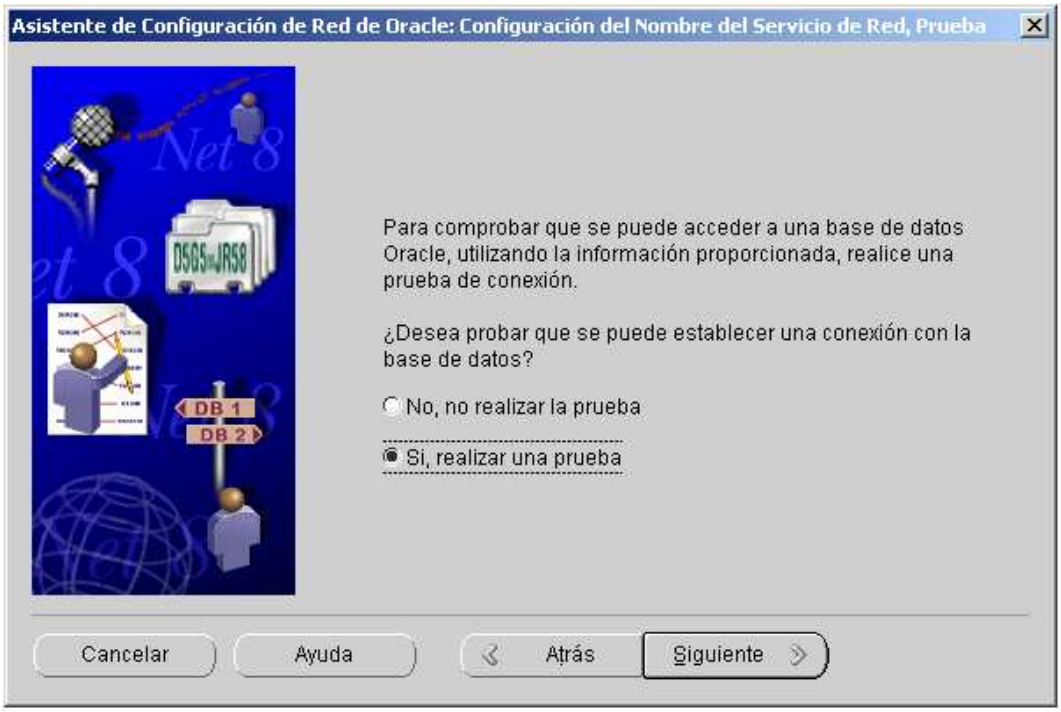


Figura 66. Configuración del nombre de servicio de red (sucursal01)

Si realizar la prueba, Siguiente

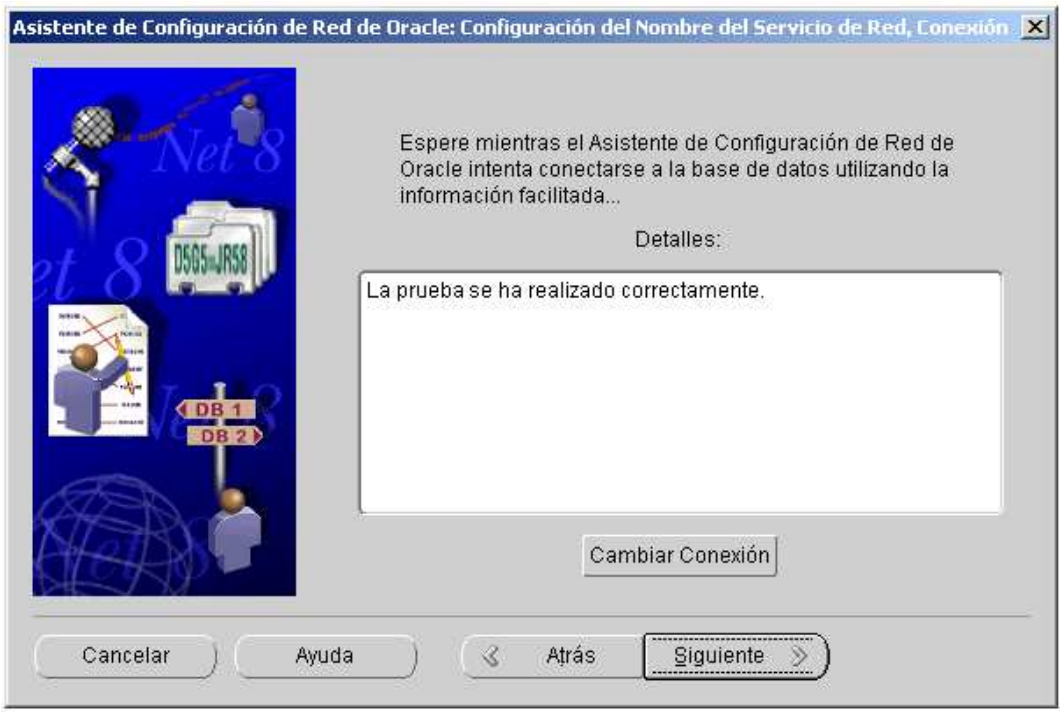


Figura 67. Configuración del nombre de servicio de red (sucursal01)

Clic en Siguiente.

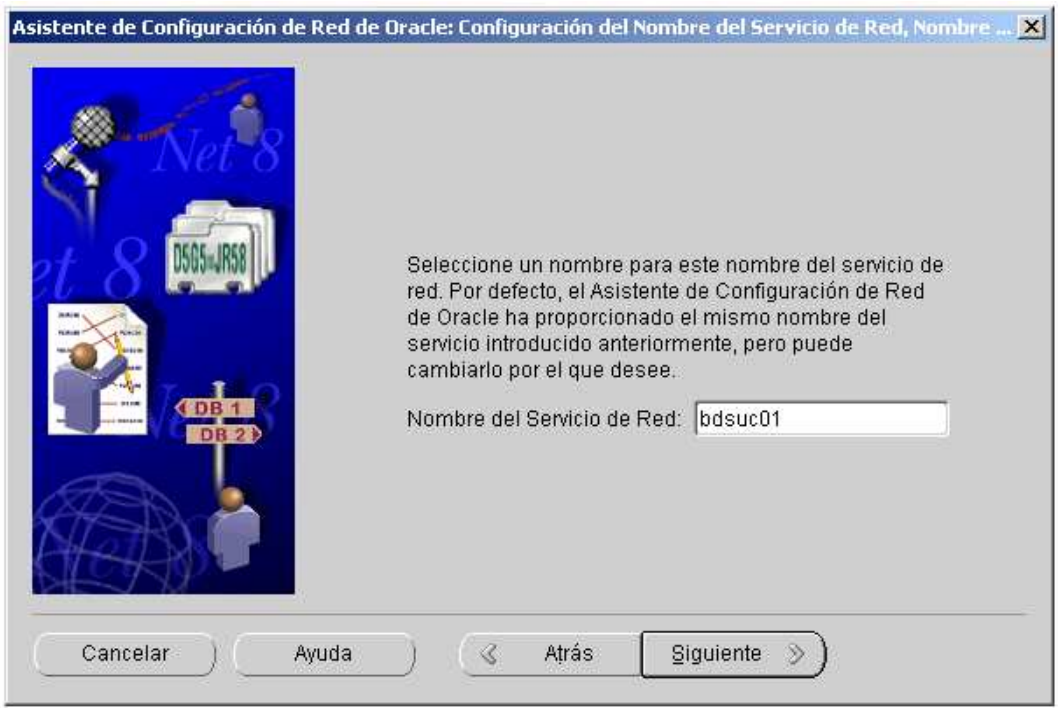


Figura 68. Configuración del nombre de servicio de red (sucursal01)

Clic en Siguiente.

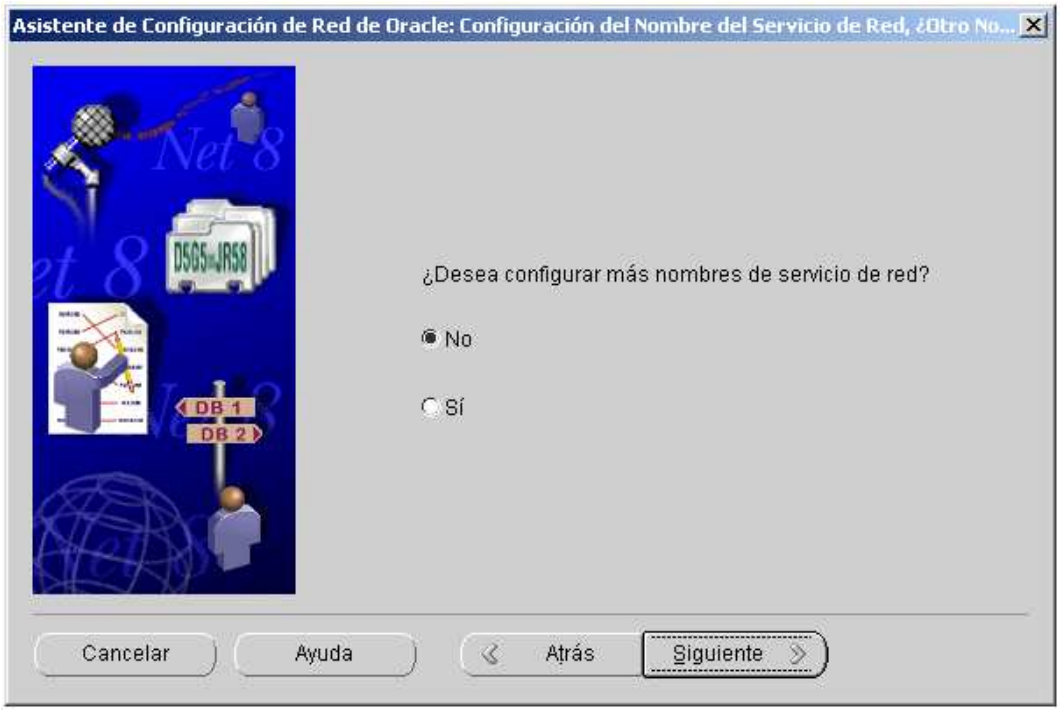


Figura 69. Configuración del nombre de servicio de red (sucursal01)

Opción No, si no se desea configurar más nombres, clic en Siguiente.

Figura 70. Configuración del nombre de servicio de red (sucursal01)

Clic en Siguiente.

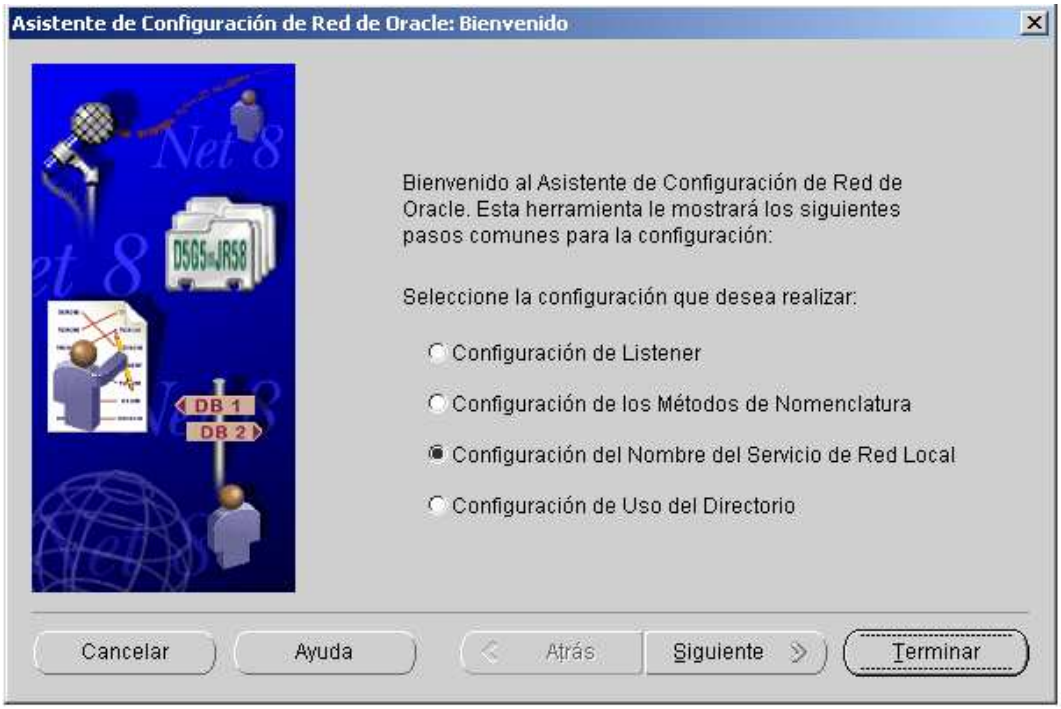


Figura 71. Configuración del nombre de servicio de red (sucursal01)

Clic en Terminar.

Servidor Principal

Para el servidor principal se siguen los pasos anteriores lo único que cambia es:

El nombre del servicio, que para el servidor principal será **bdprin** y el nombre del host, que será 100.100.100.100. (Ver figuras 72 y 73).

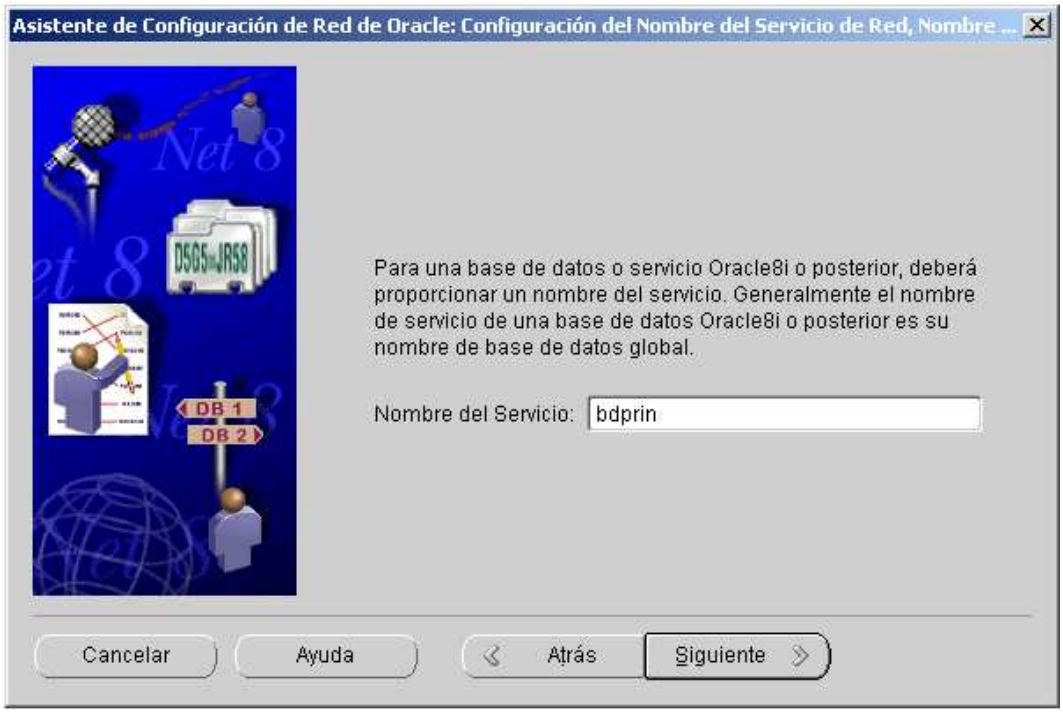


Figura 72. Configuración del nombre de servicio de red (principal)

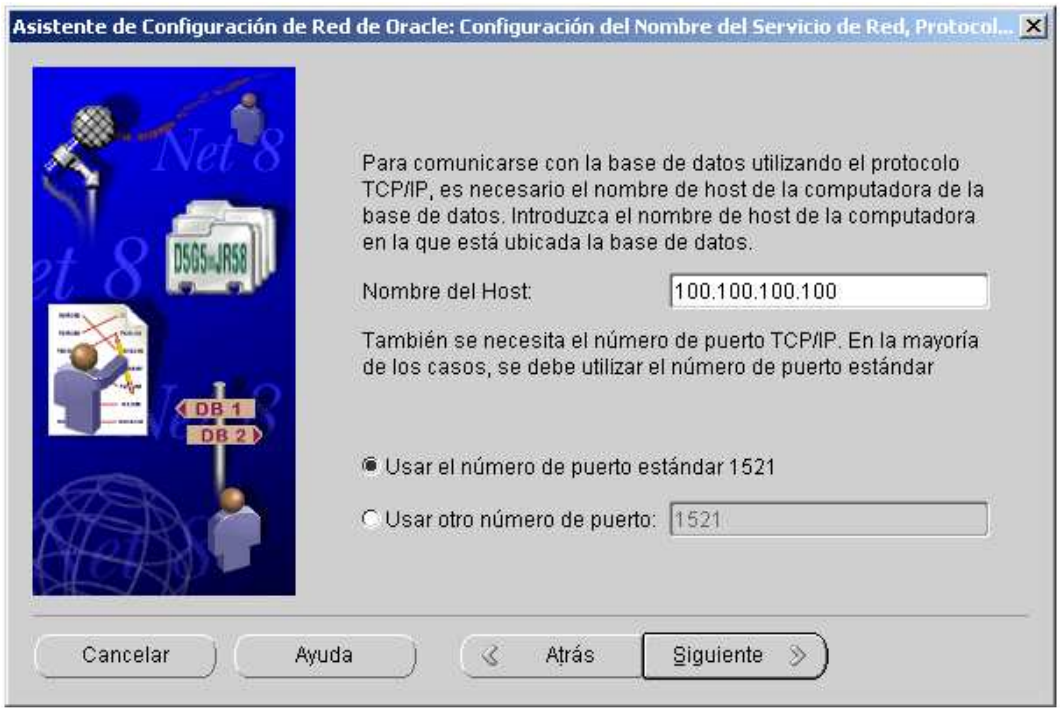


Figura 73. Configuración del nombre de servicio de red (principal)

Para las máquinas clientes es igual la configuración del nombre de servicio de red, y se debe configurar, el Nombre del Servicio y el Nombre del Host para el servidor que pertenece.

