Clase 05. Sublenguaje SQL DML

Sub Lenguaje DML

Concepto

El sublenguaje de manipulación de datos (DDL por sus siglas en inglés) es un lenguaje que permite a los usuarios tener acceso a la manipulación de los datos organizados mediante el modelo de datos correspondiente.
Los tipos de acceso son:

1. La recuperación de la información almacenada en la base de datos.
2. La inserción de información nueva en la base de datos.
3. El borrado de la información de la base de datos.
4. La modificación de la información almacenada en la base de datos.

Hay fundamentalmente dos tipos:

A. Los DDLs procedimentales necesitan que el usuario especifique qué datos se necesitan y cómo obtener esos datos.
B. Los DDLs declarativos (también conocidos como DDLs no procedimentales) necesitan que el usuario especifique qué datos se necesitan sin que haga falta que especifique cómo obtener esos datos.
   Los DMLs declarativos suelen resultar más fáciles de aprender y de usar que los procedimentales. Sin embargo, como el usuario no tiene que especificar cómo conseguir los datos, el sistema de bases de datos tiene que determinar un medio eficiente de acceso a los datos.

Sentencias y sintaxis

INSERT

La cláusula insert permite insertar datos en cada tabla.
Para la inserción de datos, se requiere también de la sentencia **into** y la sentencia **values**.

Es posible insertar una nueva fila en una tabla de dos formas distintas:

```
INSERT INTO nombre_tabla
VALUES (valor1, valor2, valor3, .)

INSERT INTO nombre_tabla (columna1, columna2, columna3,.)
VALUES (valor1, valor2, valor3, .)
```

Ejemplo:

Dada la siguiente tabla personas:

nombre	apellido1	apellido2
ANTONIO	PEREZ	GOMEZ
LUIS	LOPEZ	PEREZ
ANTONIO	GARCIA	BENITO

Si queremos insertar una nueva fila en la tabla personas, lo podemos hacer con cualquiera de las dos sentencias siguientes:

```
INSERT INTO personas
VALUES ('PEDRO', 'RUIZ', 'GONZALEZ')
INSERT INTO personas (nombre, apellido1, apellido2)
VALUES ('PEDRO', 'RUIZ', 'GONZALEZ')
```

Cualquiera de estas sentencias anteriores produce que se inserte una nueva fila en la tabla personas, quedando así dicha tabla:

nombre	apellido1	apellido2
ANTONIO	PEREZ	GOMEZ
LUIS	LOPEZ	PEREZ
ANTONIO	GARCIA	BENITO
PEDRO	RUIZ	GONZALEZ

UPDATE

La cláusula update permite modificar el contenido de un registro o fila.
Para hacer la actualización de un registro se necesita completar la instrucción con las sentencias **set** y **where**.

La sintaxis de SQL UPDATE es:

```
UPDATE nombre_tabla
SET columna1 = valor1, columna2 = valor2
WHERE columna3 = valor3
```

La cláusula SET establece los nuevos valores para las columnas indicadas.

La cláusula WHERE sirve para seleccionar las filas que queremos modificar.

Ojo: Si omitimos la cláusula WHERE, por defecto, modificará los valores en todas las filas de la tabla.

Ejemplo del uso de SQL UPDATE

nombre	**apellido1**	**apellido2**
ANTONIO	PEREZ	GOMEZ
LUIS	LOPEZ	PEREZ
ANTONIO	GARCIA	BENITO
PEDRO	RUIZ	GONZALEZ

Si queremos cambiar el apellido2 'BENITO' por 'RODRIGUEZ' ejecutaremos:

```
UPDATE personas
SET apellido2 = 'RODRIGUEZ'
WHERE nombre = 'ANTONIO'
AND apellido1 = 'GARCIA'
AND apellido2 = 'BENITO'
```

Ahora la tabla 'personas' quedará así:

nombre	**apellido1**	**apellido2**
ANTONIO	PEREZ	GOMEZ
LUIS	LOPEZ	PEREZ
ANTONIO	GARCIA	**RODRIGUEZ**
PEDRO	RUIZ	GONZALEZ

DELETE

La cláusula delete, elimina uno, varios o todos los registros de una tabla.

