TEMA 7. Lenguaje Manipulación Datos SQL.

Insert Update Delete

1. INTRODUCCIÓN.

El Lenguaje de Manipulación de Datos, nos permite realizar la gestión de los datos, es decir, realizar altas, bajas, modificaciones y consultas de los datos. Hasta ahora nosotros sólo hemos realizado consultas.

En este tema vamos a completar el estudio del Lenguaje de Manipulación de Datos, estudiante la **inserción** de datos, la **modificación** de los datos existentes y el **borrado** de los datos.

2. INSERCIÓN DE REGISTROS.

La sentencia `INSERT` permite la inserción de nuevas filas o registros en una tabla existente.

El formato más sencillo de utilización de la sentencia `INSERT` tiene la siguiente sintaxis:

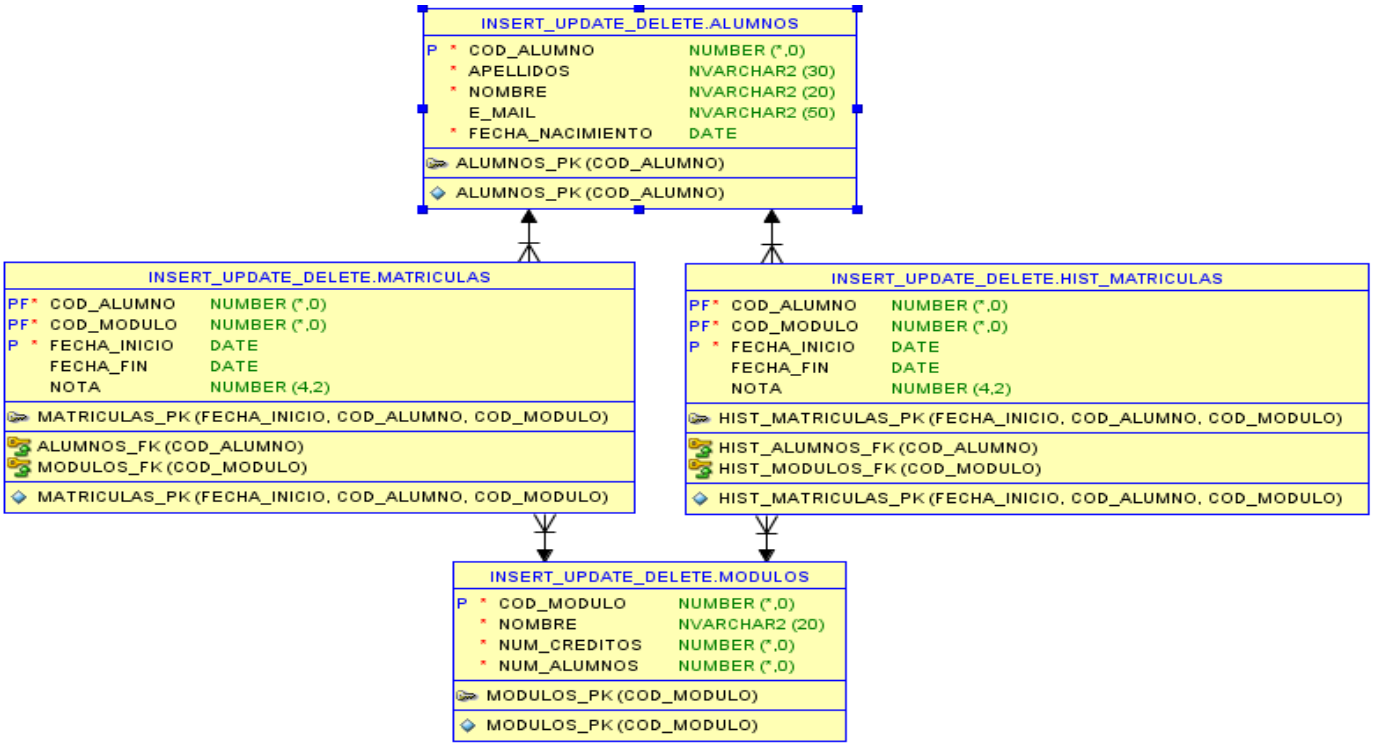
INSERT INTO nombre_tabla (lista_campos) VALUES (lista_valores);

Donde:

- ✔ **nombre_tabla:** Será el nombre de la tabla en la que quieras añadir nuevos registros.
- ✔ **lista_campos:** Se indicarán los campos de dicha tabla en los que se desea escribir los nuevos valores indicados en lista_valores. Es posible omitir la lista de campos (lista_campos), si se indican todos los valores de cada campo y en el orden en el que se encuentran en la tabla.