ANEXO II

a. Código generado por la Herramienta Case ERwin para la Creación de las Tablas de la Base de Datos.

```
DROP TABLE usuarios CASCADE CONSTRAINTS;
CREATE TABLE usuarios (
    ciusu           VARCHAR2(10) NOT NULL,
    nombusu          VARCHAR2(21) NULL,
    claveusu         VARCHAR2(30) NOT NULL,
    cargousu         VARCHAR2(20) NULL,
    dirusu           VARCHAR2(40) NULL,
    telfusu          VARCHAR2(9) NULL,
    ciudausu         VARCHAR2(15) NULL,
    tipoemp          VARCHAR2(30) NULL,
    ciudademp        VARCHAR2(30) NULL,
    direccionemp     VARCHAR2(30) NULL,
    PRIMARY KEY (claveusu));


DROP TABLE presentacion CASCADE CONSTRAINTS;
CREATE TABLE presentacion (
    codpre           VARCHAR2(4) NOT NULL,
    nombpre          VARCHAR2(15) NULL,
    PRIMARY KEY (codpre));


DROP TABLE categorias CASCADE CONSTRAINTS;
CREATE TABLE categorias (
    codcat           VARCHAR2(6) NOT NULL,
    nombcat          VARCHAR2(20) NULL,
    PRIMARY KEY (codcat));


DROP TABLE productos CASCADE CONSTRAINTS;
CREATE TABLE productos (
    codprod          VARCHAR2(7) NOT NULL,
    codcat           VARCHAR2(6) NOT NULL,
    codpre           VARCHAR2(4) NOT NULL,
```

```
    nombprod          VARCHAR2(25) NULL,
    precioventuni     NUMBER(10,2) NULL,
    generico          VARCHAR2(2) NULL,
    ivaprod           VARCHAR2(2) NULL,
    PRIMARY KEY (codprod, codcat, codpre),
    FOREIGN KEY (codpre)
    REFERENCES presentacion,
    FOREIGN KEY (codcat)
    REFERENCES categorias);


DROP TABLE clientes CASCADE CONSTRAINTS;
CREATE TABLE clientes (
    cicli             VARCHAR2(10) NOT NULL,
    nombcli           VARCHAR2(20) NULL,
    apelcli           VARCHAR2(20) NULL,
    dircli            VARCHAR2(40) NULL,
    telfcli           VARCHAR2(9) NULL,
    ciudacli          VARCHAR2(15) NULL,
    emailcli          VARCHAR2(40) NULL,
    faxcli            VARCHAR2(9) NULL,
    observacion       VARCHAR2(25) NULL,
    PRIMARY KEY (cicli));


DROP TABLE encfacven CASCADE CONSTRAINTS;
CREATE TABLE encfacven (
    codfacven         VARCHAR2(7) NOT NULL,
    cicli             VARCHAR2(10) NOT NULL,
    nombcli           VARCHAR2(20) NULL,
    apelcli           VARCHAR2(20) NULL,
    fechaven          DATE DEFAULT sysdate NULL,
    totalivaven       NUMBER(10,2) NULL,
    subtotal          NUMBER(10,2) NULL,
    totalven          NUMBER(10,2) NULL,
    PRIMARY KEY (codfacven, cicli),
    FOREIGN KEY (cicli)
    REFERENCES clientes);


DROP TABLE detfacven CASCADE CONSTRAINTS;
CREATE TABLE detfacven (
    codfacven         VARCHAR2(7) NOT NULL,
    cicli             VARCHAR2(10) NOT NULL,
```

```
    codprod          VARCHAR2(7) NOT NULL,
    codcat           VARCHAR2(6) NOT NULL,
    codpre           VARCHAR2(4) NOT NULL,