Para hacer la eliminación de algunos registros se debe complementar con la cláusula **where**, y para eliminar todos los registros de una tabla basta con usar el nombre de la tabla. Siempre se requiere complementar con la cláusula **from**.

La sintaxis de SQL DELETE es:

```
DELETE FROM nombre_tabla
WHERE nombre_columna = valor
```

Si queremos borrar todos los registros o filas de una tabla, se utiliza la sentencia:

```
DELETE * FROM nombre_tabla;
```

Ejemplo de SQL DELETE para borrar una fila de la tabla personas

nombre	**apellido1**	**apellido2**
ANTONIO	PEREZ	GOMEZ
LUIS	LOPEZ	PEREZ
ANTONIO	GARCIA	**RODRIGUEZ**
PEDRO	RUIZ	GONZALEZ

Si queremos borrar a la persona LUIS LOPEZ PEREZ, podemos ejecutar el comando:

```
DELETE FROM personas
WHERE nombre = 'LUIS'
AND apellido1 = 'LOPEZ'
AND apellido2 = 'PEREZ'
```

La tabla 'personas' resultante será:

nombre	**apellido1**	**apellido2**
ANTONIO	PEREZ	GOMEZ
ANTONIO	GARCIA	**RODRIGUEZ**
PEDRO	RUIZ	GONZALEZ

Sintaxis de SQL

La sintaxis en el lenguaje de SQL se define como el conjunto de reglas que deben seguirse al escribir el código de consultas estructuradas para considerarse como correctas y así completar la ejecución exitosamente.

Estas reglas permiten que se use de forma adecuada cada sentencia en las instrucciones sql, de esta forma si una cláusula está mal escrita, o mal ubicada, el sistema de gestión de base de datos arroja un error de sintaxis, el cual alerta de corregir lo necesario para obtener el resultado esperado.

Clausula Select y From

Una cláusula SELECT se usa para especificar los nombres de los campos que contienen los datos que quiere usar en una consulta. También puede usar operaciones matemáticas, u operaciones con funciones especiales, en lugar de o además de los campos. Incluso puede usar otra instrucción SELECT como campo, esto se conoce como una subconsulta.

Supongamos que quiere saber el número de teléfono de sus clientes. Suponiendo que el campo que almacena los números de teléfono de los clientes se denomina txtCustPhone, la cláusula SELECT es la siguiente:

SELECT txtCustPhone

La cláusula FROM permite especificar las tablas que contienen los campos que se usarán en la cláusula SELECT.

SELECT txtCustPhone FROM Customers

Si la instrucción SQL tiene dos o más campos con el mismo nombre, debe agregar el nombre del origen de datos de cada campo al nombre del campo en la cláusula SELECT. Use el mismo nombre del origen de datos que se usa en la cláusula FROM o asigne un alias al mismo.

Si quiere incluir todos los campos de un origen de datos, puede enumerarlos todos de forma individual en la cláusula SELECT, o bien puede usar el carácter comodín de asterisco (*). Cuando use el asterisco, el SGBD determina al ejecutar la consulta qué campos contiene el origen de datos e incluye todos esos campos en la consulta. Esto ayuda a asegurarse de que la consulta se mantiene actualizada si se agregan nuevos campos al origen de datos.

Puede usar el asterisco con uno o varios orígenes de datos en una instrucción SQL. Si usa el asterisco y hay varios orígenes de datos, debe incluir el nombre del origen de datos con el asterisco, para que Access pueda determinar desde qué origen de datos se incluyen los campos.

Por ejemplo, supongamos que quiere seleccionar todos los campos de la tabla Pedidos, pero solo la dirección de correo de la tabla Contactos. La cláusula SELECT podría ser similar a esta:

SELECT Pedidos.*, Contactos.Email_Address From…

Seleccionar valores distintos

Si sabe que la instrucción va a seleccionar datos redundantes y en su lugar preferiría ver solo valores diferentes, puede usar la palabra clave DISTINCT en la cláusula SELECT. Por ejemplo, supongamos que los clientes representan distintos intereses, algunos de los cuales usan el mismo número de teléfono. Si quiere asegurarse de que cada número de teléfono solo se vea una vez, la cláusula SELECT sería la siguiente: