Unidad III: Lenguaje de manipulación de datos (DML)

3.1 Inserción, eliminación y modificación de registros

La sentencia INSERT permite agregar nuevas filas de datos a las tablas existentes. Está sentencia tiene como sintaxis genérica.

INSERT INTO **tabla_o_vista [(lista_de_columnas)] VALUES (lista_de_valores_de_datos)**

Para insertar datos en una relación, se específica la tupla que se desea insertar o se formula una consulta cuyo resultado sea el conjunto de tuplas que se desea insertar. Obviamente, los valores de los atributos de las tuplas que se inserten deben pertenecer al dominio de los atributos. De igual modo, las tuplas insertadas deben ser de la aridad -**número de atributos- correcta.**

Borrará todas las filas que cumplan la condición especificada en la cláusula **WHERE**. Si esta cláusula se omite, se borrarán todas las filas de la tabla. **DELETE** borra todas las filas de una tabla, pero no la definición de la tabla del diccionario. Esta es una diferencia con la sentencia **DROPTABLE**, que elimina el contenido de la tabla y la definición de la misma.

La sintaxis es la que sigue:

DELETE FROM tabla [**WHERE** condición];

Obsérvese que cada comando **DELETE** sólo opera sobre una relación. Si se desea borrar tuplas de varias relaciones es necesario utilizar una orden **DELETE** por cada relación.

La consulta

```
DELETE FROM empleados;
```

Borra todas las tuplas de la tabla empleados

La consulta

```
DELETE FROM empleados WHERE cargo = 'Chofer';
```

Borra los empleados con cargo de chofer

Cuando trabajemos con la sentencia `DELETE` debemos tener en cuenta las siguientes consideraciones:

- Solo podemos borrar datos de una única tabla.
- Cuando borramos datos de una vista, los estamos borrando también de la tabla. Las vistas son solo una forma de ver los datos, no una copia.
- Si intentamos borrar un registro de una tabla referenciada por una FOREING KEY como tabla maestra, si la tabla dependiente tiene registros relacionados la sentencia DELETE fallará a no ser que la tabla hija contenga la sentencia ON DELETE CASCADE..

Considere el siguiente Diagrama Entidad Relación y su correspondiente script Oracle

En determinadas situaciones puede ser deseable cambiar un valor dentro de una tupla, sin cambiar todos los valores de la misma. Para ello se utiliza el comando **UPDATE** cuya sintaxis se muestra a continuación.

UPDATE tabla **SET {columna = expresión,} + [WHERE condición**;**]**

Se especificará en la cláusula **SET** las columnas que se actualizarán y con qué valores. La cláusula **WHERE** indica las filas con las que se va a trabajar, sin la cláusula **WHERE** la actualización afectará a todas las filas de la tabla.

Si se desea actualizar a nulos, se asignará el valor **NULL**.

3.2 Consultas de registros

La recuperación de los datos en el lenguaje SQL se realiza mediante la sentencia SELECT, seleccionar. Esta sentencia permite indicar al SGBD la información que se quiere recuperar. Esta es la sentencia SQL, con diferencia, más habitual. La sentencia SELECT consta de cuatro partes básicas:

La cláusula SELECT seguida de la descripción de lo que se desea ver, los nombres de las columnas a seleccionar. Esta parte es obligatoria.

La cláusula FROM seguida de la especificación de las tablas de las que se han de obtener los datos. Esta parte es obligatoria.

La cláusula WHERE seguida por un criterio de selección, una condición. Esta parte es opcional.

La cláusula ORDER BY seguida por el criterio de ordenación. Esta parte es opcional.

Una primera aproximación a la sintaxis de la sentencia SELECT puede mostrarnos la siguiente expresión:

SELECT {* | {columna,}+} FROM {tabla,}+ [WHERE condición] [ORDER BY {expresiónColumna [ASC | DESC],}+];

Como un primer uso de la sentencia SELECT podemos usarla para visualizar todas las tablas contenidas en la base de datos.

