Ordenes DML. (Manipulación de datos).

Este tipo de comandos nos permiten manipular una serie de filas de una tabla. Son tres:

INSERT.- Nos permite la inserción de datos en una tabla.

UPDATE.- Nos permite modificar datos en una tabla.

DELETE.- Nos borra datos en una tabla.

Ordenes DML. (Manipulación de datos).

Este comando añade filas a una tabla o a una vista. Su formato es el siguiente:

```
INSERT  INTO tabla
(columnas)
VALUES (valores)
```

Podemos introducir datos en una tabla de nuestro propio esquema o bien disponer del **privilegio INSERT** sobre la tabla en cuestión.

Por ejemplo, si quisiéramos insertar un nuevo curso teclearíamos la siguiente orden:

```
INSERT  INTO  curso
VALUES (5,'Programación visual');
```

Como vemos el orden y el tipo de las columnas de la tabla deben ser respetados, porque si no la inserción fallaría. Debemos introducir valores en todas las columnas. Si alguno de los datos es nulo, **podemos introducir el valor NULL.**

Si queremos introducir valores solo en algunas columnas, podemos especificar exactamente cuales son esas columnas. Sin embargo, no podemos omitir en un INSERT una columna cuya cláusula sea NOT NULL. En ese caso es obligatorio la inserción porque al no admitir esa columna como vacía fallaría el proceso. Por ejemplo viendo la descripción de la tabla ALUMNOS podemos observar que tiene dos columnas que no son obligatorias, en concreto el apellido segundo y becario.

```
SQL > desc alumnos
Name                Null?          Type
------------------  ---------------        --------------
CÓDIGO              NOT  NULL   NUMBER(5)
APE1                NOT  NULL   CHAR(20)
APE2                            CHAR(20)
NOMBRE              NOT  NULL   CHAR(20)
CODIGO_CURSO        NOT  NULL   NUMBER(6)
MATRICULA                NOT  NULL   NUMBER(6)
BECARIO                         CHAR(1)
TIPO                NOT  NULL   CHAR(15)
```

```
SQL> insert into alumnos
  2  (código,ape1,nombre,código_curso,matrícula,tipo)
  3  values
  4  (150, 'Ramos','Raul',3,135000,'BACHILLER');

1 row created.
```

Si tuviéramos una columna de tipo fecha deberíamos utilizar la función TO_DATE para convertir el dato antes de utilizarlo.

```
INSERT  INTO  TABLA
VALUES(10,'dato', TO_DATO('10-ENE-95');
```

Insertando datos desde otra tabla.- Podemos insertar datos de otra tabla, haciendo una de igual estructura. Se puede utilizar la cláusula SELECT para realizar dicha operación. Su formato es el siguiente:

```
INSERT  INTO  tabla
SELECT * from tabla1;
```

Podemos especificar columnas determinadas y en la SELECT podemos utilizar la cláusula WHERE. Supongamos que tenemos una tabla temporal llamada alumnos_tem donde están almacenados los datos de los alumnos del año que viene. Si quisiéramos incorporarla a la nuestra teclearíamos lo siguiente:

```
INSERT INSTO alumnos
SELECT *from alumnos_tem
```

Si solo quisiéramos traer a los alumnos que pagan de matrícula más de 150000 pesetas haríamos los siguientes:

```
INSERT INTO alumnos
SELECT * FROM alumnos_tem
        WHERE matrícula > 150000;
```

Modificaciones. Orden Update

Para realizar modificaciones sobre filas ya existentes en una tabla utilizamos el comando UPDATE. Su formato es el siguiente:

```
UPDATE tabla
SET columna=valor
WHERE condición
```

- Se modifican todas las filas que cumplan la condición WHERE.

- Si no especificamos esta cláusula se modifican todas.

- Se pueden utilizar expresiones para asignar valores a las columnas.

- Podemos utilizar una subconsulta para aplicar un nuevo valor.

- Esta subconsulta deberá devolver un solo valor.

- Podemos tener varias cláusulas SET.
Por ejemplo si queremos cambiar el título del alumno apellidado 'Ramos' a 'Ramas';

```
SQL > update alumnos
  2   set ape1='Ramas'
  3  where ape1='Ramos';

 1 row updated.
```

Si queremos poner como becarios a todas aquellos alumnos que paguen menos de 150 dolares de matrícula.

```
SQL > update alumnos
  2 set becario='S'
  3 where matrícula < 150;

 7 rows updated.
```

Si queremos poner becario a NULL para todas aquellos que paguen más de 150 dolares .

```
SQL > update alumnos
   2 set becario=NULL
   3 where matrícula >= 150;
```

3 rows updated.

Si quisiéramos incrementar la matrícula en 15% para el año que viene:

```
1 update alumnos
2* set matrícula=matrícula+(matrícula*0.15)

10 rows updated.
```

Podemos utilizar una orden SELECT para realizar la modificación. El formato sería el siguiente:

```
SELECT tabla
SET columna = (SELECT columna FROM tabla)
WHERE condición
```

También podemos utilizarla en la cláusula WHERE. Por ejemplo si queremos actualizar un 15% las matrículas de todos los alumnos que tengan la misma titulación que el alumno apellidado Ramas:

```
SQL > update alumnos
  2 set matrícula=matrícula*(matrícula*0.15)
  3 where tipo= (select tipo from alumnos
  4                 where ape1='Ramas');

4 rows updated.
```

Borrados. Orden Delete.

Para eliminar determinados registros usamos la orden DELETE. Su formato es el siguiente:

```
DELETE FROM tabla
WHERE condición;
```

Debemos tener cuidado a la hora de establecer la cláusula de condición de borrado. Si no especificamos ninguna borraremos todos los datos de la tabla. Por ejemplo si queremos borrar todos los que tengan como titulación 'Licenciados'.

```
SQL > delete from alumnos
  2 where tipo='LICENCIADO';

 2 rows deleted.
```

Control de transacciones

Para controlar las transacciones que realizamos disponemos de dos órdenes muy importantes:

- COMMIT.- Hace permanentes todos los cambios realizados desde el último commit.

- ROLLBACK.- Deshace los cambios desde el último COMMIT.

Por ejemplo si tecleamos equivocadamente la siguiente orden:

DELETE FROM alumnos;

Podemos arreglarlo tecleando inmediatamente:

ROLLBACK;

Las órdenes COMMIT y ROLLBACK solo afectan a nivel de usuario, no de sistema. Un fallo del sistema lleva implícito un ROLLBACK.