1º CFGS Desarrollo De Aplicaciones Multiplataforma

UD 4. El lenguaje SQL - DML.

Módulo: Bases de Datos

Centro de Enseñanza
Gregorio Fernández

1. Introducción

- **SQL** es el lenguaje fundamental de los SGBD relacionales.
- Es sin duda el lenguaje fundamental para manejar una base de datos relacional.
- Entre las principales características de SQL podemos destacar las siguientes:
  - Es un **lenguaje declarativo**, es decir, lo importante es definir qué se desea hacer, no cómo hacerlo.
  - Las instrucciones de SQL utilizan un **lenguaje natural**. De ahí que se le considere un lenguaje de cuarta generación.
  - Es un lenguaje para **todo tipo de usuarios**: administradores, desarrolladores y usuarios normales. Permite realizar consultas, actualizaciones, definición de datos y control.

1. Introducción

- El nacimiento del lenguaje SQL data de 1970 cuando Codd publica su libro: *"Un modelo de datos relacional para grandes bancos de datos compartidos"*.
- Apenas dos años después IBM (para quien trabajaba Codd) utiliza las directrices de Codd para crear el *Standard English Query Language,* al que se le llamó SEQUEL. Más adelante se le asignaron las siglas **SQL** (*Structured Query Language*, lenguaje estructurado de consulta).
- En 1979 **Oracle** presenta la primera implementación comercial del lenguaje. Poco después se convertía en un estándar en el mundo de las bases de datos avalado por los organismos ISO y ANSI.
- En el año 1986 se toma como lenguaje estándar por ANSI de los SGBD relacionales. Un año después lo adopta ISO, lo que convierte a SQL en estándar mundial como lenguaje de bases de datos relacionales.
- En 1989 aparece el estándar ISO (y ANSI) llamado **SQL89** o SQL1. En 1992 aparece la nueva versión estándar de SQL (a día de hoy sigue siendo la más conocida) llamada **SQL92**. En 1999 se aprueba un nuevo SQL estándar que incorpora mejoras que incluyen triggers, procedimientos, funciones,… y otras características de las bases de datos objeto-relacionales, dicho estándar se conoce como **SQL99**.
- A lo largo de los años se han ido añadiendo revisiones, que incorporan nuevas utilidades y mejoras.

2. Modos de ejecución de SQL

Ejecución directa. SQL interactivo

- Las instrucciones SQL se introducen a través de un cliente directamente conectado al servidor SQL. Las instrucciones se traducen sin intermediarios y los resultados se muestran en el cliente.

Ejecución incrustada o embebida

- Las instrucciones SQL se colocan como parte del código de otro lenguaje que se considera anfitrión (C, Java, Pascal, Visual Basic,...). Al compilar el código se utiliza un precompilador del propio SGBD para traducir el SQL y conectar la aplicación resultado con la base de datos a través de un software adaptador (driver) como JDBC u ODBC por ejemplo.

Ejecución a través de clientes gráficos

- Se trata de un software que permite conectar a la base de datos a través de un cliente. El software permite manejar de forma gráfica la base de datos y las acciones realizadas son traducidas a SQL y enviadas al servidor. Los resultados recibidos vuelven a ser traducidos de forma gráfica para un manejo más cómodo.

2. Modos de ejecución de SQL

Procesamiento de una instrucción SQL

1. Se analiza la instrucción, para comprobar su sintaxis.
2. Si es correcta se valora si los metadatos son correctos. Se comprueba esto con la información del diccionario de datos.
3. Si es correcta, se optimiza, a fin de consumir los mínimos recursos posibles.
4. Se ejecuta la sentencia y se muestra el resultado.

3. Elementos del lenguaje SQL: Sentencias SQL

- Son las distintas instrucciones que se pueden ejecutar utilizando SQL. El lenguaje SQL proporciona un gran repertorio de sentencias que permiten consultar datos, crear, actualizar y eliminar objetos de la base de datos, crear, actualizar y eliminar datos, controlar el acceso a la base de datos y a sus objetos.
- Dependiendo de las tareas, las sentencias SQL se pueden clasificar en los siguientes tipos:
  - **DML** (*Data Manipulation Language* - Lenguaje de Manipulación de Datos): INSERT, UPDATE, DELETE y SELECT.
  - **DDL** (*Data Definition Language* - Lenguaje de Definición de Datos). Sentencias que permiten crear y modificar la estructura de la base de datos. Instrucciones: CREATE, ALTER, DROP, RENAME y TRUNCATE.
  - **DCL** (*Data Control Language* - Lenguaje de Control de Datos). Sentencias que permiten administran los derechos y restricciones de los usuarios de la base de datos. Instrucciones GRANT y REVOKE.

3. Elementos del lenguaje SQL:
Normas de escritura

- En SQL no se distingue entre mayúsculas y minúsculas.
- Las instrucciones finalizan con el signo de punto y coma.
- Cualquier comando SQL (SELECT, INSERT,...) puede ser partido utilizando espacios o saltos de línea antes de finalizar la instrucción.
- Se pueden tabular líneas para facilitar la lectura si fuera necesario.
- Los comentarios en el código SQL comienzan por /* y terminan por */ (excepto en algunos SGBD).