3.2.1 Recuperación de datos

La recuperación de los datos en el lenguaje SQL se realiza mediante la sentencia **SELECT**, seleccionar. Esta sentencia permite indicar al SGBD la información que se quiere recuperar. Esta es la sentencia SQL, más habitual. La sentencia SELECT consta de cuatro partes básicas:

- La cláusula **SELECT** seguida de la descripción de lo que se desea ver, los nombres de las columnas a seleccionar. Esta parte es obligatoria.
- La cláusula **FROM** seguida de la especificación de las tablas de las que se han de obtener los datos. Esta parte es obligatoria.
- La cláusula **WHERE** seguida por un criterio de selección, una condición. Esta parte es opcional.
- La cláusula **ORDER BY** seguida por el criterio de ordenación. Esta parte es opcional.

Una primera aproximación a la sintaxis de la sentencia **SELECT** puede mostrarnos la siguiente expresión:

SELECT {* | {columna,}+} FROM {tabla,}+ [WHERE condición] [ORDER BY {expresiónColumna [ASC | DESC],}+];

Selección de columnas

Las columnas a seleccionar se enumeran sin más en la cláusula **SELECT**. Para seleccionar todas las columnas use el carácter asterisco ' *****'.

Cuando se consulta una base de datos, los nombres de las columnas se usan como cabeceras de presentación. Si éste resulta demasiado largo, corto o críptico, puede cambiarse con la misma sentencia SQL de consulta, creando un alias de columna.

3.2.2 Restricción y ordenación de datos

La sentencia **SELECT** recupera todas las columnas o un subconjunto de ellas de una tabla. Esto afecta a todas las filas de la tabla, a menos que especifiquemos una condición en la cláusula **WHERE**.

Esta condición regresa todas las filas que cumplen dicha condicional. La complejidad del criterio de búsqueda es prácticamente ilimitada, y en él se pueden combinar operadores de diversos tipos con funciones de columnas, componiendo expresiones más o menos complejas.

Condición de búsqueda basada en una comparación compuesta

En este ejemplo, se utiliza el operador lógico **OR** en la cláusula **WHERE** para localizar los empleados que son chofer o secretaria. La consulta y la tabla resultadas se muestra a continuación.

La expresión **CASE** permite utilizar la lógica **IF-THEN-ELSE** en sentencias SQL sin tener que invocar procedimientos. Esta expresión se incluye a partir de la versión Oracle9i Server y MySQL 5.

3.2.3 Informes de datos agregados mediante funciones de grupo

La cláusula **GROUP BY** combina los registros con valores idénticos en la lista de campos especificada en un único registro. Para cada registro se puede crear un valor agregado si se incluye una función SQL agregada como por ejemplo **SUM** o **COUNT**, en la instrucción **SELECT**. Su sintaxis es:

```
SELECT [ALL | DISTINCT]
<nombre_campo> [{,<nombre_campo>}] [{,<funcion_agregación>}]
FROM <nombre_tabla>|<nombre_vista>
[{,<nombre_tabla>|<nombre_vista>}]
[WHERE <condición> [{ AND|OR <condición>}]]
[GROUP BY <nombre_campo> [{,<nombre_campo >}]]
[HAVING <condición>[{ AND|OR <condición>}]]
[ORDER BY <nombre_campo>|<campo_índice> [ASC | DESC]
[{,<nombre_campo>|<campo_índice> [ASC | DESC ]}]]
```

En la cláusula **GROUP BY** se colocan las columnas por las que vamos a agrupar. Y en la cláusula **HAVING** filtra los registros una vez agrupados.

3.2.4 Visualización de datos de varias tablas

Todos los ejemplos ejecutados hasta ahora tienen una limitación significativa, es decir todas las columnas incluidas en la tabla resultados provienen de una misma tabla.

La verdadera potencia del SQL se alcanza cuando combinamos el contenido de más de una tabla. Suponga que desea conseguir una lista con los empleados y los departamentos para los que trabajan. Esta información está repartida en las dos tablas **empleado** y **departamento**.

Si necesitamos obtener información de más de una tabla, tendremos la opción de utilizar **subconsultas** o **combinaciones**. Si la tabla de resultados final debe contener columnas de tablas diferentes, entonces deberemos utilizar obligatoriamente una combinación. Para realizar una combinación, basta incluir más de un nombre en la cláusula **FROM**, utilizando una coma como separador y, normalmente, incluyendo una clâusula**WHERE** para especificar la columna o columnas con las que hay que realizar la combinación.

Así, podríamos intentar una consulta que seleccionara el campo nombre de la tabla empleado y el nombre del departamento. Y aquí surge el primer problema,

¿cómo distinguimos entre dos columnas que llamándose igual, pertenecen a tablas distintas?
El uso de alias para las tablas incluidas en el **FROM** o en las columnas del **SELECT** permite evitar la ambigüedad, el alias se separa por un espacio.

3.2.5 Subconsultas

Una subconsulta es una instrucción **SELECT** anidada dentro de una sentencia **SELECT**, **SELECT...INTO**, **INSERT...INTO**, **DELETE**, o **UPDATE**o dentro de otra subconsulta.

Una subconsulta, a su vez, puede contener otra subconsulta y así hasta un máximo de 16 niveles. Las particularidades de las subconsultas son:

1. **Su resultado no se visualiza, sino que se pasa a la consulta principal para su comprobación.**
2. **Puede devolver un valor único o una lista de valores y en dependencia de esto se debe usar el operador del tipo correspondiente.**
3. **No puede usar el operador BETWEEN, ni contener la setencia ORDER BY.**
4. **Puede contener una sola columna, que es lo más común, o varias columnas.Este último caso se llama subconsulta con columnas múltiples. Cuando dos o más columnas serán comprobadas al mismo tiempo, deben encerrarse entre paréntesis.**

Ejemplo: Nombres de los jugadores que han participado más que el promedio, equipo y posición durante un torneo

```
SELECT equipo, nombreJugador, posicion, minutos
FROM femexfut
WHERE minutos > ( SELECT AVG(minutos) FROM femexfut
       WHERE torneo = 'Bicentenario 2010' AND jj > 0)
       AND torneo = 'Bicentenario 2010'
ORDER BY equipo, posicion, nombreJugador
```

3.2.6 Operadores set

Las consultas multitabla o JOINS. también denominadas combinaciones o composiciones, permiten recuperar datos de dos tablas o más según las relaciones lógicas entre ellas. Las combinaciones indican cómo debería utilizar el SGBD los datos de una tabla para seleccionar los datos de otra tabla.

Una condición de combinación (o composición) define la forma en la que dos tablas se relacionan en una consulta al:

- Especificar la columna de cada tabla que debe usarse para la combinación. Una condición de combinación específica una clave externa de una tabla y su clave asociada en otra tabla. Relación padre - hija.
- Especificar un operador lógico (=, <>, etc.) para usarlo en los valores de comparación de las columnas

Es una operación que combina registros de dos tablas en una base de datos relacional que resulta en una nueva tabla (temporal) llamada tabla de JOIN. En el lenguaje de consulta SQL hay dos tipos de JOIN: **INNER** y **OUTER**.

Como caso especial, una tabla (tabla base, vista o una tabla JOIN) puede realizar la operación JOIN sobre ella misma. Esto se conoce como self-JOIN.

Ejemplo: Jugadores que participaron con el Cruz Azul durante el torneo Apertura 2011, minutos jugados y partidos en los que participo (JJ)

```
SELECT J.nombre, J.jj, J.minutos
FROM jugadores_Clausura2011 J, equipos E
WHERE (J.idEquipo = E.idEquipo) AND (E.idEquipo = 5)
ORDER BY J.jj DESC, J.minutos DESC, J.nombre
```