    nombpre          VARCHAR2(15) NULL,
    cantven          NUMBER(10,2) NULL,
    preciovenuni     NUMBER(10,2) NULL,
    ivaven           VARCHAR2(2) NULL,
    subtotalven      NUMBER(10,2) NULL,
    auxiva           NUMBER(10,2) NULL,
    PRIMARY KEY (codfacven, cicli, codprod, codcat, codpre),
    FOREIGN KEY (codprod, codcat, codpre)
    REFERENCES productos,
    FOREIGN KEY (codfacven, cicli)
    REFERENCES encfacven);


DROP TABLE proveedores CASCADE CONSTRAINTS;
CREATE TABLE proveedores (
    codprov          VARCHAR2(7) NOT NULL,
    nombprov         VARCHAR2(20) NULL,
    apelprov         VARCHAR2(20) NULL,
    nombemp          VARCHAR2(25) NULL,
    dirprov          VARCHAR2(40) NULL,
    telfprov         VARCHAR2(9) NULL,
    ciudaprov        VARCHAR2(15) NULL,
    emailprov        VARCHAR2(40) NULL,
    faxprov          VARCHAR2(9) NULL,
    PRIMARY KEY (codprov));


DROP TABLE encfaccom CASCADE CONSTRAINTS;
CREATE TABLE encfaccom (
    codfaccom        VARCHAR2(7) NOT NULL,
    codprov          VARCHAR2(7) NOT NULL,
    nombprov         VARCHAR2(20) NULL,
    apelprov         VARCHAR2(20) NULL,
    nombemp          VARCHAR2(15) NULL,
    fechacom         DATE DEFAULT sysdate NULL,
    totalivacom      NUMBER(10,2) NULL,
    subtotalcom      NUMBER(10,2) NULL,
    totalcom         NUMBER(10,2) NULL,
    PRIMARY KEY (codfaccom, codprov),
    FOREIGN KEY (codprov)
    REFERENCES proveedores);
```

```
DROP TABLE detfaccom CASCADE CONSTRAINTS;
CREATE TABLE detfaccom (
    codfaccom          VARCHAR2(7) NOT NULL,
    codprov            VARCHAR2(7) NOT NULL,
    codprod            VARCHAR2(7) NOT NULL,
    codcat             VARCHAR2(6) NOT NULL,
    codpre             VARCHAR2(4) NOT NULL,
    nombprod           VARCHAR2(25) NULL,
    nombpre            VARCHAR2(15) NULL,
    cantcom            NUMBER(10,2) NULL,
    preciocomuni       NUMBER(10,2) NULL,
    ivacom             VARCHAR2(2) NULL,
    subtotaldetalle    NUMBER(10,2) NULL,
    auxivac            NUMBER(10,2) NULL,
    PRIMARY KEY (codfaccom, codprov, codprod, codcat, codpre),
    FOREIGN KEY (codprod, codcat, codpre)
    REFERENCES productos,
    FOREIGN KEY (codfaccom, codprov)
    REFERENCES encfaccom);


DROP TABLE exiprod CASCADE CONSTRAINTS;
CREATE TABLE exiprod (
    codpre             VARCHAR2(4) NOT NULL,
    codprod            VARCHAR2(7) NOT NULL,
    codcat             VARCHAR2(6) NOT NULL,
    nombprod           VARCHAR2(25) NULL,
    exiuni             INTEGER NULL,
    eximin             INTEGER NULL,
    eximax             INTEGER NULL,
    fechacadpro        DATE NULL,
    PRIMARY KEY (codpre, codprod, codcat),
    FOREIGN KEY (codprod, codcat, codpre)
    REFERENCES productos,
    FOREIGN KEY (codpre)
    REFERENCES presentacion);
```

b. Script de Permisos para las Tablas y Sinónimos

Permisos para las Tablas

```
grant select on clientes to despachador;
grant insert on clientes to despachador;
grant update on clientes to despachador;

grant select on categorias to despachador;
grant insert on categorias to despachador;
grant update on categorias to despachador;

grant select on presentacion to despachador;
grant insert on presentacion to despachador;
grant update on presentacion to despachador;

grant select on productos to despachador;
grant insert on productos to despachador;
grant update on productos to despachador;

grant select on exiprod to despachador;
grant insert on exiprod to despachador;
grant update on exiprod to despachador;

grant select on proveedores to despachador;
grant insert on proveedores to despachador;
grant update on proveedores to despachador;

grant select on encfaccom to despachador;
grant insert on encfaccom to despachador;
grant update on encfaccom to despachador;
```

grant select on detfaccom to despachador;
grant insert on detfaccom to despachador;
grant update on detfaccom to despachador;

grant select on encfacven to despachador;
grant insert on encfacven to despachador;
grant update on encfacven to despachador;

grant select on detfacven to despachador;
grant insert on detfacven to despachador;
grant update on detfacven to despachador;

grant select on usuarios to despachador;

Permisos para los Sinónimos

grant select on clientess1 to despachador;
grant insert on clientess1 to despachador;
grant update on clientess1 to despachador;

grant select on categoriass1 to despachador;
grant insert on categoriass1 to despachador;
grant update on categoriass1 to despachador;

grant select on presentacions1 to despachador;
grant insert on presentacions1 to despachador;
grant update on presentacions1 to despachador;

grant select on productoss1 to despachador;
grant insert on productoss1 to despachador;
grant update on productoss1 to despachador;

```
grant select on exiprods1 to despachador;
grant insert on exiprods1 to despachador;
grant update on exiprods1 to despachador;


grant select on proveedoress1 to despachador;
grant insert on proveedoress1 to despachador;
grant update on proveedoress1 to despachador;


grant select on encfaccoms1 to despachador;
grant insert on encfaccoms1 to despachador;
grant update on encfaccoms1 to despachador;


grant select on detfaccoms1 to despachador;
grant insert on detfaccoms1 to despachador;
grant update on detfaccoms1 to despachador;


grant select on encfacvens1 to despachador;
grant insert on encfacvens1 to despachador;
grant update on encfacvens1 to despachador;


grant select on detfacvens1 to despachador;
grant insert on detfacvens1 to despachador;
grant update on detfacvens1 to despachador;


grant select on usuarioss1 to despachador;
```

c. Script de Secuencias, Triggers, y Vistas

Secuencia factura

```
create sequence s_factura increment by 1 start  with 1;
```

Secuencia producto

```
create sequence s_producto increment by 1 start  with 1;
```

Trigger para restar existencia

```
CREATE OR REPLACE TRIGGER  VENTA
AFTER INSERT
ON DETFACVEN
FOR EACH ROW
BEGIN
UPDATE EXIPROD SET EXIUNI  = EXIUNI  - (:NEW.CANTVEN)
WHERE CODPROD = :NEW.CODPROD;
END;
/
```

Trigger para sumar existencia

```
CREATE OR REPLACE TRIGGER  COMPRA
AFTER INSERT
ON DETFACCOM
FOR EACH ROW
BEGIN
UPDATE EXIPROD SET EXIUNI  = EXIUNI  + (:NEW.CANTCOM)
WHERE CODPROD = :NEW.CODPROD;
END;
/
```

Vistas de join entre tablas Productos y Presentación

```
create or replace  view pr as
select p.codprod,p.nombprod, p.precioventuni, p.ivaprod, r.nombpre
from productos p,  presentacion r
where p.codpre = r.codpre;
```

```
create or replace  view compras as
select p.codprod,p.nombprod, p.ivaprod, r.nombpre
from productos p,  presentacion r
where p.codpre = r.codpre;
```

Vista de join entre tablas Clientes y Encabezado de Factura de Venta

```
create or replace view cliente_venta as
select c.cicli,c.nombcli,c.apelcli,c.dircli,c.telfcli, v.fechaven,v.totalven
from clientes c, encfacven v
where c.cicli=v.cicli;
```

d. Script de Replicación mediante Snapshot entre los Servidores Principal y Sucursal01.

Replicación desde la Base de Datos BDSUC01 a la Base de Datos BDPRIN

BDSUC01

1) create snapshot log on clientess1 with primary key;

create snapshot log on categoriass1 with primary key;

create snapshot log on presentacions1 with primary key;

create snapshot log on productoss1 with primary key;

create snapshot log on exiprods1 with rowid;

create snapshot log on proveedoress1 with primary key;

create snapshot log on encfaccoms1 with primary key;

create snapshot log on detfaccoms1 with rowid;

create snapshot log on encfacvens1 with primary key;

create snapshot log on detfacvens1 with rowid;

create snapshot log on usuarioss1 with primary key;

BDPRIN

1) create database link bdsuc01
connect to gerente identified by gerente
using 'bdsuc01';

