GENERALITAT VALENCIANA
Conselleria d'Educació, Cultura i Esport

ceedcv CENTRE ESPECÍFIC D'EDUCACIÓ A DISTÀNCIA DE LA COMUNITAT VALENCIANA

UD 5. MODELO FÍSICO DML

Parte 1. Manipulación de datos

Bases de Datos (DAW/DAM)

CFGS Desarrollo de Aplicaciones Web (DAW)
CFGS Desarrollo de Aplicaciones Multiplataforma (DAM)

Abelardo Martínez y Pau Miñana
Curso 2023-2024

Créditos

- Apuntes realizados por Abelardo Martínez y Pau Miñana.
- Basados y modificados de Sergio Badal (www.sergiobadal.com) y Raquel Torres.
- Las imágenes e iconos empleados están protegidos por la licencia LGPL y se han obtenido de:
  - https://commons.wikimedia.org/wiki/Crystal_Clear
  - https://www.openclipart.org

Contenidos

¿Qué veremos en esta parte?

- Inserción de datos
- Actualización de datos
- Borrado de datos

Contenidos

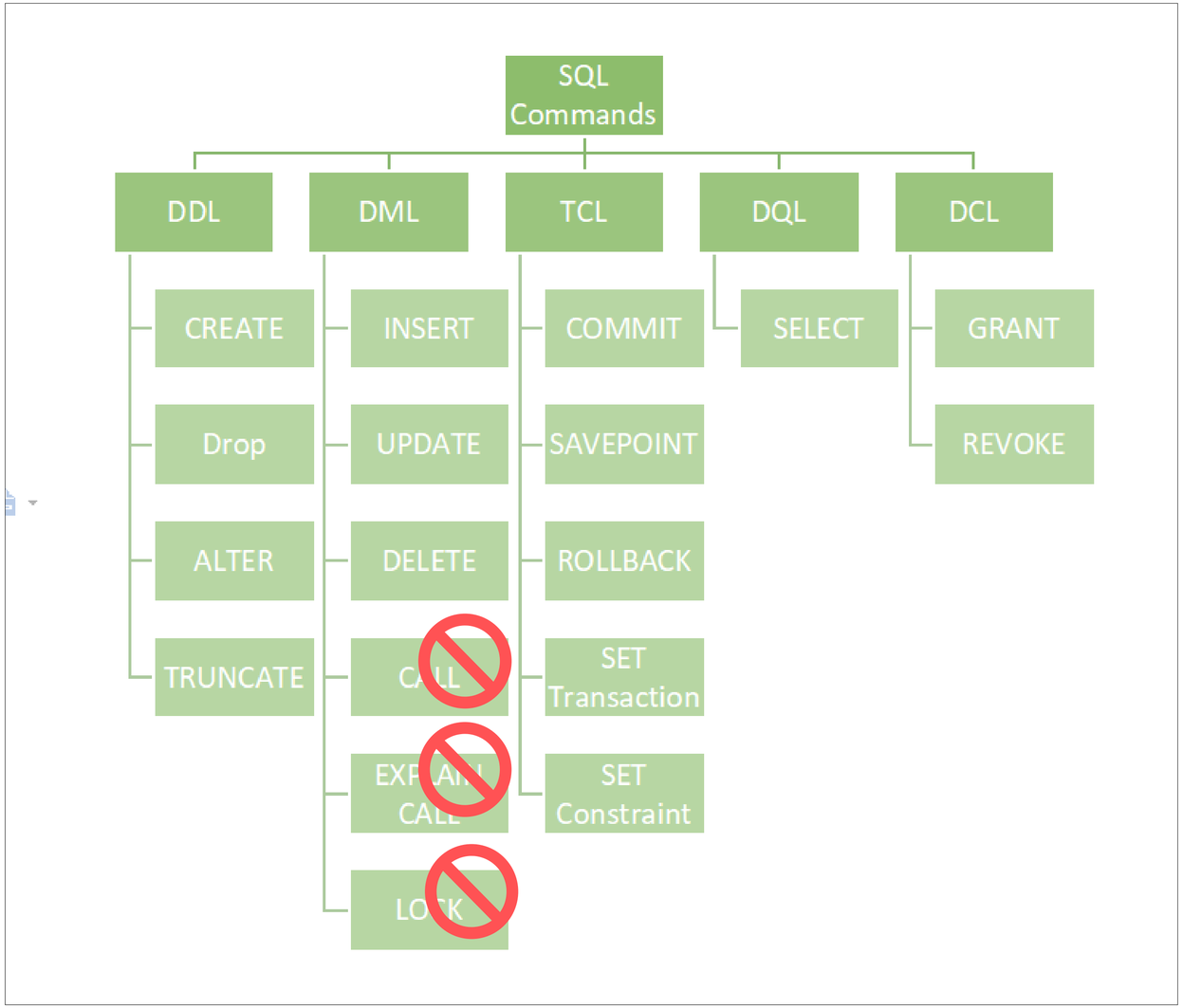
1. INSTRUCCIONES DML
2. INSERCIÓN DE DATOS
3. ACTUALIZACIÓN DE DATOS
4. BORRADO DE DATOS
5. BIBLIOGRAFÍA