3. Elementos del lenguaje SQL: Tipos de datos

Texto	
Tipo	Descripción
CHAR(n)	Texto de longitud fija
NCHAR(n)	Texto de longitud fija UNICODE
VARCHAR2(n)	Texto de longitud variable
NVARCHAR2(n)	Texto de longitud variable UNICODE
Números	
Tipo	Descripción
NUMBER	Enteros con precisión decimal
Fechas	
Tipo	Descripción
DATE	Fechas
TIMESTAMP	Fecha y hora
INTERVAL	Intervalos
Datos de gran tamaño	
Tipo	Descripción
CLOB	Texto de gran longitud
BLOB	Binario de gran longitud

4. Sentencia SELECT

- Para recuperar información o, lo que es lo mismo, realizar consultas a la base de datos, utilizaremos la sentencia **SELECT**.
- Sintaxis de la sentencia SELECT:

```
SELECT   [ALL | DISTINCT]
         [expr_columna1, expr_columna2, … , expr_columnaN | *]
FROM     tabla1 [, tabla2, … , tablaN]
[WHERE condición]
[ORDER BY expr_columna1 [DESC | ASC] [,expr_columna2 [DESC | ASC]…];
```

- Donde:
  - **expr_columna:** puede ser una columna de una tabla, una constante, una expresión aritmética, una función o varias funciones anidadas.
  - ***:** el asterisco significa que se seleccionan todas las columnas.

4.1. Sentencia SELECT: FROM

- Especifica la tabla o tablas de las que se recuperarán los datos.
- Ejemplo de consulta de los nombres de alumnos y su nota de la tabla ALUMNOS:

```
SELECT nom_alumno, nota FROM Alumnos;
```

- Si el usuario que hace la consulta no es el propietario de la tabla, es necesario especificar el nombre de usuario delante de ella: *nombre_usuario.nombre_tabla.* Por ejemplo, si el propietario de la tabla se llama PROFESOR pondremos:

```
SELECT nom_alumno, nota FROM Profesor.Alumnos;
```

- Es posible asociar un nuevo nombre a las tablas mediante *alias*. Por ejemplo podemos poner el alias A a la tabla de ALUMNOS de la siguiente forma:

```
SELECT A.nom_alumno, A.nota FROM Alumnos A;
```

4.2. Sentencia SELECT: WHERE

- Esta cláusula permite obtener las filas que cumplen la condición expresada.
- El formato de la condición es:

  expresión operador_comparación expresión

- Las expresiones pueden ser: una constante, una expresión aritmética, un valor nulo o un nombre de columna.
- Los operadores de comparación pueden ser los siguientes:

=, >, <, >=, <=, !=, <>
IN, NOT IN, BETWEEN, NOT BETWEEN
LIKE

- Se pueden construir condiciones múltiples utilizando los operadores lógicos AND, OR y NOT. Se pueden usar paréntesis para forzar el orden de evaluación.
- Veamos algunos ejemplos de condiciones en la cláusula WHERE:

```
WHERE nota=5;
WHERE (nota>=10) AND (curso=1);
WHERE (nota IS NULL) OR (UPPER(nom_alumno)='PEDRO')
```

Centro de Enseñanza Gregorio Fernández

4.3. Sentencia SELECT: ORDER BY

- El orden inicial de los registros obtenidos por una SELECT es el orden en el que fueron introducidos. Para ordenar en base a otros criterios, se utiliza esta claúsula.
- En esa cláusula se coloca una lista de campos que indica la forma de ordenar. Se ordena primero por el primer campo de la lista, si hay coincidencias por el segundo, si ahí también las hay por el tercero, y así sucesivamente.
- Se puede colocar las palabras ASC o DESC, que significan ascendente o descendente respectivamente(por defecto se toma ASC).
- Ejemplo:

```
SELECT * FROM Alumnos ORDER BY nom_alumno, curso DESC;
```

- Esta sentencia ordena por nombre de alumno ascendentemente y en caso de coincidencia por curso descendentemente.

4.4. Sentencia SELECT: ALL | DISTINCT

ALL

- Recupera todas las filas aunque algunas estén repetidas. Es la opción por defecto.

DISTINCT

- Sólo recupera las filas que son distintas. Por ejemplo, si consultamos los departamentos de la tabla EMPLEADO, vemos que aparecen números de departamento repetidos. Con DISTINCT se eliminan esas filas repetidas.

```
SELECT DISTINCT num_dpto FROM Empleados;
```

4.5. Sentencia SELECT: Alias de columnas

- Cuando se realizan consultas a la base de datos, los nombres de las columnas aparecen como cabeceras de presentación.
- Si el nombres es demasiado largo, corto ó críptico, existe la posibilidad de cambiarlo en la misma sentencia SELECT creando un *alias*.
- El *alias* se pone entre comillas dobles a continuación del nombre de la columna.
- Ejemplo:

```
SELECT nom_dpto “Departamento”, num_dpto “Nº departamento” FROM Departamentos;
```

5. Operadores aritméticos

- Los operadores aritméticos +, -, * y /, sirven para hacer cálculos en las consultas.
- Cuando se utilizan como expresión en una sentencia SELECT, no modifican los datos originales, sino que como resultado aparece una nueva columna.
- Ejemplo:

  SELECT nom_alumno, (nota1 + nota2 + nota3)/3 “Nota media” FROM Notas_Alumnos;

- La prioridad de los operadores aritméticos es la normal, tienen más prioridad la multiplicación y división, después la suma y la resta.
- En caso de igualdad de prioridad, se realiza primero la operación que esté más a la izquierda.
- Se puede modificar la prioridad usando paréntesis.

6. Concatenación de textos

- Todas las bases de datos incluyen algún operador para encadenar textos.
- En Oracle son los signos ||.
- Ejemplo:

```
SELECT tipo, modelo, tipo || '-' || modelo "Clave pieza" FROM Piezas;
```

7. Condiciones

- Se utiliza la cláusula **WHERE.**
- Permite colocar una condición que han de cumplir todas las filas.
- Las que no la cumplan no aparecen en el resultado.

7.1. Condiciones: Operadores de comparación

Operador	Significado
>	Mayor que
<	Menor que
>=	Mayor o igual que
<=	Menor o igual que
=	Igual
<>	Distinto
!=	Distinto

- Se pueden utilizar tanto para comparar números como para comparar textos y fechas. Ejemplo:

```
SELECT nom_alumno FROM Alumnos WHERE (nota1+nota2+nota3)/3 >6;
```

7.2. Condiciones: Operadores lógicos

Operador	Significado
AND	Devuelve TRUE si las dos condiciones son verdaderas.
OR	Devuelve TRUE cuando una de las dos condiciones es verdadera.
NOT	Invierte la lógica de la condición. Si era verdadera pasa a ser falsa, y viceversa.

- Ejemplo:

```
SELECT apellido, salario, num_dpto FROM Empleados
WHERE (salario>2000) AND (num_dpto=10 OR num_dpto=20);
```

7.3. Condiciones: BETWEEN

- Permite obtener datos que se encuentren dentro de un rango. Su formato es:

  <expresión> [**NOT**] **BETWEEN** valor_inicial **AND** valor_final

- Ejemplos:

```
/* Obtiene los empleados con salario entre 1500 y 2000 euros */
SELECT apellido, salario FROM Empleados WHERE salario BETWEEN 1500 AND 2000;

/* Obtiene los empleados cuyo salario no está entre 1500 y 2000 euros */
SELECT apellido, salario FROM Empleados WHERE salario NOT BETWEEN 1500 AND 2000;
```

7.4. Condiciones: IN

- Permite comprobar si una expresión pertenece o no a una lista de valores, haciendo posible la realización de comparaciones múltiples. Su formato es:

  <expresión> [**NOT**] **IN** (lista de valores separados por comas)

- Ejemplos:

```
/* Obtiene los empleados cuyo nº de departamento sea 10 ó 30 */
SELECT apellido FROM Empleados WHERE num_dpto IN (10,30);

/* Obtiene los empleados cuyo oficio no sea ni vendedor ni analista */
SELECT apellido FROM Empleados WHERE oficio NOT IN ('vendedor','analista');
```

7.5. Condiciones: LIKE

- Para comparar cadenas de caracteres que cumplan un patrón no nos sirve el operador =. Para estos casos podemos usar el operador LIKE que permite utilizar los siguientes comodines en las cadenas de comparación:

Operador	Significado
%	Representa cualquier cadena de 0 ó más caracteres
_	Representa un carácter cualquiera

- Su formato es: **WHERE** columna **LIKE** 'caracteres_especiales'
- Ejemplos:

```
/* Obtiene los empleados cuyo apellido empieza por J */
SELECT apellido FROM Empleados WHERE apellido LIKE 'J%';
/* Obtiene los empleados cuyo apellido tenga una R en la segunda posición */
SELECT apellido FROM Empleados WHERE apellido LIKE '_R%';
/* Obtiene los empleados cuyo apellido empieza por A y tengan una O en su interior */
SELECT apellido FROM Empleados WHERE apellido LIKE 'A%O%';
```

7.6. Condiciones: NULL

- Una columna de una fila es NULL si está completamente vacía.
- Para comprobar si el valor de una columna es nulo no podemos utilizar los operadores de igualdad mayor o menor. Para ello utilizamos la expresión:

columna **IS** [**NOT**] **NULL**

- Ejemplo:

```
/* Obtiene los empleados que no tienen teléfono */
SELECT apellido FROM Empleados WHERE telefono IS NULL;
```

7.7. Condiciones: Precedencia

Orden de precedencia	Operador
1	*, /
2	+, -
3	|| (concatenación)
4	Comparaciones (>, <, …)
5	IS [NOT] NULL, [NOT] LIKE, IN
6	NOT
7	AND
8	OR