Para realizar correctamente la inserción de los datos:

- ✔ Tanto la lista de campos *lista_campos* como la de valores *lista_valores*, tendrán separados por comas cada uno de sus elementos.
- ✔ Cada campo de *lista_campos* debe tener un valor válido en la posición correspondiente de la *lista_valores*. Los campos y los valores se asocian por la posición.
- ✔ Tendremos que utilizar los identificadores correspondientes de la tabla de la que procedan los campos de la *lista_campos* y los tipos datos de la *lista_valores*, tendrán que corresponder con los tipos de los datos de la tabla de la que procedan los datos.
- ✔ Cualquier columna que no se encuentre en la lista de columnas recibirá el valor NULL, siempre y cuando no esté definida como NOT NULL, en cuyo caso INSERT fallará.
- ✔ Si no se escribe la lista de columnas, se han de introducir valores en todas las columnas.



Veamos unos ejemplos:

Añadir un nuevo empleado a la tabla de empleados.

```
INSERT INTO ALUMNOS (COD_ALUMNO, APELLIDOS, NOMBRE, E_MAIL, FECHA_NACIMIENTO)
   VALUES (21,'HUERTAS','MIGUEL', 'HUESTAS@GMAIL.COM','12/10/1955');
```

Podríamos realizar la misma inserción de la siguiente manera:

```
INSERT INTO ALUMNOS VALUES (21, 'HUERTAS','MIGUEL', 'HUESTAS@GMAIL.COM', '12/10/1955');
```

Añadir un nuevo departamento a la tabla de departamentos:

```
INSERT INTO MODULOS (COD_MODULO, NOMBRE, NUM_CREDITOS, NUM_ALUMNOS)
VALUES (10,'DISEÑO GRAFICO', 5, 15);
```

Podríamos realizar la misma inserción de la siguiente manera:

```
INSERT INTO MODULOS VALUES (10,'DISEÑO GRAFICO', 5, 15);
```

De esta forma podemos insertar nuevos registros en las tablas uno a uno, pero que sucede si necesitamos realizar un alta masiva de datos en una tabla, que proceden de otras tablas.

2.1. INSERCIÓN CON SELECT

En este caso necesitamos disponer de todos los datos antes de realizar la inserción, esto lo podemos hacer mediante una consulta y posteriormente insertar los datos en la tabla. (La estructura es similar a una subconsulta).

```
INSERT INTO NOMBRETABLA1 [(COLUMNA [, COLUMNA] ...)]
    SELECT {COLUMNA [, COLUMNA] ... | *}
    FROM NOMBRETABLA2 [CLÁUSULAS DE SELECT];
```

Veamos unos ejemplos:

Añadir a la tabla histórico de matrículas, las matrículas del módulo 1.

```
INSERT INTO HIST_MATRICULAS
```

```
SELECT COD_ALUMNO, COD_MODULO, FECHA_INICIO, FECHA_FIN, NOTA
FROM MATRICULAS
WHERE COD_MODULO = 1;
```

Observa que no ponemos la cláusula VALUES

Como resultado se han insertado todos los registros de la tabla de matrículas que tengan módulo 1, en nuestro caso 4 registros.

Esto nos permite agilizar la inserción de datos en una tabla.

Un elemento a tener en cuenta es que los campos que obtengamos como resultado de la consulta deben ser igual en número y tipo que los campos de la tabla donde queremos insertar los datos, además de encontrarse en el mismo orden.