1. INSTRUCCIONES DML

Instrucciones DML

Hasta ahora hemos conseguido crear y modificar los metadatos (DDL); el próximo paso será incluir registros en las tablas que hemos creado.

- El **DML** (**D**ata **M**anipulaton **L**anguage) lo forman las instrucciones capaces de modificar los datos de las tablas.
- Algunos autores incluyen un subgrupo dentro del DML al que llaman **TCL** (**T**ransaction and **C**ontrol **L**anguage) para tratar las transacciones que veremos la semana que viene.
- También se incluyen otras como CALL, EXPLAIN o LOCK que no veremos este curso.



2. INSERCIÓN DE DATOS

MySQL. Cláusula INSERT

En ella podemos distinguir las palabras reservadas **INSERT INTO** seguidas del nombre de la tabla en la que vamos a guardar los nuevos datos, pudiendo insertar uno o varios registros a la vez. Opcionalmente podemos poner entre paréntesis los nombres de los campos.

Ésta es su sintaxis:

```
INSERT INTO nombre_tabla [(nombre_col, …)]
VALUES ({expresión | DEFAULT | NULL}, …)
```

Por ejemplo (tabla **departamentos** del tema anterior):

```
INSERT INTO departamentos (cod_dpt, nombre_dpt, ubicacion)
VALUES (‘INF’. ‘Informática’, ‘Planta sótano’);
```

Lista de campos. Inclusión/no inclusión

A) La **lista de campos** a rellenar (cod_dpt, nombre, ubicacion) **se indica solo si** no queremos rellenar todos los campos o queremos indicar un orden determinado.

- Si NO conocemos el orden de los campos al hacer CREATE TABLE, indicamos el orden:

INSERT INTO departamentos (cod_dpt, nombre_dpt, ubicacion)
VALUES (‘INF’. ‘Informática’, ‘Planta sótano U3’);

B) **Si no incluimos esa lista de campos**, el SGBD entenderá que el orden es el especificado cuando creamos la tabla con CREATE TABLE, por lo que debemos conocer el orden de los campos al hacer CREATE TABLE y dar valor a **todos** los campos:

INSERT INTO departamentos
VALUES (‘INF’, ‘Informática’, ‘Planta sótano U3’);

Lista de campos. Valores

C) En cualquier caso, **siempre podremos cambiar el orden de los campos con la precaución de** cambiar también el orden de los VALUES.

INSERT INTO departamentos (nombre_dpt, ubicacion, cod_dpt)
VALUES (‘Informática’, ‘Planta sótano U3’, ‘INF’);

D) Los **campos no rellenados explícitamente con la orden INSERT**, se rellenan con su valor por defecto (DEFAULT) o bien con NULL si no se indicó valor alguno.

INSERT INTO departamentos
VALUES (‘INF’, NULL, Planta sótano U3);

Si algún campo tiene restricción de obligatoriedad (NOT NULL), ocurrirá un error si no rellenamos el campo con algún valor.

Errores más comunes de la cláusula INSERT

1) **Repetir la inserción de un registro que ya existe**. Como la CP debe ser única debe mostrar un error al intentar insertar un registro con la misma clave. Por ejemplo, insertar dos veces el mismo empleado.
   - Los códigos/id pueden hacer que se repitan registros sin notarlo, siempre que el código/id cambie.

2) Insertar registro(s) con un campo **clave ajena a otro registro que todavía no se ha insertado**. Por ejemplo, insertar un empleado asociado a un departamento que no se ha insertado aún.

3) Insertar registro(s) con **campos NOT NULL y no indicar el valor** de esos campos.

