Cómo usar un DBMS centralizado para implementar una base de datos distribuida

Conexión entre localidades

Para implementar una base de datos distribuida usando un sistema de gestión de bases de datos centralizadas, como es ORACLE, lo primero es disponer de una serie de cuentas Oracle que son las que actuarán de localidades del sistema distribuido. En nuestro caso, para realizar la práctica, se necesitarán cuatro cuentas que se corresponderán con las cuatro localidades que se indican en el enunciado.

Para conectar estas cuatro "localidades" y poder acceder desde cualquiera de ellas a los datos almacenados en las demás localidades, lo único que hay que hacer es, desde cada cuenta, autorizar a las demás cuentas a acceder a los datos allí almacenados. Para ello utilizaremos la instrucción GRANT de SQL. Esta instrucción tiene el siguiente formato:

```
GRANT {privilegios | ALL [PRIVILEGES]} [(columna [, columna]..)]
        [,{privilegios | ALL [PRIVILEGES]} [(columna [, columna]..)]] ...
        ON [esquema.]objeto
        TO usuario [, usuario]...
        [WITH GRANT OPTION]
```

privilegios:

Algunos de ellos son: DELETE, INSERT, REFERENCES, SELECT y UPDATE.

ALL PRIVILEGES:

Otorga todos los privilegios sobre el objeto indicado.

columna:

Indica una columna de una tabla o vista sobre la cual se están otorgando los privilegios.

ON:

Indica el objeto sobre el cual se están otorgando los privilegios, Puede ser, entre otros, una tabla o una vista.

TO:

Indica los usuarios a los que se les otorga los privilegios indicados.

WITH GRANT OPTION:

Le otorga al usuario al que se le ha dado un privilegio sobre un objeto el permiso de otorgar, a su vez, dicho privilegio a otros usuarios.

El uso de los privilegios puede hacerse desde el mismo momento en que son otorgados.

Al otorgar privilegios a un usuario, estos se añaden a los privilegios que ya poseía..

Ejemplo 1:

Se tienen definidas las siguientes cuatro cuentas (usuarios): diamante1, diamante2, diamante3 y diamante4.

Se tiene definida una relación (tabla) Empleados que está fragmenta horizontalmente en los siguiente cuatro fragmentos: Empleado1, Empleado2, Empleado3 y Empleado4. Cada uno de estos fragmentos está almacenado en una de las cuentas antes mencionadas: Empleado1 está almacenado en diamante1, Empleados2 en diamante2, y así sucesivamente.

Para poder acceder desde cualquier cuenta (localidad) a los datos almacenados en cualquiera de los fragmentos, hay que ejecutar las siguientes instrucciones:

Desde la cuenta diamante1 (en la que se almacena el fragmento Empleados1)

```
GRANT DELETE, INSERT, UPDATE, SELECT
ON Empleados1
TO diamente2, diamente3, diamante4;
commit;
```

Desde la cuenta diamante2 (en la que se almacena el fragmento Empleados2)

```
GRANT DELETE, INSERT, UPDATE, SELECT
ON Empleados2
TO diamente1, diamente3, diamante4;
commit;
```

Desde la cuenta diamante3 (en la que se almacena el fragmento Empleados3)

```
GRANT DELETE, INSERT, UPDATE, SELECT
ON Empleados3
TO diamente1, diamente2, diamante4;
commit;
```

Desde la cuenta diamante4 (en la que se almacena el fragmento Empleados4)

```
GRANT DELETE, INSERT, UPDATE, SELECT
ON Empleados4
TO diamente1, diamente2, diamante3;
commit;
```

Con la ejecución de estas instrucciones desde las correspondientes cuentas (usuarios), es posible borrar, insertar, actualizar y consultar datos de cualquier fragmento desde cualquier localidad (cuenta)

La instrucción commit, es necesaria para que los privilegios otorgados por la instrucción GRANT se hagan efectivos.

Como se puede observar en el formato de la instrucción GRANT, se pueden otorgar, en una misma instrucción, varios privilegios a varios usuarios, pero solamente sobre un único objeto (tabla o vista)

Esto quiere decir, que si desde una cuenta (usuario o localidad), en la que se tiene almacenadas varias tablas, se desea otorgar privilegios a varios otros usuarios sobre todas ellas, habrá que definir y ejecutar una instrucción GRANT para cada tabla.

Transparencia

Ya tenemos resuelto el problema de cómo acceder a los datos almacenados en las distintas localidades (cuentas) desde cualquier localidad (cuenta). Ahora nos queda resolver el problema de la transparencia, es decir, el problema de que cuando se acceda a los datos no haya que saber ni dónde están ubicados ni cómo están fragmentados (en caso de que lo estén).

Para hacer las consultas transparentes, hay que definir, en cada cuenta (localidad) y para cada tabla fragmentada, una vista que reconstruya la relación (tabla) global a partir de sus fragmentos.

Ejemplo 2:

Supongamos la relación (tabla) Empleados fragmentada horizontalmente en cuatro fragmentos: Empleados1, Empleados2, Empleados3 y Empleados4. Estos cuatro fragmentos están asignados a las localidades (cuenta) tal y como se especifica en el Ejemplo1.

En cada localidad (cuenta) hay que tener definida la siguiente vista:

```
CREATE VIEW Empleados AS
SELECT * FROM diamente1.Empleados1
UNION
SELECT * FROM diamente2.Empleados2
UNION
SELECT * FROM diamente3.Empleados3
UNION
SELECT * FROM diamente4.Empleados4;
```

Se observa que al nombre de la tabla que se corresponde con un fragmento, hay que anteponer el nombre de la cuenta (análogo al identificador de la localidad) en la que se almacena dicho fragmento. Evidentemente, para poder definir esta vista, es necesario que todos los usuarios (cuentas) tengan el privilegio SELECT para todas las tablas de las que no es propietario.

Teniendo definida esta vista, en cada lofaclidad, cuando se desee hacer una consulta que involucre datos de empleados, dicha consulta se podrá definir sobre la vista "Empleados". Por ejemplo:

```
SELECT nombre, salario
FROM Empleados
WHERE salario > 2000;
```

Para hacer transparentes las actualizaciones, éstas se definirán e implementarán mediante procedimientos (práctica 4)