Pero que pasaría si necesitamos añadir información que obtenemos de una consulta con otras informaciones que no podemos obtener de una consulta. Si observamos los datos de la tabla de que acabamos de insertar observamos que la fecha de finalización y la nota final, toman valores NULL. Vamos a realizar una inserción para añadir los registros del módulo 8 de la tabla matriculas en la tabla histórica de matrículas, pero en este caso vamos a poner como fecha de finalización del curso, la fecha de hoy y como nota le vamos a poner un 5 a todos los alumnos.

```
INSERT INTO HIST_MATRICULAS (COD_ALUMNO, COD_MODULO, FECHA_INICIO, FECHA_FIN, NOTA)
    SELECT COD_ALUMNO, COD_MODULO, FECHA_INICIO, SYSDATE, 5
    FROM MATRICULAS
    WHERE COD_MODULO = 8;
```

Obtendríamos el mismo resultado de la siguiente forma:

```
INSERT INTO HIST_MATRICULAS
    SELECT COD_ALUMNO, COD_MODULO, FECHA_INICIO, SYSDATE, 5
    FROM MATRICULAS
    WHERE COD_MODULO = 8;
```

Una tarea que podemos tener que realizar es el volcado de toda la información de una tabla en otra. Si la tabla histórica de matrículas estuviera totalmente vacía, podríamos pasar toda la información de la tabla de matriculas de la siguiente forma:

```
INSERT INTO HIST_MATRICULAS SELECT * FROM MATRICULAS;
```

3. MODIFICACIÓN DE REGISTROS.

La sentencia UPDATE permite modificar una serie de valores de determinados registros de las tablas de la base de datos.

La manera más sencilla de utilizar la sentencia UPDATE tiene la siguiente sintaxis:

```
UPDATE NOMBRE_TABLA
    SET NOMBRE_CAMPO = VALOR [, NOMBRE_ CAMPO = VALOR]...
    [WHERE CONDICIÓN];
```

Donde nombre_tabla será el nombre de la tabla en la que quieras modificar datos. Se pueden especificar los nombres de campos que se deseen de la tabla indicada. A cada campo especificado se le debe asociar el nuevo valor utilizando el signo = Cada emparejamiento campo=valor debe separarse del siguiente utilizando comas (,).

La cláusula WHERE seguida de la condición es opcional (como pretenden indicar los corchetes). Si se indica, la actualización de los datos sólo afectará a los registros que cumplen la condición. Por tanto, ten en cuenta que, si no indicas la cláusula WHERE, los cambios afectarán a todos los registros.

Veamos unos ejemplos:

Modificar la nota, incrementándola en uno, de aquellos módulos que se hayan finalizado:

UPDATE HIST_MATRICULAS SET NOTA=NOTA+1;

Puede que ahora estemos pensando en varias situaciones:

- ✔ ***Qué pasa con los alumnos que tenía un 10.***
- ✔ ***Qué pasaría si realizáramos la consulta sobre la tabla de matrículas.***

¿Cómo podemos solucionarlo?

Como tendrías que realizar la modificación de la nota de todos los alumnos matriculados actualmente para que tengan un 4.9:

UPDATE MATRICULAS SET NOTA = 4.9;

En este otro ejemplo puedes ver la actualización de dos campos, poniendo la nota a 5 y la fecha de finalización de curso con la fecha de hoy:

UPDATE MATRICULAS SET NOTA = 5, FECHA_FIN = SYSDATE;

Para que los cambios afecten a determinados registros hay que especificar una condición. Por ejemplo, cambiar la nota de los alumnos del modulo 1 y que tengan como nueva nota un 7.

UPDATE MATRICULAS SET NOTA = 7 WHERE COD_MODULO=1;

Cuando termina la ejecución de una sentencia UPDATE, se muestra la cantidad de registros (filas) que han sido actualizadas, o el error correspondiente si se ha producido algún problema.

3.1. MODIFICACIÓN CON SELECT

```
UPDATE NOMBRE_TABLA
   SET NOMBRE_CAMPO = VALOR [, NOMBRE_ CAMPO = VALOR]...
     WHERE CAMPO = SELECT {COLUMNA [, COLUMNA] ... | *}
     FROM NOMBRE_TABLA2 [CLÁUSULAS DE SELECT];
```

Vamos a ver un ejemplo

Vamos a modificar la fecha de finalización, para que la finalización de los cursos (fecha fin de la tabla MATRICULAS) sea dentro de tres días, de aquellos MÓDULOS que tengan más de tres alumnos.

```
UPDATE MATRICULAS SET FECHA_FIN= SYSDATE + 3
WHERE COD_MODULO IN (SELECT COD_MODULO
                     FROM MODULOS
                     WHERE NUM_ALUMNOS > 3);
```

