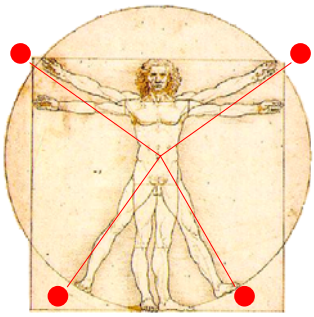
TECNOLOGÍ@ y *DESARROLLO*

Revista de Ciencia, Tecnología y Medio Ambiente

VOLUMEN VI . AÑO 2008

SEPARATA



DISEÑO DE UN SISTEMA GESTOR DE BASE DE DATOS DISTRIBUIDA BASADO EN ORACLE9i

Celia Gutiérrez Cosío

UNIVERSIDAD ALFONSO X EL SABIO
Escuela Politécnica Superior

Villanueva de la Cañada (Madrid)

ISSN: 1696-8085
Editor: Julio Merino García tecnologia@uax.es

DISEÑO DE UN SISTEMA GESTOR DE BASE DE DATOS DISTRIBUIDA BASADO EN ORACLE9i

Celia Gutiérrez Cosío

Dra Informática
Departamento de Ingeniería de Software e Inteligencia Artificial, Facultad de Informática, Universidad Complutense de Madrid
C/ Profesor José García Santesmases s/n, Ciudad Universitaria, 28040 Madrid, tlf.: 91 394 7576, e-mail: celiagutierrez@hotmail.com

RESUMEN: Los sistemas de información actuales se están enfocando cada vez más hacia arquitecturas descentralizadas y autónomas. Han surgido así los sistemas de datos distribuidas y más concretamente las bases de datos distribuidas para dar respuesta a estas necesidades. Los sistemas gestores de bases de datos distribuidas deben ofrecer al usuario un entorno que parezca centralizado, apareciendo así el concepto de transparencia a varios niveles: fragmentación, replicación y de consulta. En el presente trabajo, se ha diseñado un sistema gestor de base de datos distribuida que ofrece estas características de transparencia, utilizando la base de datos Oracle9i.

PALABRAS CLAVE: base de datos distribuida, sistema gestor de base de datos, transparencia, fragmentación, replicación, consulta.

ABSTRACT: Current information systems are focusing on increasingly descentralized and autonomous architectures. To give support to these needs in this way, distributed data systems and more exactly, distributed database systems have appeared. As distributed database management systems must offer a centralized looking environment to the user, transparency concept has appeared at several levels: fragmentation, replication and queries. In this work, a distributed database management system that offers these transparency features has been designed using Oracle 9i database.

KEYWORDS: distributed database, database management system, transparency, fragmentation, replication, query.

1 . Introducción

Desde la aparición de las primeras bases de datos, centralizadas, ha habido una evolución hacia la descentralización, paralela al desarrollo de redes informáticas, dando lugar a arquitecturas como cliente-servidor, bases de datos paralelas, y bases de datos distribuidas. De hecho, esta estructura de organización de la información empresarial imita a la propia estructura organizativa de la empresa, que está estructurada en unidades como divisiones, departamentos, etc. cada una de las cuales mantiene sus propios datos operacionales [DATE 2000]. Los sistemas gestores de bases de datos distribuidos se han diseñado para cubrir dos objetivos: la compartición de los datos y la eficiencia de acceso

a ellos, ubicando la información en aquellos nodos donde se acceda con más frecuencia, pero haciéndola a la vez disponible al resto de los nodos. Esta nueva filosofía de arquitectura de bases de datos nos ayuda a resolver el problema de las islas de información, nombre bajo el cual se han considerado a las bases de datos, por estar situadas en lugares remotos e incomunicados [CONOLLY 2005].

Se considera una BDD (base de datos distribuida) a "una colección lógicamente interrelacionada de datos compartidos (junto con una descripción de estos datos) fisicamente distribuidas por una red informática", [CONOLLY 2005].

Se considera un SGBDD (sistema gestor de base de datos distribuido) a un "sistema software que permite gestionar la base de datos distribuida y hace que dicha distribución sea transparente para los usuarios", [CONOLLY 2005].

Un SGBDD se compone de una única base de datos lógica, que físicamente está dividida en fragmentos ubicados en nodos distintos e interconectados mediante una red de comunicaciones. Así, existirán aplicaciones locales, que solo accederán a los datos almacenados en el mismo nodo, y aplicaciones globales, que accederán a datos de varios nodos. El SGBDD ejecutará así las funciones requeridas para dar servicio a este último tipo de aplicaciones.

Además de la fragmentación, en el contexto de BDD también se habla de replicación, que consiste en mantener réplicas de fragmentos en varios nodos, según una política de replicación que va desde una replicación mínima (un fragmento replicado en otro nodo) a una replicación total (todos los fragmentos son replicados). Nuevamente el SGBDD debe garantizar la consistencia de los fragmentos que están replicados (una actualización en un fragmento debe provocar la actualización en todas sus réplicas). Un caso de replicación son los datawarehouse: proporcionan soporte para consultas complejas para ayuda a la decisión que recurren a numerosas fuentes de datos, ya que se crea una copia de todos los datos en un mismo emplazamiento. Los datawarehouse pueden ser vistos como un caso de replicación asíncrona, pero con una actualización poco frecuente. [AGRAWAL 2002].

La figura 1 muestra un ejemplo de arquitectura de bases de datos distribuida, extraída de [ELMASRI 2000], en la cual existen 5 nodos conectados por una red, y uno de ellos (Chicago) es el nodo central. Chicago contiene replicada toda la base de datos, que consiste en las tablas de EMPLEADO, PROYECTOS y TRABAJA_EN. Los otros

nodos contienen los fragmentos de las tablas anteriores correspondientes a los datos de sus ubicaciones:

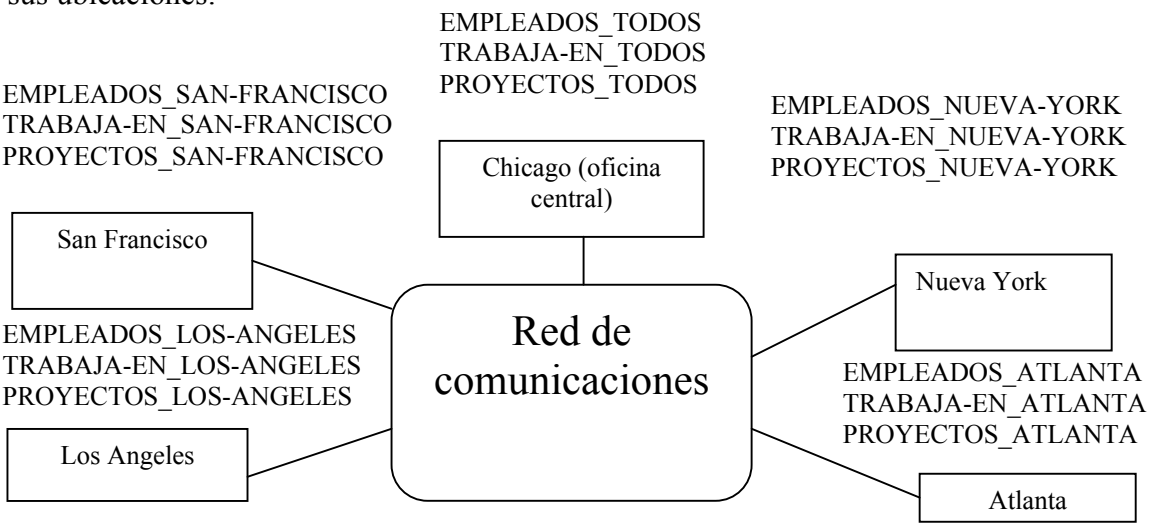


Figura 1. Arquitectura de bases de datos distribuida

También conviene entender la diferencia entre SGBD distribuido y SGBD paralelo definido como "un SGBD que se ejecuta sobre múltiples procesadores y utilizando múltiples discos y que está diseñado para ejecutar las operaciones en paralelo", [CONOLLY 2005]. Está basada en un sistema de compartición de recursos, dando lugar a arquitecturas paralelas de memoria compartida, de disco compartido, jerárquicas (híbridas) ó sin compartición, según sea el recurso que se esté compartiendo. La figura 2 muestra un ejemplo de arquitectura de bases de datos paralela, con disco compartido, extraído de [SILBERSCHATZ 2002]:

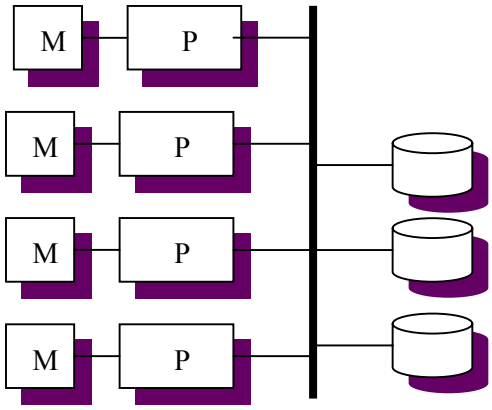


Figura 2. Arquitectura de bases de datos paralela de disco compartido

A veces se confunde la arquitectura de base de datos paralela sin compartición con la distribuida. Los factores que diferencian a esta última son:

1. Las bases de datos distribuidas suelen estar en lugares geográficamente distintos.
2. Se administran de forma separada.
3. Poseen una conexión más lenta.
4. Se dan dos tipos de transacciones (locales, globales).

Este trabajo pretende demostrar cómo se puede desarrollar un SGBDD con transparencia de fragmentación, de replicación y de consultas, esto es, el usuario final debe poder usar la BDD como si fuera centralizada. La sección 2 describe el estado actual de los SGBDD, la sección 3 plantea el problema, la sección 4 describe la resolución del problema, la sección 5 plantea las conclusiones y líneas futuras y por último, la sección 6 enumera la bibliografía.

2 Estado actual de los SGBDD

Existen varios productos comerciales con bases de datos distribuídas, como:

- INGRES/STAR de Relational Technology, Inc.
- SQL*STAR, de Oracle Corp.
- DB2 versión 2 Edición 2, de IBM.

El factor más importante que diferencia a unos SGBDD de otros es la transparencia. Esto supone que no se requiere soporte para la manipulación de datos (transparente para el usuario) pero si se requieren operaciones de definición de datos (FRAGMENT, REPLICATE).

Desde el punto de vista académico los dos tipos de software conocidos son:

- Microsoft Access, con un SGBD muy sencillo porque da soporte a una base de datos de oficina muy poco sofisticada. No proporciona funcionalidad de BDD, pero se pueden implementar módulos que simulen la fragmentación, replicación y consultas distribuidas. Sin embargo, no se garantiza la fiabilidad del SGBDD desarrollado.
- MySQL, no proporciona actualmente soporte alguno para los BDD verdaderos.
- Oracle9i, (KOCH, G., 2003); no ofrece utilidades para fragmentar, replicar y realizar consultas optimizadas, pero se la ha elegido como plataforma para desarrollar la transparencia debido a sus prestaciones como SGBD.

Existe otro elemento fundamental, que es el catálogo. El catálogo en una base de datos centralizada proporciona la información relativa a las relaciones, vistas e índices se almacena en un CATALOGO o DICCIONARIO DE DATOS. Por guardar información descriptiva sobre la base de datos, también es referido como metainformación. El catálogo se usa sobretodo para optimizar consultas. En una base de datos distribuida el catálogo se usa también para hacer un seguimiento de los datos distribuidos en varios sitios. Ese seguimiento debe tener en cuenta cómo han sido fragmentadas y replicadas las relaciones. Existen varios enfoques sobre donde guardar este catálogo, destacando el enfoque R*, versión distribuida de System R de IBM en la década de los 80, [SIMKOVICS 1998], en el cual existe en cada nodo un puntero para cada fragmento que se almacenó inicialmente allí, más un puntero por cada fragmento almacenado actualmente en dicho nodo.

3 . Planteamiento del problema

Dado que el núcleo fundamental es desarrollar de manera automática las principales características de los SGBDD, se procede a la definición de los requisitos:

1. Fragmentación de la base de datos original: La fragmentación debe poder ser mixta y derivada con asignación de los fragmentos a distintos nodos. La fragmentación mixta es una combinación de horizontal y vertical. La fragmentación horizontal se aplica cuando se extraen fragmentos por selección de tuplas que cumplan una determinada condición. Normalmente se usan los valores de un(os) atributo(s) para realizar dicha selección. La fragmentación vertical se aplica cuando se extraen fragmentos por selección de atributos de todas las tuplas. Por tanto, la fragmentación mixta se aplica a la selección de varios atributos de las tuplas que cumplan una determinada condición. La fragmentación derivada es aquella que afecta a más de una tabla porque el criterio de fragmentación de una tabla afecta a atributo(s) que actúa(n) como clave extranjera en otra(s). Esto se realizará una sola vez por un usuario con privilegios de administrador de base de datos.

2. Replicación de fragmentos de la base de datos: Se debe poder indicar los fragmentos que se replican y donde se replican. Además, se debe implementar la funcionalidad de un SGBDD que realice la actualización de los fragmentos replicados cuando el original se actualice. Esto se realizará una sola vez por un usuario con privilegios de administrador de base de datos.

3. Consultas optimizadas, en la cual se puedan realizar tanto consultas locales como consulta globales a varios nodos, y que por tanto, involucran a tablas de varios

nodos. Esto se puede realizar después de la fragmentación y replicación cualquier número de veces por un usuario corriente. El SGBDD debe poder realizar consultas optimizadas, es decir, una vez introducidos los parámetros, debe localizar la información en los sitios más cercanos. Esto es más complicado si existen fragmentos replicados, porque debe localizar la manera óptima de obtener datos mediante descomposición de las consultas.

En la implementación de estos requisitos, el catálogo va a tener un rol fundamental, puesto que va a proporcionar un seguimiento los fragmentos en cada momento, además de indicar su procedencia (de qué tabla de la base de datos original procede).

4. Resolución del problema

4.1 Configuración del Oracle

La máquinas deben tener instalado el Oracle9i, en modo dedicado. Crear una base de datos en cada una de ellas. Estas máquinas van a representar los nodos de una BDD.

