

UD 4. Manipulación de Bases de Datos

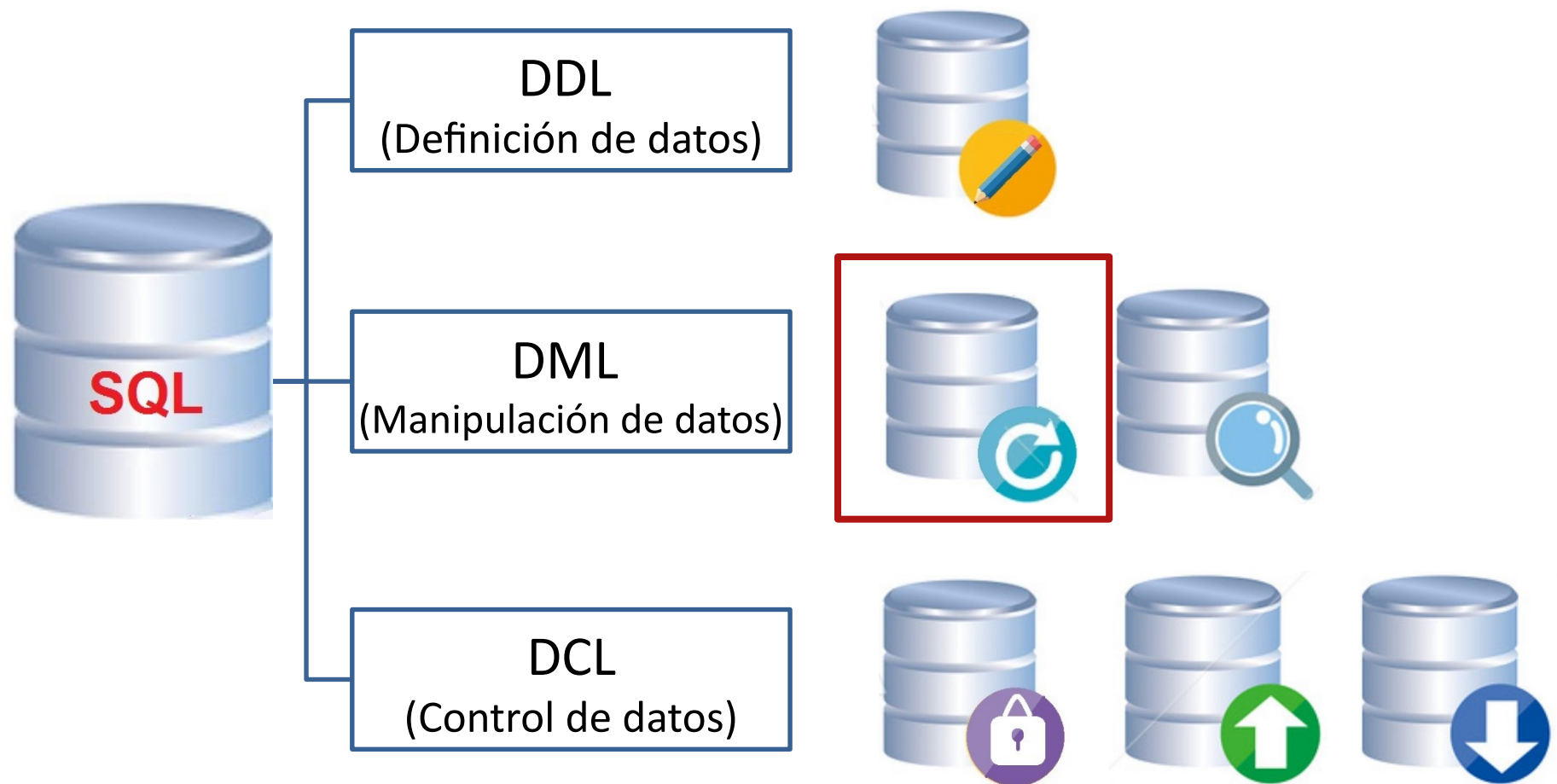


Índice

- Introducción
- INSERT: Inserción de datos
- UPDATE: Modificación de datos
- DELETE: Eliminación de datos
- Ejecución de comandos DML sobre vistas.

Introducción

Dentro de los sublenguajes SQL tenemos:



Introducción

Una vez que se ha creado de forma conveniente las tablas, el siguiente paso consiste en insertar datos en ellas, es decir, añadir tuplas.

Durante la vida de la base de datos será necesario, además, borrar determinadas tuplas o modificar los valores que contienen.

Dentro de los Sublenguajes de SQL, **DML** (Data Manipulation Language) permite hacer **actualizaciones** y consultas.

Los comandos para actualizar son:

- **INSERT**
- **UPDATE**
- **DELETE.**

INSERT: Inserción de datos

El comando **INSERT** de SQL permite introducir datos en una tabla o en una vista de la base de datos. Sintaxis:

```
INSERT INTO {nombre_tabla | nombre_vista } [(columna1 [, columna2]...)]  
VALUES (valor1 [, valor2] ... );
```

Indicando la tabla donde (**INTO**) se añaden los datos que se especifiquen tras **VALUES** en un nuevo registro.

Los valores deben corresponderse con el orden de las columnas. Si no es así se puede indicar tras el nombre de la tabla y entre paréntesis los campos.