Vamos a modificar la consulta anterior para que la fecha de finalización sea igual a la fecha de hoy, pero solo vamos a modificar la fecha de finalización de los cursos que, a los que anteriormente les hayamos dado fecha de finalización.

```
UPDATE MATRICULAS SET FECHA_FIN= SYSDATE
WHERE FECHA_FIN IS NOT NULL AND
      COD_MODULO IN (SELECT COD_MODULO
                     FROM MODULOS
                     WHERE NUM_ALUMNOS > 3);
```

3.2. MODIFICACIÓN DE CAMPOS CON SELECT

```
UPDATE NOMBRE_TABLA
SET NOMBRE_CAMPO1 = (SELECT NOMBRE_CAMPO1 …),
    NOMBRE_CAMPO2 = (SELECT NOMBRE_CAMPO2 …)
WHERE condición;
```

Veamos un ejemplo:

Modificar el módulo de sistemas para que tenga el mismo numero de alumnos que el módulo que alumnos tenga y la media de créditos de todos los módulos.

```
UPDATE MODULOS SET
      NUM_ALUMNOS = (SELECT(MAX(NUM_ALUMNOS)) FROM MODULOS),
      NUM_CREDITOS = (SELECT AVG(NUM_CREDITOS) FROM MODULOS)
WHERE NOMBRE LIKE 'MONTAJE';
```

También podemos hacer la modificación de varios campos al mismo tiempo con una consulta. En este caso el resultado de la consulta y los campos a consultar deben coincidir en número de campos, orden y tipo.

```
UPDATE NOMBRE_TABLA1
SET (NOMBRE_CAMPO1, NOMBRE_CAMPO2, NOMBRE_CAMPO2, …)
    =  (NOMBRE_CAMPO1, NOMBRE_CAMPO2, NOMBRE_CAMPO2, FROM TABLA1 [clausulas
SELECT])
[WHERE condición];
```

Veamos un ejemplo.

Vamos a actualizar los créditos y el número de alumnos del módulo 6 con los datos del módulo 7

```
UPDATE MODULOS
SET (NUM_CREDITOS, NUM_ALUMNOS)
  = (SELECT NUM_CREDITOS, NUM_ALUMNOS FROM MODULOS WHERE COD_MODULO = 7 )
WHERE COD_MODULO=6;
```

Para poder realizar este tipo de **consultas es muy importante tener en cuenta que la subconsulta sólo puede devolvernos un valor.**

También hay que tener en cuenta que podemos modificar sólo un registro o bien un conjunto dependiendo de la cláusula WHERE del UPDATE.

4. BORRADO DE REGISTROS.

La sentencia DELETE es la que permite eliminar o borrar registros de una tabla.

Esta es la sintaxis que debes tener en cuenta para utilizarla:

```
DELETE FROM nombre_tabla [ WHERE condición];
```

Al igual que hemos visto en las sentencias anteriores, nombre_tabla hace referencia a la tabla sobre la que se hará la operación, en este caso de borrado. Se puede observar que la cláusula WHERE es opcional. Si no se indica, debes tener muy claro que se borrará todo el contenido de la tabla, aunque la tabla seguirá existiendo con la estructura que tenía hasta el momento. Por ejemplo, si usas la siguiente sentencia, borrarás todas las matrículas de la tabla MATRICULAS:

```
DELETE FROM MATRICULAS;
```

Para ver un ejemplo de uso de la sentencia DELETE en la que se indique una condición, supongamos que SOLO queremos eliminar las matrículas de los alumnos del módulo 1 de la tabla de MATRICULAS.

```
DELETE FROM MATRICULAS WHERE COD_MODULO=1;
```

Como resultado de la ejecución de este tipo de sentencia, se obtendrá un mensaje de error si se ha producido algún problema, o bien, el número de filas que se han eliminado.

4.1. BORRADO CON SELECT

Veamos un ejemplo

Borrar las matriculas actuales de los alumnos que hayan nacido a partir de los 90.

```
DELETE FROM MATRICULAS
WHERE COD_ALUMNO IN
  (SELECT COD_ALUMNO
  FROM ALUMNOS
  WHERE ALUMNOS.FECHA_NACIMIENTO > '01/01/1990');
```