4) Insertar registro(s) con **valores que no corresponden con el tipo de datos** de ese atributo.

5) Insertar los datos de los **campos en un orden distinto** al de los mismos.
   - Si el tipo de datos es compatible no salta ningún error, lo que hace más difícil detectarlo. **Incluye siempre los nombres de los campos en el INSERT para tener claro el orden**.

Errores más comunes de la cláusula INSERT

6) Usar caracteres que **no se correspondan al juego de caracteres y/o colación** especificados.

7) Introducir textos que contengan **comillas** simples y/o dobles.
   - Se puede usar el tipo de comilla contrario para abrir y cerrar la cadena para que funcione.
   - Si un texto tiene comillas de los 2 tipos se puede usar la barra (\' o \").
   - La barra permite también introducir caracteres especiales como saltos de línea (\n).
   - Para poder introducir la barra como carácter, simplemente se dobla (\\).

3. ACTUALIZACIÓN DE DATOS

MySQL. Cláusula UPDATE

La instrucción indica que queremos actualizar (UPDATE) una tabla (nombre_tabla) estableciendo (SET) los campos que queremos modificar a una expresión o valor determinado. Podemos colocar tantos campos con sus nuevos valores separados por comas como necesitemos.

Por último, aunque es opcional, **debemos incluir la cláusula WHERE** como condición que nos permitirá seleccionar sobre qué registro/s se debe realizar la actualización. Ésta es su sintaxis:

```
UPDATE nombre_tabla
SET columna1 = {expresión | DEFAULT | NULL} [, columna2…]…
[WHERE condición]
```

Por ejemplo (tabla **departamentos** del tema anterior):

```
UPDATE departamentos
SET ubicacion = 'Planta sótano U3'
WHERE cod_dpt = 'INF';
```

Operadores condición WHERE

La condición WHERE puede llegar a ser extremadamente compleja pero esto lo veremos más adelante cuando veamos las sentencias SELECT (DQL). Lo habitual es que haga referencia, al menos, a algún campo de la tabla. Se puede utilizar cualquiera de los siguientes operadores de comparación y de lógica:

Operador	Significado
>	Mayor que
<	Menor que
>=	Mayor o igual que
<=	Menor o igual que
=	Igual
!=	Distinto
<>	Distinto

Operador	Significado
AND	Devuelve verdadero si las expresiones a su izquierda y derecha son ambas verdaderas.
OR	Devuelve verdadero si cualquiera de las dos expresiones a izquierda y derecha del OR, son verdaderas.
NOT	Invierte la lógica de la expresión que está a su derecha. Por ejemplo, si era verdadera, mediante NOT pasa a ser falso, y viceversa.

Por ejemplo:

```
UPDATE departamentos
SET ubicacion = 'Planta cuarta U5'
WHERE cod_dpt = 'MKT' OR cod_dpt = 'ALM';
```

Condición WHERE. Aspectos a tener en cuenta

¡CUIDADO!

Si no ponemos el WHERE, la ACTUALIZACIÓN se realizará en todos los registros de la tabla.

Por ejemplo:

```
UPDATE departamentos
SET ubicacion = 'Planta cuarta U5'
WHERE cod_dpt = 'MKT';
```

4. BORRADO DE DATOS

MySQL. Cláusula DELETE

Para eliminar un registro utilizaremos la instrucción DELETE.

Como ocurría con UPDATE, aunque es opcional, **debemos incluir la cláusula WHERE** como condición que nos permitirá seleccionar sobre qué registro/s se debe realizar el borrado. Ésta es su sintaxis:

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

Por ejemplo (tabla **departamentos** del tema anterior):

```
DELETE FROM departamentos
WHERE cod_dpt = 'MKT';
```

Condición WHERE. Aspectos a tener en cuenta

¡CUIDADO!

Si no ponemos el WHERE, la instrucción BORRARÁ todos los registros de la tabla.

Por ejemplo:

```
DELETE FROM departamentos
WHERE cod_dpt = 'MKT';
```

5. BIBLIOGRAFÍA

Recursos

- MySQL 8.0 Reference Manual. https://dev.mysql.com/doc/refman/8.0/en/
- Oracle Database Documentation. https://docs.oracle.com/en/database/oracle/oracle-database/index.html
- MySQL Tutorial. https://www.w3schools.com/mysql/
- GURU99. Tutorial de MySQL para principiantes Aprende en 7 días. https://guru99.es/sql/