GENERALITAT VALENCIANA
Conselleria d'Educació, Cultura i Esport

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

UD4. MODELO FÍSICO DDL

BASES DE DATOS (BD)
CFGS DAM/DAW

Pau Miñana
Abelardo Martínez
Basado y modificado de Sergio Badal y Raquel Torres

UD4. MODELO FÍSICO DDL

1. DDL
2. Instalación de MySQL
3. CREATE DATABASE
4. Tipos de datos
5. CREATE TABLE
6. Scripts
7. Restricciones de tabla
8. ALTER TABLE
9. Borrado de tablas
10. Bibliografía

1. DDL

SQL (*Structured Query Language*) es el lenguaje específico diseñado para administrar y recuperar información de los SGBDR (Sistemas Gestores de Bases de Datos Relacionales). Aunque la diversidad de añadidos particulares que incluyen las distintas implementaciones comerciales del lenguaje es amplia, el soporte al estándar SQL-92 es general y muy amplio.

DDL (*Data Definition Language*) es la parte de **SQL** que se encarga de la **creación, modificación y eliminación** de los **objetos de la BD**, entre los que se encuentran las tablas y los campos, principalmente. Es el lenguaje con el que accedemos a la base de datos que realiza la función de definición de datos de cualquier SGBD.

Sentencias SQL. Nomenclatura

Todas las **sentencias SQL** siempre terminan con **;** en caso contrario aunque pulses intro los SGBD simplemente muestran una nueva línea en pantalla para organizar la sentencia a nivel visual en varias líneas, pero no afecta a su funcionamiento.

Del mismo modo, aunque las sentencias SQL son indiferentes a las mayúsculas, se usan para distinguir fácilmente las palabras reservadas de los nombres usados en la BD (campos o tablas).

Cuidado, pues las sentencias DDL no pueden deshacerse y se puede perder toda la información al borrar tablas o bases de datos si no se tienen copias de seguridad.

Nombres. Nomenclatura

- Los nombres deben ser auto-descriptivos.
- Deben comenzar con una letra.
- No deben tener más de 30 caracteres.
- Sólo utilizar letras minúsculas del alfabeto (inglés), no se pueden usar espacios, usar un guion bajo para "separar" palabras.
- Dos tablas no se pueden llamar igual en el mismo esquema/BD.
- No puede haber dos campos con el mismo nombre en la misma tabla, en tablas distintas sí.
- No pueden coincidir con el nombre de una palabra reservada SQL (por ejemplo no se puede llamar SELECT a una tabla).

2. Instalación de MySQL

Para cualquier SO las instrucciones de instalación se pueden encontrar en la web de MySQL:

- https://dev.mysql.com/doc/mysql-installation-excerpt/8.0/en/

Para Ubuntu y otros sistemas similares que usen apt simplemente abre un terminal y escribe:

```
sudo apt update
sudo apt install mysql-server
sudo systemctl start mysql.service
sudo mysql
```

3. CREATE DATABASE

Nomenclatura

```
CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] nombre_bd
[especificación_create [especificación_create]…]

especificación_create:
{ [DEFAULT] CHARACTER SET juego_de_caracteres
| [DEFAULT] COLLATE nombre_de_colación }
```

- Cuando se puede elegir un elemento entre varios éstos van entre llaves **{ }** y separados por un pipe **|**.
- Cuando un elemento es opcional (puede incluirse u omitirse) se pone entre corchetes **[ ]**.

Detalles

- *IF NOT EXISTS* (opcional) antes *nombre_bd*. Solo se crea la BD si no existe previamente. Evita que salte un error y permite al proceso continuar si tenemos un script con varias sentencias, que se aplicarán probablemente sobre esa BD ya existente.
- Detrás del *nombre_bd* se pueden añadir especificaciones
  - *CHARACTER SET* establece el juego de caracteres a usar, se recomienda "*utf8mb4*".
  - *COLLATE* influye sobre como se tratan caracteres especiales. Crucial para comparar o ordenar alfabéticamente campos con acentos o la letra ñ. Se recomienda usar "*utf8mb4_spanish_ci*".

Ejemplo

```
CREATE DATABASE IF NOT EXISTS prueba1 CHARACTER SET utf8mb4 COLLATE utf8mb4_spanish_ci;
```

Si lo ejecutamos una segunda vez no hace nada, puesto que la BD "*prueba1*" ya existe, pero lanza un *WARNING*. Los errores se detallan inmediatamente por pantalla pero de los warnings solo se indica su número.

```
SHOW WARNINGS; -- Muestra los detalles de los warnings
```

```
SHOW DATABASES; -- Muestra una lista con todas las BD existentes
```

```
USE nombre_bd -- Cambia la base de datos que se está usando
```

```
DROP DATABASE [IF EXISTS] nombre_bd; -- Borra una BD (si existe); mucha precaución al usarlo
```

```
mysql>  CREATE DATABASE IF NOT EXISTS prueba1 CHARACTER SET utf8mb4 COLLATE utf8mb4_spanish_ci;
Query OK, 1 row affected (0.00 sec)

mysql>  CREATE DATABASE IF NOT EXISTS prueba1 CHARACTER SET utf8mb4 COLLATE utf8mb4_spanish_ci;
Query OK, 1 row affected, 1 warning (0.00 sec)

mysql> SHOW WARNINGS;
+-------+------+-----------------------------------------------------+
| Level | Code | Message                                             |
+-------+------+-----------------------------------------------------+
| Note  | 1007 | Can't create database 'prueba1'; database exists |
+-------+------+-----------------------------------------------------+
1 row in set (0.00 sec)

mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| prueba1            |
| sys                |
+--------------------+
5 rows in set (0.01 sec)

mysql> USE prueba1
Database changed
mysql> DROP DATABASE prueba1;
Query OK, 0 rows affected (0.01 sec)
```

4. Tipos de datos

NUMÉRICOS	DESCRIPCIÓN
INTEGER	Los valores mínimo y máximo dependen del SGBD, INT suele ser equivalente
SMALLINT	Puede contener un rango de valores más pequeño
BIGINT	Puede contener un rango de valores más grande, