

BBDD distribuidas

Contenidos

- ☞ BBDD y SGBD distribuidos
- ☞ Componentes de una BBDD distribuida
- ☞ Técnicas de fragmentación
- ☞ Consultas distribuidas
- ☞ Transacciones distribuidas
- ☞ Optimización de consultas sobre bases de datos distribuidas

Objetivos

- ☞ Reconocer la utilidad de las bases de datos distribuidas
- ☞ Conocer cómo realizar consultas
- ☞ Conocer el funcionamiento de las transacciones distribuidas
- ☞ Describir las distintas políticas de fragmentación de la información

En este capítulo se explican las peculiaridades de las consultas en bases de datos distribuidas, se realizan operaciones básicas para que el alumno aprecie la complejidad de este tipo de sistemas y al mismo tiempo su sencilla implantación, gracias a las propias herramientas proporcionadas por los SGBD.

7.1. BBDD y SGBD distribuidos

Cuando una base de datos está controlada por más de un servidor se dice que está distribuida. Estos servidores están interconectados mediante una red de telecomunicaciones y los SGBD proporcionan mecanismos para poder consultar la información independientemente del servidor y la ubicación de los datos. Así por ejemplo, una consulta con una join entre la tabla de pedidos y la de clientes puede estar accediendo a dos servidores con dos instancias de bases de datos distintas. Incluso, si las tablas están particionadas es posible que la misma consulta esté accediendo a más de dos servidores. Esto proporciona una especie de balanceo de carga para que la información pueda ser accedida lo más rápidamente posible. De esta manera se consiguen los siguientes beneficios:

1. Más rapidez en el acceso a los datos y capacidad de almacenamiento. Al aumentar el número de ordenadores donde se depositan los datos, aumenta la capacidad de computación, y por tanto, la capacidad de búsqueda de la información.
2. Acceso más flexible. El número de usuarios conectados que puede mantener el sistema se incrementa al haber más capacidad de cómputo y, gracias a la red de la BBDD distribuida, los usuarios pueden acceder desde más ubicaciones.
3. Escalabilidad. Distribuyendo una base de datos en distintas ubicaciones se permite incrementar de forma muy sencilla los recursos en cualquier momento para acomodar el sistema a las demandas de los usuarios y las aplicaciones.
4. Fiabilidad y cierta capacidad de tolerancia a fallos. Realizando una buena política de fragmentación de la base de datos es posible conseguir que aunque algún nodo de la base de datos distribuida sufra algún percance, el resto del sistema siga funcionando con normalidad.

Por otro lado, no todo son beneficios, no hay que perder de vista que estos sistemas son muy costosos de implementar no solo por la duplicidad de recursos en términos de red y servidores, también por la dificultad de la administración de los datos ubicados en distintas localizaciones y porque se introduce un elemento caótico como la dependencia de las redes de comunicaciones para el correcto funcionamiento del intercambio de los datos. Además, la atomicidad de las operaciones es más complicada de conseguir por lo que la gestión transaccional tiene dificultades añadidas como se comentará en la sección 7.3.2.

7.1.1. Componentes de una BBDD distribuida

Para poder crear una base de datos distribuida hay que tener los siguientes componentes.

1. Sistemas gestores con capacidades distribuidas o un SGBD distribuido en sí mismo.
2. Un conjunto de bases de datos locales a cada uno de los servidores que albergarán la BBDD.
3. Una red de comunicaciones, generalmente basada en el protocolo TCP-IP para poder aprovechar las infraestructuras existentes.
4. Enlaces entre las BBDD locales. Los SGBD deben ser capaces de establecer enlaces entre las diversas bases de datos locales, de tal manera que cualquier consulta pueda ser redirigida al servidor donde está la BBDD local.
5. Un diccionario de datos global que indique en qué ubicación está cada uno de los datos que componen la BBDD distribuida.

BASE DE DATOS DISTRIBUIDA

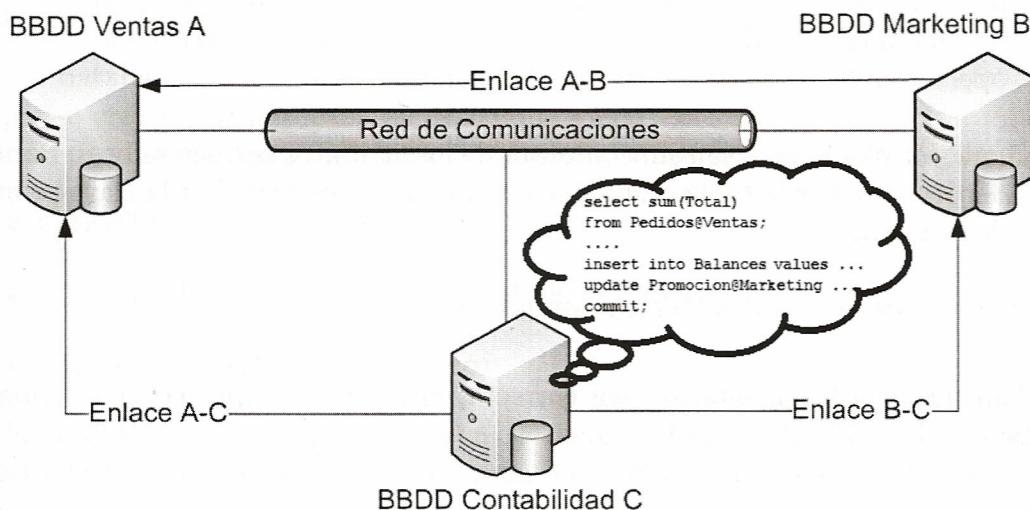


Figura 7.1: Componentes de una base de datos distribuida.

Dependiendo del tipo de componentes elegidos en la formación de una base de datos distribuida se pueden tener:

1. BBDD distribuidas homogéneas: En estas el SGBD utilizado es el mismo para todas las BBDD locales. Generalmente, el SGBD proporciona la tecnología para el enlace de las distintas BBDD locales.
2. BBDD distribuidas heterogéneas: Las bases de datos están distribuidas independientemente del SGBD utilizado como local. En este caso, las aplicaciones

que consultan la base de datos distribuida juegan un papel fundamental como mediador.

7.2. Técnicas de fragmentación

El contenido de una tabla puede separarse en varios fragmentos que contendrán suficiente información como para poder reconstruir la tabla original en caso de que sea necesario. Cada fragmento se encontrará en un nodo diferente, pero esta distribución a través de distintos nodos no duplica los registros, solo existirá una copia de cada elemento, por lo que, al no haber replicación, disminuye el coste de almacenamiento en detrimento de la disponibilidad y fiabilidad de los datos en caso de caída de algún nodo.

La fragmentación puede ser horizontal o vertical y se puede aplicar sucesiva y alternativamente sobre la misma tabla:

Horizontal: los fragmentos son subconjuntos de una tabla y se definen a través de una operación de selección. La tabla original se reconstruye en base a una operación de unión de los fragmentos componentes.

Vertical: los fragmentos son subconjuntos de los atributos con sus valores. Para poder recomponer la tabla original, cada fragmento debe incluir la clave primaria de la tabla.

Mixta: se mezclan las dos técnicas anteriores.

Para que una fragmentación sea correcta, esta debe cumplir con las siguientes reglas:

- Debe ser completa, es decir, cada elemento de la tabla debe estar en algún fragmento.
- Debe ser reconstruible, lo cual implica que debe ser posible conseguir una tabla completa a partir de la combinación de todos los fragmentos.
- Los fragmentos deben ser disjuntos, esto significa que si la fragmentación es horizontal, un elemento que esté en un fragmento concreto no puede estar en ningún otro fragmento más que en aquél. En el caso de fragmentación vertical, puesto que es necesario que se repitan las claves primarias, esta condición solo se tiene que cumplir para el conjunto de atributos que no son clave primaria.

7.3. Consultas distribuidas

Algunos SGBD manejan el concepto de sistemas de bases de datos distribuidas de forma que cualquier aplicación que haga consultas al gestor pueda acceder a su base de datos de forma local y además, acceder a datos de bases de datos remotas.

Oracle basa su concepto de bases de datos distribuidas en un objeto llamado *DATABASE LINK* (Enlace a base de datos). Los DATABASE LINKs son una conexión unidireccional entre una base de datos (cliente) y otra (servidor). Oracle puede soportar bases de datos homogéneas y heterogéneas. Para soportar sistemas heterogéneos es necesario que Oracle pueda acceder a los datos como si estuviese accediendo a otra base de datos Oracle, por lo que cuenta con agentes llamados *Transparent Gateways* (Pasarelas transparentes). Estos agentes son interfaces que proveen todo lo necesario para poder recuperar, insertar, actualizar o borrar información de forma *transparente* para las aplicaciones que se ejecutan desde Oracle.

7.3.1. DB Links

Un database link es una conexión entre bases de datos. Es unidireccional por lo que se crea desde una base de datos que quiere referenciar a otra. Para que un usuario pueda crear estos DB Links son necesarios los siguientes privilegios:

- CREATE DATABASE LINK (En local).
- CREATE PUBLIC DATABASE LINK (En local).
- CREATE SESSION (En remoto).

Es posible consultar los DB Links en las vistas USER_DB_LINKS, ALL_DB_LINKS y DBA_DB_LINKS dependiendo del usuario con el que se esté conectado al SGBD.

Existen también los enlaces compartidos o *SHARED DB LINKS*. La diferencia es que múltiples procesos clientes pueden usar el enlace compartido simultáneamente. Para crear un DB link es necesario que exista la conexión entre las bases de datos a través de Oracle Net Services, por lo que se debe crear una entrada en el tnsnames.ora de la base de datos a la que se quiere referenciar. Dentro de la definición se puede elegir si la conexión es dedicada o compartida, y no es lo mismo definir que la conexión que va a usar el DB link sea compartida a crear un DB link compartido. A continuación, se muestra un ejemplo de creación de un DB link para conectar una BBDD en versión 10gR2 con una BBDD 11gR2. La entrada en el fichero tnsnames.ora es:

```
t11r2_u =
(DESCRIPTION =
  (ADDRESS = (PROTOCOL = TCP)(HOST = 192.168.1.129)(PORT = 1521))
  (CONNECT_DATA =
    (SERVER = DEDICATED)
    (SID = t11r2)
  )
)
```

t11r2_u en este caso, es el nombre de la entrada para la conexión remota al servidor con dirección IP *192.168.1.129*.

Una vez agregada la entrada al fichero se puede utilizar la herramienta tnsping para comprobar que hay conectividad:

```
C:\Documents and Settings\usuario>tnsping t11r2_u

TNS Ping Utility for 32-bit Windows: Version 10.2.0.4.0 - Production on 05-MAY-2011 23:23:47

Copyright (c) 1997, 2007, Oracle. All rights reserved.

Archivos de parámetros utilizados:
C:\oracle\product\10.2.0\db_1\network\admin\sqlnet.ora

Adaptador TNSNAMES utilizado para resolver el alias
Attempting to contact (DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(HOST = 192.168.1.129)
(PORT = 1521)) (CONNECT_DATA = (SERVER = DEDICATED) (SID = t11r2)))
Realizado correctamente (40 mseg)
```

Una vez creada la entrada del fichero tnsnames y probada, se pueden crear DB links públicos o privados, y es posible, además, usar un usuario específico para la conexión en la base de datos a referenciar. Para crear un DB Link desde el usuario local al usuario remoto, se ejecutaría el siguiente comando:

```
SQL> CREATE DATABASE LINK tj_11r2
  2  CONNECT TO usuario IDENTIFIED BY password
  3  USING 't11r2_u';

Enlace con la base de datos creado.
```

Al ejecutar el comando anterior, se crea un objeto llamado *tj_11r2* que usará la entrada tnsnames *t11r2_u* para conectarse a la BBDD remota. Después, para probar la conexión, se utiliza el nombre del enlace (*tj_11r2*) para ejecutar una consulta:

```
SQL> select * from dual@tj_11r2;
```

```
D  
-  
X
```

También se pueden crear DB Links públicos que pueden ser usados por cualquier usuario, y además, sean usados por usuarios que también existan en la BBDD que se referencia:

```
SQL> CREATE PUBLIC DATABASE LINK tj_11r2_2 USING 't11r2_u';  
Enlace con la base de datos creado.
```

7.3.2. Ejecución de consultas distribuidas

Las consultas distribuidas son consultas que recuperan información de 2 o más nodos en un entorno de bases de datos distribuido. No es lo mismo que hablar de consultas remotas, las consultas remotas son consultas que se realizan a una o más tablas de una base de datos remota. Es importante diferenciar entre estos dos tipos de consultas las distribuidas y remotas porque internamente son diferentes.

Un ejemplo de una consulta remota es el siguiente:

```
SQL> select count(8)  
  2   from pacientes@t11r2;  
  
COUNT(8)  
-----  
      100
```

T11R2 es un DB LINK público creado a una BBDD 11gR2 en Linux, y la base de datos cliente está en 10gR2 en Windows. La ejecución de la consulta anterior prueba que existen conexión entre versiones diferentes. Esto se logra a través de Oracle Net Services.

En una consulta distribuida puede darse el caso de que existan relaciones fragmentadas. Esto se ilustra en el ejemplo siguiente:

Primero se muestra la tabla PACIENTES, fragmentada en dos bases de datos en distintos servidores y con la siguiente descripción:

```
SQL> desc pacientes
Nombre          Nullable?  Tipo
-----
ID              NOT NULL NUMBER
NOMBRE          VARCHAR2(100)
HABITACION      NUMBER

SQL> desc pacientes@t11r2
Nombre          Nullable?  Tipo
-----
ID              NOT NULL NUMBER
NOMBRE          VARCHAR2(100)
FECHA_ENTRADA  DATE
```

Ahora, se consulta un paciente con ID 1, su habitación y fecha de ingreso:

```
SQL> select l.id, l.habitacion, r.fecha_entrada
  2  from pacientes l,
  3      pacientes@t11r2 r
  4  where l.id=r.id
  5  and l.id=1
  6 /

ID HABITACION FECHA_EN
-----
1          10 26/04/11
```

Esta consulta extrae información de dos bases de datos ubicadas en servidores distintos, por tanto, es distribuida. Además, desde la base de datos servidor se puede ver lo siguiente en la vista V\$SESSION:

```
SQL> select sid, serial#, username, program, event, machine , server
  2  from v$session
  3  where username='TJ';

SID SERIAL# USERNAME PROGRAM      EVENT                MACHINE    SERVER
-----
29     48 TJ      ORACLE.EXE   SQL*Net message from client  VILLEGUILLO DEDICATED
```

Esto demuestra que cada vez que se realice una consulta a una base de datos remota, se abre una sesión con server process dedicado si se define en el tnsnames.ora que sea dedicada. Dejar las conexiones abiertas puede llegar a ser un problema: si se va a crear un proceso que lanza cientos de consultas y que va a muchos nodos en un

entorno de bases de datos distribuido, y que además este proceso lo van a ejecutar muchas sesiones, en diferentes nodos, es seguro que se generará una explosión de conexiones abiertas, incrementando de forma radical el consumo de recursos.

Para evitar esta explosión de conexiones abiertas, se puede ejecutar el cierre de los DB Links ejecutando:

```
SQL> alter session close database link t11r2;
Sesión modificada.

--Esto cierra la sesión en la base de datos servidor:
SQL> select sid, serial#, username, program, event, machine, server
  2  from v$session
  3* where username='TJ'

no rows selected
```

7.4. Transacciones distribuidas

Una transacción distribuida es una transacción que contiene una o más sentencias y que se realiza en 2 o más nodos dentro de un entorno de base de datos distribuido.

También existen las transacciones remotas, y al igual que las consultas remotas, son transacciones con una o más sentencias que se realizan en una base de datos remota.

Los elementos que pueden intervenir a tener en cuenta en una transacción distribuida se pueden ver en la siguiente figura:

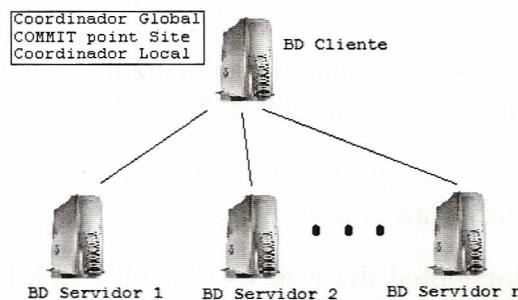


Figura 7.2: Transacción distribuida.

En esta figura se observan los siguientes elementos:

BD Cliente Es un nodo que actúa como cliente donde se referencia la transacción distribuida. Esta BD tiene los DB links que conectan con el resto.

BD Servidor Son las bases de datos que contienen información que va a ser modificada por la transacción distribuida.

Coordinador Local Es un nodo que debe referenciar datos sobre otros nodos para completar su parte de la transacción distribuida.

Coordinador Global El nodo donde se origina la transacción distribuida.

Commit Point Site Es el nodo que inicia las operaciones de COMMIT o ROLLBACK con las instrucciones dadas por el coordinador global. Este punto se define con el parámetro COMMIT_POINT_STRENGTH, en un entorno homogéneo todas las BBDD tienen un parámetro como este. La base de datos con el valor mayor en este parámetro se convierte en COMMIT POINT SITE.

7.4.1. TWO-PHASE COMMIT

Un SGBD debe garantizar la atomicidad en la información de sus bases de datos y en los entornos con bases de datos distribuidas también. Para que esto se produzca, se usa el mecanismo de TWO-PHASE COMMIT o proceso de COMMIT en entornos distribuidos. Su ejecución es un poco más complicada. Se divide en 3 partes:

Fase de preparación Es la primera fase y consiste en que el coordinador global informa a todos los nodos que van a realizar COMMIT o ROLLBACK. Esto se traduce en que cada nodo se *prepara* para una finalización de la transacción distribuida. Para ello realiza 2 tareas:

1. Almacena la información de Redologs.
2. Genera un bloqueo distribuido sobre la tabla para prevenir lecturas.

Cuando se realizan estas 2 operaciones el nodo dice que está *comprometido* para la finalización de la transacción distribuida.

Fase de commit Después de que todos los nodos se comprometen a realizar la transacción, se realizan los siguientes pasos:

1. El coordinador global avisa al COMMIT POINT SITE de que se va a realizar el COMMIT.
2. El COMMIT POINT SITE realiza el COMMIT.
3. El COMMIT POINT SITE indica al coordinador global que hizo el COMMIT.
4. Los coordinadores globales y locales envían mensajes a todos los nodos para hacer COMMIT de la transacción.

5. Cada nodo realiza su parte local del COMMIT de la transacción distribuida y quitan el bloqueo distribuido generado.
6. Todos los nodos notifican al coordinador global que han realizado el COMMIT.

Fase de olvido Cuando todos notifican al COMMIT POINT SITE que se ha realizado el COMMIT, se realizan las siguientes tareas:

1. El COMMIT POINT SITE borra la información acerca del estado de la transacción distribuida.
2. El COMMIT POINT SITE informa al coordinador global que ha borrado ese estado.
3. El coordinador global borra la información de la transacción distribuida.

7.4.2. Ejemplo de transacción distribuida

Con la tabla pacientes ubicada de forma fragmentada en dos bases de datos, se realizan las siguientes operaciones:

```

SQL> select *
  2  from pacientes;

      ID NOMBRE          HABITACION
----- 1 Paciente1           10

SQL> select *
  2*   from pacientes@t11r2

      ID NOMBRE          FECHA_EN
----- 1 Paciente1       26/04/11

SQL> update pacientes
  2  set nombre='P1'
  3  where id=1;

1 fila actualizada.

SQL> update pacientes@t11r2
  2  set nombre='P1'
  3* where id=1

1 fila actualizada.

SQL> update pacientes
  2  set habitacion=10
  3  where id=2;

1 fila actualizada.

```

Antes de realizar commit, se puede observar qué está sucediendo, las transacciones creadas y los bloqueos generados en la BD servidor que en este caso es t11r2:

```
SQL> select saddr, sid, serial#, username, program, machine
  2  from v$session
  3  where username ='TJ';

SADDR      SID SERIAL# USERNAME      PROGRAM          MACHINE
-----  -----  -----  -----
37EFD738    28      46 TJ      sqlplus@caar (TNS V1-V3)  caar
37EFACCC    29      72 TJ      ORACLE.EXE        VILLEGUILLO

SQL> select SES_ADDR,START_TIME,STATUS,XID
  2  from v$transaction;

SES_ADDR START_TIME      STATUS      XID
-----  -----  -----
37EFACCC 05/06/11 16:03:43  ACTIVE      0600000062010000

SQL> select *
  2  from v$lock
  3  where sid=29;

ADDR      KADDR      SID TY      ID1      ID2      LMODE      REQUEST      CTIME      BLOCK
-----  -----  -----  -----
37ACA524 37ACA550    29 AE      100      0        4          0        731        0
00579434 00579464    29 TM      13704     0        3          0        593        0
369C8094 369C80D4    29 TX      393216    354      6          0        593        0
```

Se observa que la transacción existe, que la transacción es generada por la sesión de la conexión de la BD cliente y que tiene un bloqueo exclusivo sobre fila (LMODE=3 rowX) sobre la tabla PACIENTES:

```
SQL> select object_id
  2  from dba_objects
  3  where object_name='PACIENTES';

OBJECT_ID
-----
13704
```

A continuación, también se muestran sesiones, transacciones y bloqueos en la BD Cliente:

```
SQL> select saddr, sid, serial#, username, program
  2  from v$session
  3* where username='TJ'
```

```

SADDR          SID    SERIAL# USERNAME           PROGRAM
-----        -----
30F42BFC      146     1773 TJ                  sqlplus.exe

SQL> select SES_ADDR,START_TIME,STATUS,XID
  2*   from v$transaction

SES_ADDR START_TIME      STATUS      XID
-----  -----  -----
30F42BFC 05/06/11 16:02:46 ACTIVE      08002E00F8180000

SQL> select *
  2*  from v$lock
  3* where sid=146;

ADDR      KADDR      SID TY     ID1      ID2      LMODE REQUEST      CTIME      BLOCK
-----  -----  -----
2F5C67E4 2F5C67FC  146 TM     45926      0       3       0      1629      0
2F616204 2F616228  146 TX     524334    6392      6       0      1749      0

```

También se puede observar la información de los DB links abiertos en la sesión con la vista V\$DBLINK. Esta consulta se ejecuta en la propia sesión que tiene la transacción:

```

SQL> desc v$dblink
Nombre          ¿Nulo?  Tipo
-----  -----
DB_LINK          VARCHAR2(128)
OWNER_ID         NUMBER
LOGGED_ON        VARCHAR2(3)
HETEROGENEOUS   VARCHAR2(3)
PROTOCOL         VARCHAR2(6)
OPEN_CURSORS    NUMBER
IN_TRANSACTION   VARCHAR2(3)
UPDATE_SENT      VARCHAR2(3)
COMMIT_POINT_STRENGTH NUMBER

SQL> select *
  2*  from v$dblink;

DB_LINK      OWNER_ID LOG HET PROTOC OPEN_CURSORS IN_ UPD COMMIT_POINT_STRENGTH
-----  -----
T11R2        38 YES NO UNKN          0 YES YES          1

```

Una vez realizado el COMMIT, desaparecen las transacciones y los bloqueos. Y la vista de estado de DB links abiertos en la sesión muestra:

```

SQL> select *
  2*  from v$dblink;

DB_LINK      OWNER_ID LOG HET PROTOC OPEN_CURSORS IN_ UPD COMMIT_POINT_STRENGTH
-----  -----
T11R2        38 YES NO UNKN          0 NO NO          1

```

Cuando una transacción falla en alguno de los puntos la transacción se convierte en *transacción dudosa*. Puede pasar cuando algún nodo se cae, la conexión de red en la comunicación de algún nodo falla o cualquier error de software no controlado. El proceso RECO es el encargado de resolver las transacciones dudosas automáticamente y hasta que RECO no pueda resolver la transacción dudosa, el dato queda bloqueado para lectura o escritura.

7.5. Optimización de consultas sobre bases de datos distribuidas

Cuando se realiza una consulta distribuida, hay que tener en cuenta los siguientes parámetros:

OPEN_LINKS Configura la cantidad de máxima de conexiones abiertas concurrentemente para bases de datos remotas por cada sesión.

COMMIT_POINT_STRENGTH Especifica el valor de commit point site en una transacción. El nodo con el valor más alto se transforma en el commit point site.

Lo primero a tener en cuenta de optimización de consultas distribuidas son las estadísticas, tanto de las estadísticas de sistemas, como las de todos los segmentos que puedan intervenir en la consulta. Se ha de recordar que es transmisión de información por red, por lo que es importante que sea la mínima cantidad.

Para optimizar consultas distribuidas en Oracle se suelen utilizar dos técnicas. Usar consultas derivadas para disminuir la cantidad de información que se quiere recuperar a través de la red y la utilización de los *hints NO_MERGE* y *DYNAMIC_SITE*.

7.5.1. Optimización mediante consultas derivadas

Las consultas derivadas o *IN LINE VIEWS* se convierten en una estructura en memoria. Utilizando estas construcciones, se puede disminuir la cantidad de información que se desea recuperar. Por ejemplo, la consulta:

```
select l.id, l.habitacion, r.fecha_entrada
  from pacientes l,
       pacientes@t11r2 r
 where l.id=r.id
   and l.id=1
```

se puede optimizar utilizando una tabla derivada de la siguiente manera:

```
select l.id, l.habitacion, r.fecha_entrada
  from pacientes l,
       (select id, fecha_entrada
        from pacientes@t11r2
       where id=1) r
 where l.id=r.id
   and l.id=1
```

De esta manera, se recupera tan solo la información concerniente a un único paciente (el paciente con id=1), en lugar de todos los pacientes de la base de datos remota.

7.5.2. Optimización mediante hints

En entornos distribuidos es más común que el optimizador pueda no ser del todo bueno con los planes de ejecución, así que el uso de Hints (y su constante control) puede ser útil en mucho casos.

Con los Hints se pueden controlar cómo realizar una consulta. Por ejemplo, NO_MERGE evita los posibles no usos de consultas derivadas porque el optimizador puede determinar cambiar una tabla derivada por una join.

Con DRIVING_SITE se define en qué nodo se va a desarrollar la consulta, de tal manera que la BD donde se mezcle la información sea la que va a recibir menor cantidad de información a través de la red. Ejemplos de los cambios que producen estos Hints:

Ejecución sin hint:

```
SQL> select l.id, l.habitacion, r.fecha_entrada
  2   from pacientes l,
  3        (select id, fecha_entrada from pacientes@t11r2 where id=1) r
  4  where l.id=r.id and l.id=1
      ID HABITACION FECHA_EN
-----
 1          10 26/04/11
```

Ejecución con el hint NO_MERGE:

```
SQL> select /*+ NO_MERGE(r) */ l.id, l.habitacion, r.fecha_entrada
  2      from pacientes l, pacientes@t11r2 r
  3     where l.id=r.id and l.id=1;
    ID HABITACION FECHA_EN
  -----
    1          10 26/04/11
```

Ejecución con el hint DRIVING_SITE:

```
SQL> select /*+ DRIVING_SITE(r) */
  2      l.id, l.habitacion, r.fecha_entrada
  3      from pacientes l, pacientes@t11r2 r
  4     where l.id=r.id and l.id=1;
    ID HABITACION FECHA_EN
  -----
    1          10 26/04/11
```

7.6. Prácticas Resueltas

Práctica 7.1: Configuración de BBDD distribuidas

Prepara dos máquinas virtuales con Oracle y configúralas dando a una máquina la dirección 192.168.1.101 y a la otra la dirección 192.168.1.102. A continuación:

1. Edita el fichero *tnsnames.ora* que está ubicado en el directorio `\$ORACLE_HOME/network/admin` de ambas máquinas para agregar dos entradas que permitan enlazar las dos instancias.

Editar el fichero tnsnames en cada una de las máquinas introduciendo el siguiente código:

```
JARDIN=
(DESCRIPTION =
  (ADDRESS = (PROTOCOL = TCP)(HOST = 192.168.1.101)(PORT = 1521))
  (CONNECT_DATA = (SERVER = DEDICATED)
    (SERVICE_NAME = jardin)
  )
)

JARDIN2 =
(DESCRIPTION =
  (ADDRESS = (PROTOCOL = TCP)(HOST = 192.168.1.102)(PORT = 1521))
  (CONNECT_DATA = (SERVER = DEDICATED)
    (SERVICE_NAME = jardin)
  )
)
```

2. Comprueba que haya conectividad tns entre las dos instancias.

*En la máquina 192.168.1.101 y con el usuario oracle, introducir el comando:
tnsping jardin2
y en la máquina 192.168.1.102 introducir el comando:
*tnsping jardin**

3. Crea enlaces entre cada máquina necesarios para poder hacer consultas desde una máquina a la otra con el esquema *jardineria*. Desde la máquina 192.168.1.101 a la 102 el enlace se llamará *contabilidad*, mientras que de la 102 a la 101 se llamará *facturacion*.

*Primero hay que dar al usuario jardineria permisos de CREATE LINK en cada una de las máquinas:
sqlplus / as sysdba
Después, ejecutamos el comando SQL:
grant create database link to jardineria;
Finalmente, podemos ejecutar los comandos de creación de los enlaces. Desde la máquina 192.168.1.101:*

```
CREATE DATABASE LINK contabilidad
CONNECT TO jardineria IDENTIFIED BY jardineria
USING 'jardin2';
```

Y en la máquina 192.168.1.102:

```
CREATE DATABASE LINK facturacion\\
CONNECT TO jardineria IDENTIFIED BY jardineria\\
USING 'jardin';\\
}
```



Práctica 7.2: Trabajando con BBDD distribuidas

Realiza los siguientes ejercicios para aprender cómo trabajar con DB Links:

1. Edita la tabla empleados en la máquina 192.168.1.101 para añadir un campo que indique qué porcentaje de comisión se lleva cada empleado de los pedidos que gestiona y pon a todos los empleados un 5 % de comisión.

Desde la máquina 192.168.1.101, ejecutar:
alter table empleados add PorcentajeComision number(5,2);
update empleados set PorcentajeComision = 5;

2. Crea una tabla llamada Comisiones en la máquina 192.168.1.102 con los campos CódigoEmpleado, CódigoPedido, TotalPedido y Comision.

No es posible ejecutar comandos DDL a través de DB Links, por tanto, desde la máquina 192.168.1.102, ejecutamos el siguiente comando DDL:

```
create table Comisiones(
    CódigoEmpleado integer,
    CódigoPedido integer,
    TotalPedido number(15,2),
    Comision number(7,2)
);
```

3. Rellena la tabla de comisiones con los datos de los pedidos mediante una consulta distribuida a la tabla de pedidos en la máquina 192.168.1.101.

Hay que ejecutar la siguiente sentencia de tipo insert-select:

```
INSERT into comisiones@contabilidad
SELECT CódigoEmpleado, CódigoPedido, TotalPedido,
       TotalPedido*PorcentajeComisión/100
  FROM Empleados inner join Clientes on
        Empleados.CódigoEmpleado=Clientes.CódigoEmpleadoRepVentas natural join
        (select CódigoCliente,CódigoPedido,sum(Cantidad*PrecioUnidad) as
         TotalPedido from Pedidos natural join DetallePedidos
          group by CódigoCliente,CódigoPedido) Totales
}
```

4. Crea un trigger en la tabla Empleados que mantenga actualizada la tabla de comisiones cuando se modifica el porcentaje de comisión de la misma. Modifica la comisión de un empleado y comprueba que, efectivamente se ha modificado.

En la máquina 192.168.1.101 se ejecuta el siguiente código:

```
create or replace trigger TEmpleadoComisiones
after update of PorcentajeComisión on empleados for each row
begin
    update comisiones@contabilidad
    set comisión = totalpedido*:new.PorcentajeComisión/100
    where CódigoEmpleado=:old.CódigoEmpleado;
end;
/
```

5. Modifica el Empleado con código 31 estableciendo un porcentaje de comisión del 10% y verifica que efectivamente, se han modificado las comisiones:

```
update empleados set porcentajecomision=10 where codigoempleado=31;
select * from comisiones@contabilidad where codigoempleado=31;
```



7.7. Prácticas propuestas

Práctica 7.3: Fragmentación horizontal de tablas

Una empresa de telefonía está implementando un sistema de recepción de incidencias. Estas incidencias se recogerán desde un único número de asistencia telefónica y se almacenarán en una base de datos distribuida. Esta tendrá 3 nodos Oracle. Se va a fragmentar una tabla llamada incidencias de forma horizontal, de tal manera que las provincias cuyo nombre comience por las letras de la A a la C estén en el nodo 0, de la D a la M en el nodo 1 y de la O a la Z en el nodo 2. De esta manera estarán equilibradas en número de registros por el número de población. Se pide:

1. Crear la estructura de las tablas en los tres nodos. La tabla incidencia tendrá los campos IdIncidencia, Fecha, Código de Cliente, Texto de la incidencia, Provincia y Estado (abierta / cerrada).
2. Crear un enlace de base de datos entre cada uno de los servidores.
3. Ejecutar una consulta distribuida que devuelva todos los registros de la tabla incidencia.
4. Crear un procedimiento en el nodo 1 que inserte una incidencia en el nodo que le corresponda según la letra de la provincia.
5. Crear un procedimiento que borre una incidencia del nodo adecuado basándose en el campo de provincia.

A continuación se desea fragmentar la tabla a través del campo IdIncidencia y el operador Mod (Resto de la división entera). La operación IdIncidencia Mod 3 indicará en qué nodo debe estar almacenado el registro (0,1,2). Para asignar el IdIncidencia sin duplicados se va a crear una secuencia (objeto sequence) para que antes de insertar un registro se consulte y se inserte en el nodo adecuado. Se pide:

1. Crea la secuencia con el comando create sequence en el nodo 0.
2. Crea una función llamada SiguienteIncidencia que devuelva el siguiente número de secuencia.
3. Ejecuta la consulta select siguienteIncidencia()@nodo0 desde los otros dos nodos.



Práctica 7.4: Fragmentación vertical

Una aplicación comercial de marketing tiene una tabla de clientes muy grande con los siguientes campos:

- Campos principales: Código de Cliente, Nombre, Apellido1, Apellido2, email, identificador de facebook, dirección, código postal y país.
- Campos secundarios: película favorita 1, pelicula favorita 2, hobbie 1, hobbie 2, otros intereses y comentarios.

Se pretende fragmentar la tabla utilizando una base de datos distribuida de dos nodos, para lo cual se pide:

1. Realizar la fragmentación de la tabla separando los campos principales y creándolos en un nodo y los campos secundarios (gustos y aficiones) creándolos en otro nodo.
2. Crear un procedimiento que muestre por pantalla la información de un cliente determinado. El procedimiento recibirá como parámetro el código de cliente y consultará los dos nodos para mostrar toda la información.
3. Crear un procedimiento que elimine un cliente de la base de datos.



Práctica 7.5: Consultas distribuidas

Sigue los siguientes pasos para ejecutar consultas distribuidas en Oracle:

1. Genera una copia de la máquina virtual con tu servidor Oracle, cámbiale la dirección IP y asegúrate de que tienen conectividad entre ambas.
2. Localiza el fichero tnsnames de tu servidor oracle original y agrega una entrada para que se pueda crear un enlace a la máquina virtual copiada.
3. Prueba con la utilidad tnsping que es posible conectar de la máquina virtual original a la copiada.
4. Crea un usuario llamado User1 en la instancia oracle de la máquina virtual original con los permisos CREATE DATABASE LINK y CREATE PUBLIC DATABASE LINK y el rol RESOURCE.
5. Crea un usuario llamado User2 en la instancia oracle de la máquina virtual copiada con permiso CREATE SESSION y el rol RESOURCE.
6. Crea una tabla llamada Cliente(Código, Nombre, Dirección) en la máquina virtual original en el esquema User1 e inserta dos registros.
7. Crea una tabla llamada Mascotas(Código, Tipo, Raza, Nombre) en la máquina virtual copiada en el esquema User2 e inserta dos mascotas para cada Cliente.
8. Ejecuta una consulta que devuelva un listado con las mascotas de cada cliente incluyendo la dirección y el nombre del cliente.
9. Crea ahora la misma consulta, pero utilizando una tabla derivada para traer solamente las mascotas del primer cliente.
10. Ejecuta la consulta anterior utilizando los hints NO_MERGE y DRIVING_SITE comparando su plan de ejecución con el comando explain plan.



7.8. Resumen

Los conceptos clave de este capítulo son los siguientes:

- Una base de datos está controlada por más de un servidor se dice que está distribuida.
- Este tipo de organización proporciona más rapidez y flexibilidad en el acceso y más capacidad de almacenamiento. Proporciona también escalabilidad y fiabilidad.
- Un sistema de BBDD distribuidas requiere una red de comunicaciones y enlaces entre bases de datos locales a cada servidor (DBLINKS).
- Se clasifican en:
 - Homogéneas: todas las BBDD locales usan el mismo SGBD.
 - Heterogéneas: las BBDD locales pueden tener distinto SGBD.
- Se puede fragmentar una BBDD distribuida verticalmente, donde los fragmentos son conjuntos de campos de una tabla u horizontalmente donde los fragmentos son conjuntos de registros.
- Una consulta distribuida es aquella que consulta múltiples tablas ubicadas en distintas ubicaciones. Se puede hacer uso de *nombreTabla@servidor* para hacer referencia a una tabla en un servidor enlazado con un DBLINK.
- El comando DML *create database link* se utiliza para crear enlaces a través de un enlace declarado en el fichero tnsnames.ora.
- Con la utilidad tnsping se puede comprobar si hay conectividad con una BBDD remota.
- Una transacción distribuida utiliza el método TWO PHASE COMMIT a través de un coordinador local, un coordinador global y un Commit Point Site.
- En Oracle se puede consultar la vista v\$session para ver las sesiones abiertas con servidores remotos.
- Se puede eliminar una conexión abierta con el comando *alter session close database link*.
- Es posible utilizar *hints* en Oracle para parametrizar cómo se va a guiar la ejecución de la consulta.

7.9. Test de repaso

1. Un DBLink es

- a) Una conexión unidireccional entre una BBDD cliente y una servidora
- b) Una conexión bidireccional entre dos BBDD locales
- c) Una conexión bidireccional entre dos BBDD distribuidas
- d) Ninguna de las anteriores

2. Es posible tener BBDD Distribuidas heterogéneas

- a) Si, a través de DB Links comunes
- b) Sí, a través de software que hace las BBDD compatibles
- c) No, es imposible comunicar dos BBDD con distintos SGBD
- d) Sí, a través de agentes llamados *transparent gateways*

3. Con muchos DBLinks puede ocurrir que:

- a) Mejora el rendimiento del servidor
- b) No hay problema con las conexiones, puesto que se cierran solas
- c) Es posible que se queden demasiadas conexiones abiertas
- d) Hay problemas de seguridad

4. TWO PHASE COMMIT es

- a) Un enlace doble a BBDD
- b) Un mecanismo para realizar transacciones locales
- c) Un mecanismo para realizar transacciones distribuidas
- d) Un mecanismo para realizar transacciones distribuidas y locales

5. Las fases de TWO PHASE COMMIT

- a) Son dos fases
- b) Son la fase de preparación, commit y olvido
- c) Son 4 fases
- d) Son las fases de commit y rollback

6. Un hint es un mecanismo para

- a) Cambiar la forma en la que una consulta funciona
- b) Ejecutar una consulta distribuida
- c) Ejecutar una consulta local
- d) Ninguna de las anteriores

7. ¿Cuál de estas sentencias es la correcta?

- a) CREATE DATA BASE LINK;
- b) CREATE DB LINK;
- c) CREATE DATABASE LINK;
- d) Ninguna de las anteriores

8. Para comprobar si hay conectividad entre BBDD remotas

- a) Se usa el comando tnsnames
- b) Se usa el comando using
- c) Se usa el comando dblink
- d) Ninguna de las anteriores

9. Open links configura:

- a) El valor de commit en una transacción
- b) Máximo número de conexiones abiertas
- c) Tiempo máximo de ejecución
- d) Ninguna de las anteriores

Soluciones: 1.a, 2.d, 3.c, 4.c, 5.b, 6.a, 7.c, 8.d, 9.b

7.10. Comprueba tu aprendizaje

1. ¿Qué es una base de datos distribuida?
2. Enumera los tipos de BBDD distribuidas según el SGBD que se utiliza.
3. Enumera las características de una BBDD distribuida.
4. ¿Qué es la escalabilidad?
5. ¿Qué significa que una BBDD es fiable?
6. Enumera los 5 elementos que tiene una BBDD distribuida.
7. ¿Por qué se fragmenta una BBDD distribuida?
8. Define los siguientes conceptos:
 - Fragmentación vertical
 - Fragmentación horizontal
 - Fragmentación mixta
9. ¿Qué reglas hay para que una fragmentación sea correcta?
10. ¿Qué diferencia hay entre una consulta distribuida y una remota?
11. ¿Cómo se ejecuta una consulta distribuida?
12. ¿En qué consiste un *transparent gateway* o pasarela transparente?
13. ¿Qué es un DBLINK?
14. ¿Qué requisitos debe tener un usuario para poder crear un DB Link?
15. ¿Qué almacena el fichero tnsnames.ora?
16. ¿Qué diferencia hay entre un dblink público y uno privado?
17. Escribe el comando DDL para crear DBLINKS.
18. ¿Para qué sirve la utilidad tnsping?
19. ¿Cómo se cierra una conexión a una base de datos remota?
20. Describe el funcionamiento de una transacción distribuida a través del mecanismo TWO PHASE COMMIT.
21. Enumera los hints que tiene Oracle para optimizar las consultas distribuidas.