4.2 Conectividad entre todas las máquinas

Se debe configurar la parte servidora de cada uno de los nodos (este paso va a indicar qué parte de un nodo va a ser accesible al resto). En la figura 3 aparece la opción del Oracle Enterprise Manager Console:

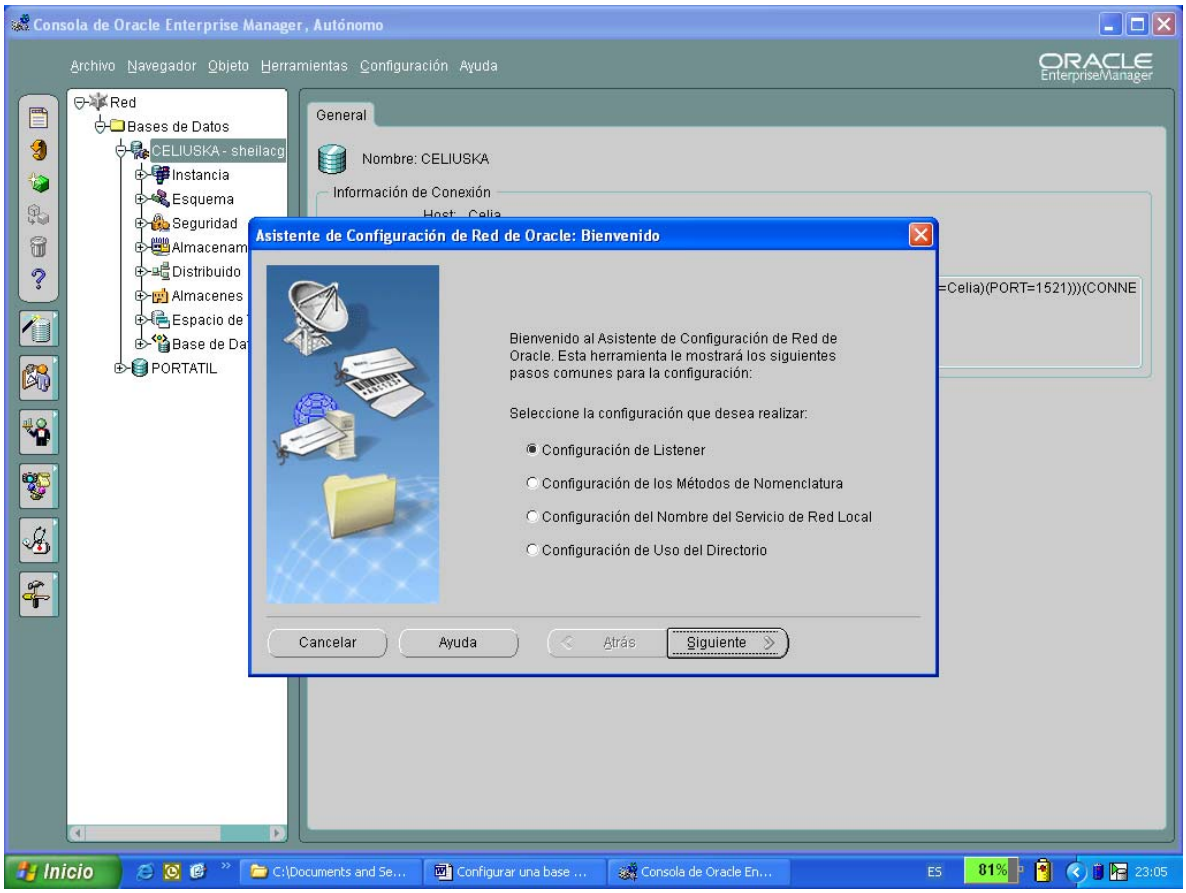


Figura 3. Configuración de la parte servidora

Se debe configurar la parte cliente de cada uno de los nodos (este paso va a indicar a qué partes servidoras de otros nodos se va a conectar). En la figura 4 aparece la opción del Oracle Enterprise Manager Console:

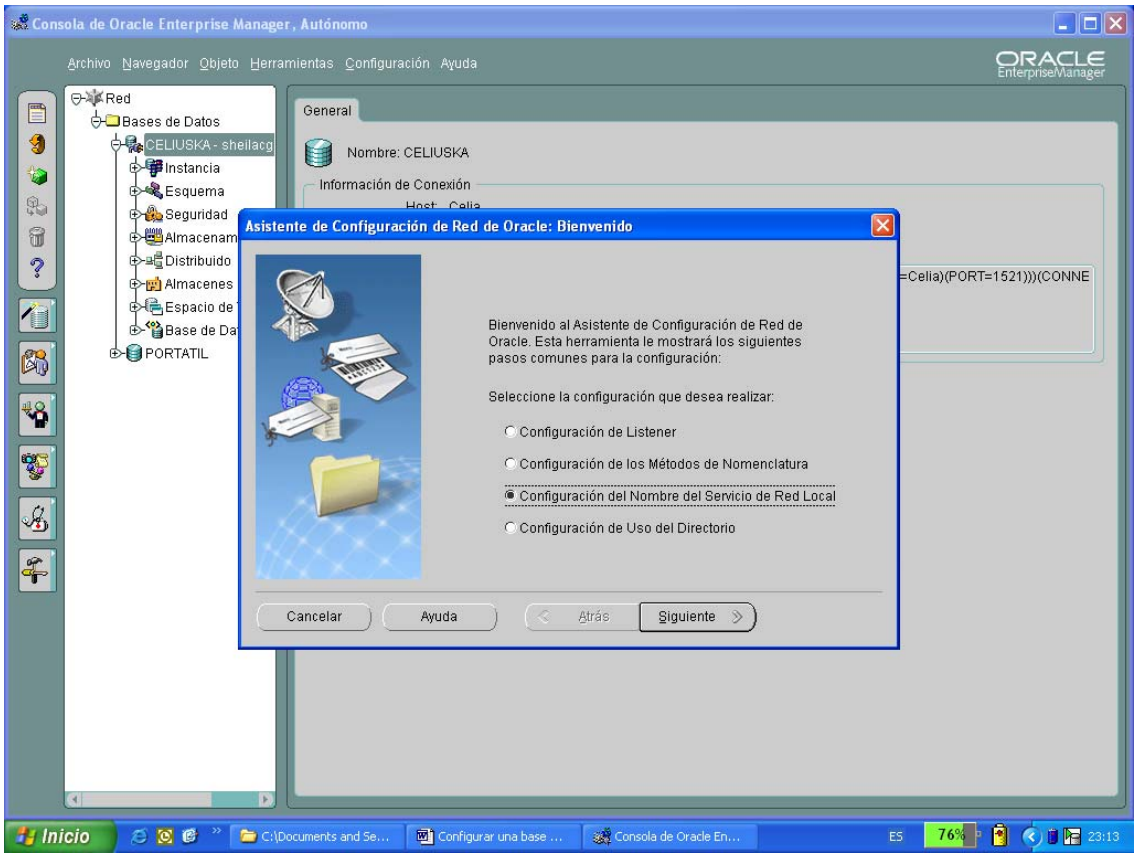


Figura 4. Configuración de la parte cliente

Para terminar se realizará una prueba de conexión como se puede ver en la figura 5:

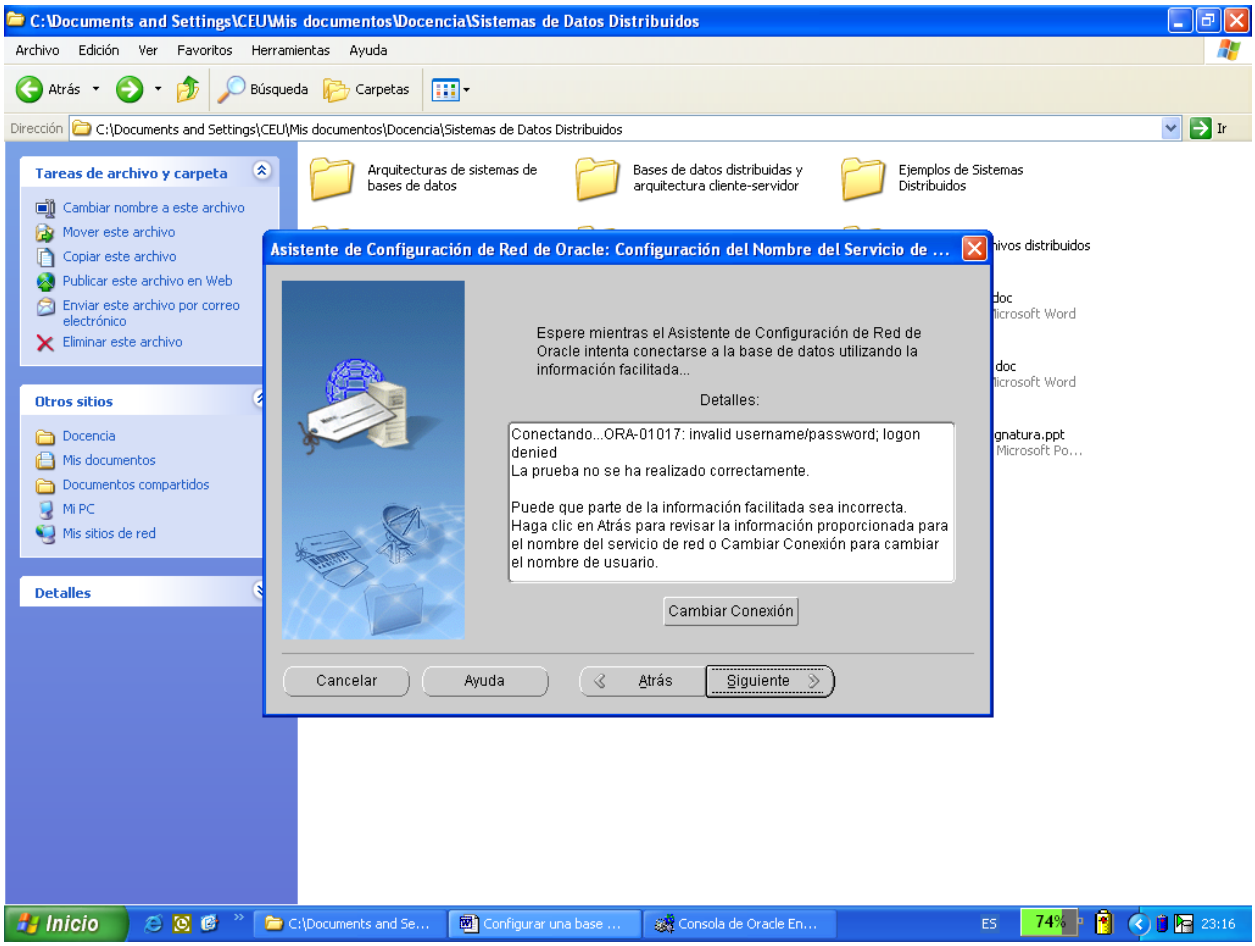


Figura 5. Prueba de conexión fallida

4.3 Creación de enlaces de la base de datos ó dblinks

La figura 6 muestra este paso. Los dblinks proporcionan un enlace a una base de datos remota, de tal manera que se puedan ejecutar sentencias SQL desde un nodo sobre otro distinto.

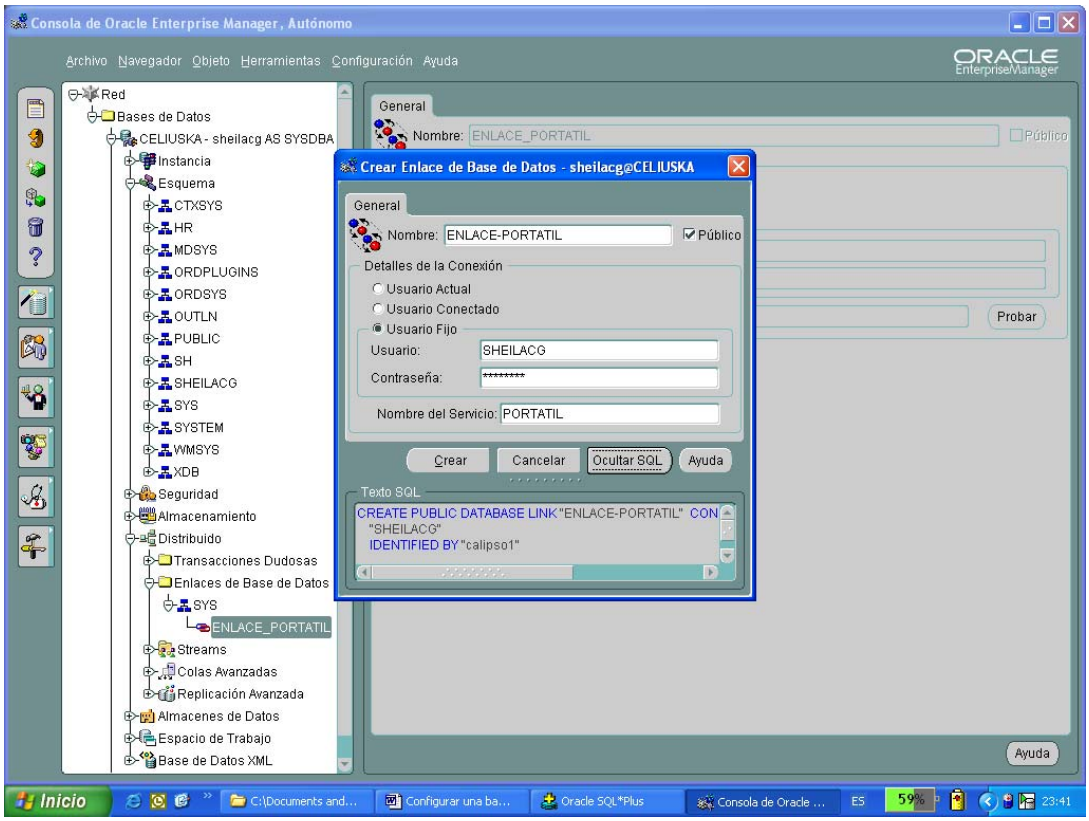


Figura 6. Creación de dblinks

4.4 Creación de transparencia de Fragmentación

La sentencia SQL para crear fragmentos sigue la siguiente sintaxis:

CREATE TABLE tabla AS SELECT * FROM nom-esquema.nom-tabla@dblink

que creará un fragmento de nombre tabla con la selección de todas las tuplas de un esquema llamado nom-esquema dentro una tabla llamada nom-tabla accediendo a una base de datos remota a través de un enlace llamado dblink.
Otra manera más sintética es usando sinónimos:

CREATE SYNONYM nom-sinonimo FOR nom-esquema.nom-tabla@dblink

Esta sentencia creará el sinónimo y sustituye a la sentencia anterior por:
CREATE TABLE tabla AS SELECT * FROM nom-sinonimo

Esta sentencia debe ser Introducida en el nodo que se desea que se almacene el fragmento. Para crear la transparencia de fragmentación se debe crear un procedimiento almacenado, [ALI 2004]. Un procedimiento almacenado es un conjunto de instrucciones en PL/SQL, que pueden ser llamado usando el nombre que se le haya asignado.

La sintaxis para crear un procedimiento es la siguiente:
CREATE [OR REPLACE] PROCEDURE name [(param [IN|OUT|IN OUT|] datatype) . . .][IS|AS] pl/sql_subprogram

En nuestro caso se le proporcionan los parámetros:

1. La condición de selección de las tuplas.
2. Los atributos de selección.
3. El nombre del esquema, tabla y dblink.
4. Nombre del fragmento de destino.
5. El sitio donde se quiere que se almacene inicialmente.

La creación de un fragmento debe producir una actualización en el catálogo de la BDD. El catálogo es una tabla del SGBDD que contiene tantas filas como fragmentos residen en la BDD, y formada por los siguientes atributos (se va tomar como ejemplo el fragmento EMPLEADOS_SAN-FRANCISCO):

1. Tabla de origen del fragmento: EMPLEADOS
2. Lista de atributos: * (enumeración de los atributos de la tabla de origen que forman parte de este fragmento; es el reflejo de la fragmentación vertical).
3. Condición de guarda: ciudad = SAN-FRANCISCO (condición booleana que sirve para seleccionar las tuplas de la tabla origen que forman parte de este fragmento; es el reflejo de la fragmentación horizontal).
4. Tabla de destino: EMPLEADOS_SAN-FRANCISCO
5. Nodo donde se va a guardar inicialmente: San Francisco

Nótese que los argumentos del procedimiento almacenado no solo van a servir para crear el fragmento, sino también para actualizar el catálogo.

4.5 Creación de transparencia de replicación

La sentencia SQL para replicar fragmentos es la misma que el paso previo:
CREATE TABLE tabla AS SELECT * FROM nom-esquema.nom-tabla@dblink ó

CREATE TABLE tabla AS SELECT * FROM nom-sinonimo

La primera sentencia sin sinónimos, la segunda con sinónimos.
Al igual que en el caso anterior, la replicación se producirá a través de la llamada a un procedimiento almacenado con el paso de parámetros:

1. Fragmento de origen
2. Fragmento de destino
3. Nodo(s) donde se va a guardar

La creación de un fragmento replicado actualizará el catálogo distribuido, como en la creación de cualquier otro fragmento, donde la lista de atributos es * y la condición de guarda es simplemente TRUE.

La complejidad de la replicación está en la actualización de uno de los fragmentos replicados, ya que el resto también debe reflejar el cambio. Se ha elegido el modo síncrono de replicación por facilidad en la implementación, que como se dice en [RAMAKRISHNAN 2003], las actualizaciones de un fragmento se ven reflejadas inmediatamente en sus réplicas. Para implementar este tipo de replicación se van a usar los triggers ó disparadores [WENTE 1983], que son procedimientos que se ejecutan cuando ocurre un evento. En la implementación de un trigger se deben especificar las acciones que se van a realizar y opcionalmente las condiciones. La sintaxis de un trigger en Oracle es:

CREATE [OR REPLACE] TRIGGER
{BEFORE|AFTER} {DELETE|INSERT|UPDATE [OF col1, col2, . . ., colN]
[OR {DELETE|INSERT|UPDATE [OF col1, col2, . . ., colN]. . .]}
ON table
[REFERENCING OLD AS oldname, NEW as newname]
[FOR EACH ROW [WHEN (condition)]]
pl/sql_block

Existirán tantos triggers como acciones puede haber sobre una base de datos:

1. Borrado de registros en un fragmento replicado: las acciones asociadas implican el borrado de los registros correspondientes en sus réplicas.
2. Actualización de registros en un fragmento replicado: las acciones asociadas implican la actualización de los registros correspondientes en sus réplicas.
3. Inserción de registros en un fragmento replicado: las acciones asociadas implican la inserción de los registros correspondientes en sus réplicas.

Para llevar a cabo la las acciones asociadas se deben buscar las réplicas a través del catálogo.

4.6 Creación de transparencia en Consultas distribuidas

Para realizar una consulta distribuída sencilla (que involucra una sola tabla), la sintaxis es la siguiente:

SELECT * FROM nom-esquema.nom-tabla@dblink ó

SELECT * FROM nom-sinonimo

No existe transparencia de consultas, ya que es el propio usuario el que ha de introducir donde residen los fragmentos a los que quiere acceder, con los nombres de los fragmentos. Para implementar la transparencia en las consultas, el usuario debe introducir la sentencia como si fuera una base de datos centralizada.