INSERT: Inserción de datos

Ejemplos

Partiendo del siguiente diseño físico de una tabla:

```
CREATE TABLE EMPLEADOS (  
    COD NUMBER(2) PRIMARY KEY,  
    NOMBRE VARCHAR2(50) NOT NULL,  
    LOCALIDAD VARCHAR2(50) DEFAULT 'Écija',  
    FECHANAC DATE );
```

La forma más habitual de introducir datos es la siguiente:

```
INSERT INTO EMPLEADOS VALUES (1, 'Pepe', 'Osuna', '01/01/1970');  
INSERT INTO EMPLEADOS VALUES (2, 'Juan', DEFAULT, NULL);  
INSERT INTO EMPLEADOS VALUES (3, 'Sara', NULL, NULL);
```

Es obligatorio introducir valores para los campos COD y NOMBRE.
Dichos campos no pueden tener valor NULL.

INSERT: Inserción de datos

Casos especiales

- Insertar **sólo** el valor de **ciertos campos**.
Hay que indicar los campos a insertar y el orden en el que los introducimos:
INSERT INTO EMPLEADOS(NOMBRE, COD) VALUES ('Ana', 5);
- Insertar **datos** en una tabla que hayan sido obtenidos **de una consulta** realizada a otra tabla/vista u otras tablas/vistas. Su forma es:

```
INSERT INTO tabla  
SELECT ...
```

Debe respetarse lo dicho anteriormente respecto a los campos. La consulta SELECT debe devolver la misma cantidad y tipo de campos que los definidos en la tabla.

INSERT: Inserción de datos

Ejemplos

Teniendo:

```
CREATE TABLE SOLICITANTES (  
    NUM NUMBER(2) PRIMARY KEY,  
    NOMBRE VARCHAR2(50),  
    CIUDAD VARCHAR2(50),  
    NACIMIENTO DATE,  
    ESTUDIOS VARCHAR2(50) );
```

```
INSERT INTO EMPLEADOS  
SELECT NUM, NOMBRE, CIUDAD, NACIMIENTO  
FROM SOLICITANTES WHERE ESTUDIOS='CFGS DAW';
```

```
INSERT INTO EMPLEADOS(FECHANAC, NOMBRE, COD)  
SELECT NACIMIENTO, NOMBRE, NUM FROM SOLICITANTES  
WHERE ESTUDIOS='CFGS DAW';
```


UPDATE: Modificación de datos

Se utiliza el comando **UPDATE** para modificar los registro de las tablas o vistas que ya tienen datos.

Sintaxis

```
UPDATE {nombre_tabla | nombre_vista}  
SET columna1=valor1 [, columna2=valor2] ...  
[WHERE condición];
```

SET indica las columnas a modificar y sus nuevos valores.

WHERE permite especificar qué registros serán modificados.

UPDATE: Modificación de datos

Ejemplos

-- Todos los nombres a mayúsculas y todas las localidades a Estepa

UPDATE EMPLEADOS

SET NOMBRE=UPPER(NOMBRE), LOCALIDAD='Estepa';

-- Para los empleados que nacieron a partir de 1970

-- ponemos nombres con inicial mayúscula y localidades Marchena

UPDATE EMPLEADOS

SET NOMBRE=INITCAP(NOMBRE), LOCALIDAD='Marchena'

WHERE FECHANAC >= '01/01/1970';

Actualización de datos usando una subconsulta. Por ejemplo:

UPDATE empleados

SET sueldo=sueldo*1.10

WHERE id_seccion = (SELECT id_seccion FROM secciones

WHERE nom_seccion='Producción');

DELETE: Eliminación de datos

DELETE elimina los registros de la tabla que cumplan la condición indicada.



Sino existe **condición**, **elimina todos** los registros

No confundir con DROP que elimina la estructura de la tabla.

Sintaxis:

```
DELETE [ FROM ] {nombre_tabla | nombre_vista}  
[WHERE condición] ;
```

DELETE: Eliminación de datos

Ejemplos

-- Borramos empleados de Estepa

DELETE EMPLEADOS

WHERE LOCALIDAD='Estepa';

-- Borramos empleados cuya fecha de nacimiento sea anterior a 1970 y localidad sea Osuna

DELETE EMPLEADOS

WHERE FECHANAC < '01/01/1970' AND LOCALIDAD = 'Osuna';

-- Borramos TODOS los empleados;

DELETE EMPLEADOS;

DELETE: Eliminación de datos

Integridad

Ten en cuenta que **el borrado** de un registro no **puede provocar fallos de integridad**.

La opción de integridad **ON DELETE CASCADE** (clave secundaria o foránea) hace que no sólo se **borren** los **registros** indicados sino todos los **relacionados**.

En la práctica significa que **no** se pueden **borrar registros** cuya **clave primaria** sea **referenciada por** alguna **clave foránea** en otra tabla, a **no** ser que dicha tabla secundaria tenga activada la clausula **ON DELETE CASCADE** en su clave foránea, en cuyo caso se borraría el/los registro/s de la tabla principal y los registros de tabla secundaria cuya clave foránea coincide con la clave primaria eliminada en la tabla primera.

DELETE: Eliminación de datos

Con Subconsulta

Al igual que en el caso de las instrucciones INSERT o SELECT, DELETE dispone de cláusula **WHERE** y en dicha cláusulas podemos utilizar **subconsultas**.

Ejemplo: Una subconsulta creada con el operador IN. Eliminarán los empleados cuyo identificador esté dentro de la tabla operarios.

DELETE empleados

WHERE id_empleado **IN** (**SELECT** id_empleado **FROM** operarios);



No se puede borrar datos de la misma tabla que se lee.