TEMA 7 (Parte 1)

LENGUAJE DE MANIPULACIÓN DE DATOS (DML)

INSERT, UPDATE, DELETE

DML
INSERT
UPDATE
DELETE
SELECT

CURSO 23-24

ÍNDICE

0.- INTRODUCCIÓN

RECORDANDO...

- El lenguaje de manipulación de datos (DML) se ocupa de:
  - Consultar
  - Insertar
  - Modificar
  - Eliminar

DATOS

DML (MANIPULACIÓN DATOS)	
SENTENCIA	**DESCRIPCION**
SELECT	CONSULTA datos
INSERT	INSERCIÓN datos
UPDATE	MODIFICACIÓN datos
DELETE	BORRADO datos

0.- INTRODUCCIÓN

- Obviamente cuando se crea por primera vez un esquema lógico está vacío (no tiene datos)
- Lo normal es que este esquema lógico vacío evolucione a través de los datos que se insertan, modifican y eliminan de sus tablas.
  - Es decir siendo utilizado por los usuarios diariamente a través de una aplicación determinada.
- En este tema veremos las sentencias SQL que hacen evolucionar el estado de la BD (esquema lógico)

SENTENCIA	DESCRIPCION
INSERT	INSERCIÓN datos
UPDATE	MODIFICACIÓN datos
DELETE	BORRADO datos

1.- INSERCIÓN DE DATOS

```
INSERT INTO tabla [(comalista_atributo)]
VALUES comalista_valores| sentencia_ select
```

valores::= valor| NULL| DEFAULT

- La claúsula VALUES **añade una única tupla o fila**:
  - comalista_atributo representa la lista de atributos donde se van a introducir valores
    - Si se omite se asume que se introducen valores en todas.
  - comalista_*valores* representa la lista de valores que se van a introducir en comalista_atributo
    - comalista_*valores* debe coincidir en número y tipo con cada uno de los atributos especificados en comalista_atributo
  - Los atributos de tabla que no aparecen en comalista_atributo toman el valor NULL:
    - Si en alguno existe restricción de valor no nulo INSERT fallará.
- La opción *sentencia_ select* **permite añadir varias filas a la vez**.

1.- INSERCIÓN DE DATOS

◎ **Ejemplos**: (esquema ciclismo)

CICLISTA (dorsal, nombre, edad, nomeq)[1]
CP= {dorsal}
VNN= {nombre, nomeq}

CICLISTA

dorsal	nombre	edad	nomeq
1	Indurain	32	Banesto
2	Delgado	35	Banesto
3	Zulle	27	Once

1
```
INSERT INTO ciclista
VALUES
 (101,
 'Peris',
 20,
 'Kelme')
```

dorsal	nombre	edad	nomeq
1	Indurain	32	Banesto
2	Delgado	35	Banesto
3	Zulle	27	Once
101	Peris	20	Kelme

2
```
INSERT INTO ciclista
 (dorsal,
 nombre,
 nomeq)
VALUES
 (101,
 'Peris',
 'Kelme')
```

dorsal	nombre	edad	nomeq
1	Indurain	32	Banesto
2	Delgado	35	Banesto
3	Zulle	27	Once
101	Peris		Kelme

3
```
INSERT INTO ciclista
 (dorsal,
 nombre,
 edad)
VALUES
 (101,
 'Peris',
 20)
```

ERROR!

1: se omiten los dominios por simplificar (son los ya vistos).

1.- INSERCIÓN DE DATOS

◎ **Ejemplos**: (esquema ciclismo)

NUEVOS_CICLISTAS (dorsal, nombre, edad, nomeq, incluir: booleano)
CP= {dorsal}
VNN= {nombre, nomeq}

NUEVOS_CICLISTAS

dorsal	nombre	edad	nomeq	incluir
102	Fernandez	18	Banesto	NO
103	Silon	20	Kelme	SI
104	Antunez	27	Banesto	NO
105	Doin	24	Once	SI

```
INSERT INTO ciclista
SELECT    dorsal, nombre, edad, nomeq
FROM      nuevos_ciclistas
WHERE     incluir='SI'
```

dorsal	nombre	edad	nomeq
1	Indurain	32	Banesto
2	Delgado	35	Banesto
103	Silon	20	Kelme
105	Doin	24	Once

Conviene tener en cuenta que un INSERT puede fallar si se violan las restricciones de la BD (no nulos, claves primarias, ajenas, etc.)