2) create snapshot s_clientess1
refresh fast start with sysdate next sysdate + 1/86400
with primary key as
select * from clientess1@bdsuc01;

create snapshot s_categoriass1
refresh fast start with sysdate next sysdate + 1/86400
with primary key as
select * from categoriass1@bdsuc01;

create snapshot s_presentacions1
refresh fast start with sysdate next sysdate + 1/86400
with primary key as
select * from presentacions1@bdsuc01;

create snapshot s_productoss1
refresh fast start with sysdate next sysdate + 1/86400
with primary key as
select * from productoss1@bdsuc01;

create snapshot s_exiprods1
refresh fast start with sysdate next sysdate + 1/86400
with rowid as
select * from exiprods1@bdsuc01;

create snapshot s_proveedoress1
refresh fast start with sysdate next sysdate + 1/86400
with primary key as
select * from proveedoress1@bdsuc01;

```
create snapshot s_encfaccoms1
refresh fast start with sysdate next sysdate + 1/86400
with primary key as
select * from encfaccoms1@bdsuc01;


create snapshot s_detfaccoms1
refresh fast start with sysdate next sysdate + 1/86400
with rowid as
select * from detfaccoms1@bdsuc01;


create snapshot s_encfacvens1
refresh fast start with sysdate next sysdate + 1/86400
with primary key as
select * from encfacvens1@bdsuc01;


create snapshot s_detfacvens1
refresh fast start with sysdate next sysdate + 1/86400
with rowid as
select * from detfacvens1@bdsuc01;


create snapshot s_usuarioss1
refresh fast start with sysdate next sysdate + 1/86400
with primary key as
select * from usuarioss1@bdsuc01;
```

3) create synonym clientess1 for s_clientess1;

create synonym categoriass1 for s_categoriass1;

create synonym presentacions1 for s_presentacions1;

create synonym productoss1 for s_productoss1;

create synonym exiprods1 for s_exiprods1;

create synonym proveedoress1 for s_proveedoress1;

create synonym encfaccoms1 for s_encfaccoms1;

create synonym detfaccoms1 for s_detfaccoms1;

create synonym encfacvens1 for s_encfacvens1;

create synonym detfacvens1 for s_detfacvens1;

create synonym usuarioss1 for s_usuarioss1;

Replicación desde la Base de Datos BDPRIN a la Base de Datos BDSUC01

BDPRIN

1) create snapshot log on clientes with primary key;

create snapshot log on categorias with primary key;

create snapshot log on presentacion with primary key;

create snapshot log on productos with primary key;

create snapshot log on exiprod with rowid;

create snapshot log on proveedores with primary key;

create snapshot log on encfaccom with primary key;

create snapshot log on detfaccom with rowid;

create snapshot log on encfacven with primary key;

create snapshot log on detfacven with rowid;

create snapshot log on usuarios with primary key;

BDSUC01

1) create database link bdprin
connect to gerente identified by gerente
using 'bdprin';

2) create snapshot s_clientes
refresh fast start with sysdate next sysdate + 1/86400
with primary key as
select * from clientes@bdprin;

create snapshot s_categorias
refresh fast start with sysdate next sysdate + 1/86400
with primary key as
select * from categorias@bdprin;

create snapshot s_presentacion
refresh fast start with sysdate next sysdate + 1/86400
with primary key as
select * from presentacion@bdprin;

create snapshot s_productos
refresh fast start with sysdate next sysdate + 1/86400
with primary key as
select * from productos@bdprin;

create snapshot s_exiprod
refresh fast start with sysdate next sysdate + 1/86400
with rowid as
select * from exiprod@bdprin;

create snapshot s_proveedores
refresh fast start with sysdate next sysdate + 1/86400
with primary key as
select * from proveedores@bdprin;

```
create snapshot s_encfaccom
refresh fast start with sysdate next sysdate + 1/86400
with primary key as
select * from encfaccom@bdprin;

create snapshot s_detfaccom
refresh fast start with sysdate next sysdate + 1/86400
with rowid as
select * from detfaccom@bdprin;

create snapshot s_encfacven
refresh fast start with sysdate next sysdate + 1/86400
with primary key as
select * from encfacven@bdprin;

create snapshot s_detfacven
refresh fast start with sysdate next sysdate + 1/86400
with rowid as
select * from detfacven@bdprin;

create snapshot s_usuarios
refresh fast start with sysdate next sysdate + 1/86400
with primary key as
select * from usuarios@bdprin;
```

3) create synonym clientes for s_clientes;

create synonym categorias for s_categorias;

create synonym presentacion for s_presentacion;

create synonym productos for s_productos;

create synonym exiprod for s_exiprod;

create synonym proveedores for s_proveedores;

```
create synonym encfaccom for s_encfaccom;

create synonym detfaccom for s_detfaccom;

create synonym encfacven for s_encfacven;

create synonym detfacven for s_detfacven;

create synonym usuarios for s_usuarios;
```