En nuestra BDD (figura 1), se introduce la sentencia SELECT nombre FROM EMPLEADOS WHERE salario=10000 desde el nodo NUEVA-YORK.
Esto se puede sustituir por la llamada a un procedimiento almacenado, cuyos parámetros van a ser:

1. Los atributos que se buscan: NOMBRE
2. Tabla de origen: EMPLEADOS
3. Las condiciones de búsqueda: ciudad=SAN-FRANCISCO and salario=10000
4. Nodo desde el cual se inicia la consulta: Nueva York

Nuevamente se busca en el catálogo los fragmentos que cunplan estos requisitos, y se encuentran dos posibilidades:

1. Opción A:Fragmento EMPLEADOS_SAN-FRANCISCO, que reside en el nodo San Francisco.
2. Opción B: Fragmento EMPLEADOS_TODOS, que reside en el nodo central.

Un SGBDD con consultas optimizadas, realizará una evaluación de las posibles soluciones a una consulta, de tal manera que ejecutará la más eficiente, midiendo la eficiencia en términos de bytes transferidos, [ELMASRI 2000]. Teniendo en cuenta que se seleccionan únicamente los datos que interesan de los fragmentos, en este caso los bytes que viajan en los dos casos son iguales, así que se optará por cualquiera de las dos:

N_A = tamaño atributo nombre en EMPLEADOS_SAN-FRANCISCO
M_A= número de tuplas que contengan salario=10000 en EMPLEADOS_SAN-FRANCISCO
N_B = tamaño atributo nombre en EMPLEADOS_TODOS
M_B= número de tuplas que contengan salario=10000 en EMPLEADOS_TODOS

Como el fragmento EMPLEADOS_TODOS, se ha formado por la unión de los fragmentos de todos los nodos, el tamaño de los atributos coincide con el de los fragmentos de los nodos. El número de tuplas que cumplen una condición basada en el salario y la ciudad al mismo tiempo es el mismo que el del fragmento origen. Luego $N_A = N_B$ y $M_A = M_B$ en este caso. Bajo las fórmulas (1) y (2) se calculan los datos que viajan para cada caso:

Bytes que viajan (opción A)= $N_A * M_A$ (1)
Bytes que viajan (opción B)= $N_B * M_B$ (2)

Una variación al resultado de esta consulta sería que el nodo desde el cual fué introducida fue el nodo central. Las fórmulas (3) y (4) calculan los datos que viajan:

Bytes que viajan (opción A)= $N_A * M_A$ (3)
Bytes que viajan (opción B)= 0 (4)

Luego el SGBDD optará por la opción B.

5. Conclusiones y líneas futuras

Como se ha podido ver, aunque no todas los SGBDD actuales ofrecen transparencia, existen maneras de construirlas. En este trabajo se ha elegido la plataforma Oracle9i, y aprovechando los recursos que esta ofrece, se ha desarrollado código fuente para añadirle al SGBD Oracle9i la funcionalidad de transparencia.

Este trabajo ofrece otras posibilidades de ampliación y mejora, como son el desarrollo de consultas optimizadas anidadas (que involucran a varios fragmentos) y la implantación del SGBDD en un servidor web accesible a otros usuarios, de tal manera que todos ellos puedan fragmentar, replicar y realizar consultas distribuidas con solo llamar al procedimiento del servidor con los parámetros adecuados. También se pueden diseñar catálogos distribuidos más optimizados, de tal manera que se mejore la eficiencia en las consultas.

6. Bibliografía

AGRAWAL D., EL ABBADIA., MOSTEFAQUIA A., RAYNAL M., ROY M. y MALKHI DAHLIA (2002). *"The lord of the rings: Efficient maintenance of views at data warehouse"*. Proceedings of Distributed Computing: 16th International Conference, Volume 2508/2002, 33-47. Lecture Notes in Computer Science. Springer Berlin/Heidelberg,

ALI A.R., HARB H.M. (2004). *"Two phase locking concurrency control in distributed database with N-tier architecture"*. International Conference on Electrical, Electronic and Computer Engineering, 2004. 149- 153. IEEE.

CONOLLY T.M., BEGG C.E. (2005). *"Sistemas de bases de datos"*. Pearson Education. Madrid.

DATE C. J. (2000). *"Introducción a los Sistemas de Bases de Datos"*. Séptima Edición. Ed. Prentice Hall.

ELMASRI R. (2000). *"Fundamentos de Sistemas de bases de datos"*. Ed. Addison Wesley.

KOCH G., LONEY K. (2003). "*Oracle9i. Manual de Referencia*". Mc Graw-Hill.

RAMAKRISHNAN R., GEHRKE J. (2003). "*Database Management Systems"*. Mc-Graw-Hill Professional.

SILBERSCHATZ A. (2002). *"Fundamentos de Bases de Datos*". Mc Graw Hill.

SIMKOVICS S. (1998). *"Enhancement of the ANSI SQL Implementation of PostgreSQL* ". Department of Information Systems, Vienna University of Technology. http://es.tldp.org/Postgresql-es/web/navegable/tutorial/sql.html.

WEN-TE K. LIN, DANIEL R. RIES, BARBARA T. BLAUSTEIN, R. MARK CHILENSKAS (1983). "*Office procedures as a distributed database application*", ACM SIGMIS Database, vol. 15 (2), 5-10. ACM Press. New York, NY, USA.