2.- MODIFICACIÓN DE DATOS

```
UPDATE  tabla
SET     comalista_asignacion
[WHERE condicion]
```

asignacion::= atributo= {*ref_columna*| NULL| DEFAULT}

◎ Se modifica el valor de los atributos especificados en comalista_*asignacion* para aquellas filas de tabla que cumplan la *condicion* WHERE

- El nuevo valor de un atributo puede ser: *ref_columna!!*
  - atributo| constante| *expresion_aritmetica*| *ref_funcion*
  - NULL
  - valor por defecto para el atributo (si está definido).
- Si no se especifica claúsula WHERE se modifican todas las filas de la tabla.

2.- MODIFICACIÓN DE DATOS

- **Ejemplos**:

CICLISTA

dorsal	nombre	edad	nomeq
1	Indurain	32	Banesto
2	Delgado	35	Banesto
3	Zulle	27	Once

```
UPDATE  ciclista
SET     nomeq='Kelme'
WHERE   nomeq='Banesto'
```

dorsal	nombre	edad	nomeq
1	Indurain	32	Kelme
2	Delgado	35	Kelme
3	Zulle	27	Once

```
UPDATE  ciclista
SET     edad=edad-10
WHERE   edad>30
```

dorsal	nombre	edad	nomeq
1	Indurain	22	Kelme
2	Delgado	25	Kelme
3	Zulle	27	Once

```
UPDATE  ciclista
SET     edad=20,
        nomeq='Once'
```

dorsal	nombre	edad	nomeq
1	Indurain	20	Once
2	Delgado	20	Once
3	Zulle	20	Once

Conviene tener en cuenta que un UPDATE puede fallar si se violan las restricciones de la BD (no nulos, claves primarias, ajenas, etc.)

3.- ELIMINACIÓN DE DATOS

DELETE [FROM] tabla
[WHERE *condicion*]

- Se eliminan de tabla aquellas filas que cumplan la *condicion* especificada en la claúsula WHERE
  - Si no se especifica claúsula WHERE se borran todas las filas de la tabla.
  - **Ejemplo**:

CICLISTA

dorsal	nombre	edad	nomeq
1	Indurain	32	Banesto
2	Delgado	35	Banesto
3	Zulle	27	Once

```
DELETE FROM ciclista
WHERE edad>30
```

dorsal	nombre	edad	nomeq
3	Zulle	27	Once

Conviene tener en cuenta que un DELETE puede fallar si se violan las restricciones de la BD (claves ajenas, etc)

4.- CONTROL DE TRANSACCIONES

- VER FOTOCOPIA LIBRO.
- **TRANSACCIÓN**
  - **Conjunto de SENTENCIAS SQL** que modifican el estado de la BD, que deben realizarse de forma **atómica**.
    - En bloque (**o todas o ninguna**)
    - ¿Que sentencias SQL modifican el estado de la BD?:
      - Inserción, modificación, o borrado de datos
  - ¿Por qué se considera que un conjunto de sentencias SQL son una transacción?
    - Porque si no se realizan todas la BD quedaría en un estado INSCONSISTENTE.

CONSIDERACIONES FINALES

- **CARGA DE DATOS**:
  - Cuando creamos un esquema lógico es posible no comenzar a utilizarlo vacío desde cero
  - Podemos desear cargar datos ya existentes en él
    - Recrear una BD con datos
    - Cambio a otro SGBD
    - Trasvases de datos de sistemas previos...
  - En estos casos se debe **evitar que los datos a cargar violen las restricciones del esquema lógico**
    - Se deben analizar los datos a cargar y realizar minuciosamente los scripts de carga.
      - El orden en que se cargan los datos es importante
  - Algunos mecanismos del lenguaje pueden ayudar...

CARGA DE DATOS
Desactivación de restricciones

- Por ejemplo, es posible desactivar una restricción de una tabla.
- Esta opción es útil para realizar cargas masivas de datos que sabemos **previamente correctos**:
  - Por ejemplo al realizar trasvases de datos o recrear desde cero bases de datos con datos.
  - De esta forma evitamos tener que meter los datos en un orden determinado para NO violar las restricciones
    - Sobre todo claves ajenas.
    - Un ejemplo muy claro son las claves ajenas reflexivas.
  - No debe utilizarse esta opción para la inserción rutinaria de datos en una BD
    - En un entorno real de trabajo.

CARGA DE DATOS
Desactivación de restricciones

- Para desactivar una restricción de una tabla:

alteracion_tabla::= ALTER TABLE tabla
....
[{ENABLE|DISABLE} [CONSTRAINT restriccion]]

```
ALTER TABLE CICLISTA
DISABLE CONSTRAINT ciclista_PK
```

- Podemos volver a activarla de la forma:

```
ALTER TABLE CICLISTA
ENABLE CONSTRAINT ciclista_PK
```

- Por defecto, las restricciones se crean activadas, sin embargo, se pueden crear desactivadas al hacer CREATE o ALTER TABLE

```
ALTER TABLE CICLISTA
ADD CONSTRAINT nombre_UNI UNIQUE (nombre) DISABLE
```


CARGA DE DATOS
Desactivación de restricciones

- **Ejemplo**: TABLAS EMPLE Y DEPART

```
CREATE TABLE DEPART (
DEPT_NO     NUMBER(2)
            CONSTRAINT DEPART_PK PRIMARY KEY,
DNOMBRE     VARCHAR2(14),
LOC         VARCHAR2(14)
) ;

CREATE TABLE EMPLE (
EMP_NO      NUMBER(4)
            CONSTRAINT EMPLE_PK PRIMARY KEY,
APELLIDO    VARCHAR2(10) ,
OFICIO      VARCHAR2(10) ,
DIR         NUMBER(4)
            CONSTRAINT EMPLE_EMPLE_FK REFERENCES EMPLE (EMP_NO) DISABLE,
FECHA_ALT   DATE,
SALARIO     NUMBER(7,2),
COMISION    NUMBER(7,2),
DEPT_NO     NUMBER(2)
            CONSTRAINT DEPART_DEPTNO_VNN NOT NULL
            CONSTRAINT EMPLE_DEPART_FK REFERENCES DEPART (DEPT_NO)
)
```


CARGA DE DATOS
Desactivación de restricciones

- Esta carga de datos en EMPLE fallaría si la restricción estuviera activa:

```
-- TABLA DEPART:
INSERT INTO DEPART VALUES (10,'CONTABILIDAD','SEVILLA');
INSERT INTO DEPART VALUES (20,'INVESTIGACIÓN','MADRID');
INSERT INTO DEPART VALUES (30,'VENTAS','BARCELONA');
INSERT INTO DEPART VALUES (40,'PRODUCCIÓN','BILBAO');

-- TABLA EMPLE:
INSERT INTO EMPLE VALUES (7369,'SÁNCHEZ','EMPLEADO',7902,'17/12/1980', 1040,NULL,20);
INSERT INTO EMPLE VALUES (7499,'ARROYO','VENDEDOR',7698,'20/02/1980', 2080,390,30);
INSERT INTO EMPLE VALUES (7521,'SALA','VENDEDOR',7698,'22/02/1981', 1625,650,30);
INSERT INTO EMPLE VALUES (7566,'JIMÉNEZ','DIRECTOR',7839,'02/04/1981', 3867,NULL,20);
INSERT INTO EMPLE VALUES (7654,'MARTÍN','VENDEDOR',7698,'29/09/1981', 1625,1820,30);
INSERT INTO EMPLE VALUES (7698,'NEGRO','DIRECTOR',7839,'01/05/1981', 3705,NULL,30);
INSERT INTO EMPLE VALUES (7782,'CEREZO','DIRECTOR',7839,'09/06/1981', 3185,NULL,10);
INSERT INTO EMPLE VALUES (7788,'GIL','ANALISTA',7566,'09/11/1981', 3900,NULL,20);
INSERT INTO EMPLE VALUES (7839,'REY','PRESIDENTE',NULL,'17/11/1981', 6500,NULL,10);
INSERT INTO EMPLE VALUES (7844,'TOVAR','VENDEDOR',7698,'08/09/1981', 1950,0,30);
INSERT INTO EMPLE VALUES (7876,'ALONSO','EMPLEADO',7788,'23/09/1981', 1430,NULL,20);
INSERT INTO EMPLE VALUES (7900,'JIMENO','EMPLEADO',7698,'03/12/1981', 1235,NULL,30);
INSERT INTO EMPLE VALUES (7902,'FERNÁNDEZ','ANALISTA',7566,'03/12/1981', 3900,NULL,20);
INSERT INTO EMPLE VALUES (7934,'MUÑOZ','EMPLEADO',7782,'23/01/1982', 1690,NULL,10);
COMMIT;
```

CARGA DE DATOS
Desactivación de restricciones

- No debemos olvidar volver a activarla…

```
ALTER TABLE EMPLE
ENABLE CONSTRAINT EMPLE_EMPLE_FK
```

  - Si los datos no fueran consistentes la activación fallaría
- Otra opción sería no crear la restricción desde el principio y luego crearla modificando la tabla…
  - Yo lo suelo hacer así
- Insisto esto sólo tiene sentido cuando estamos muy seguros de tratar con datos correctos de antemano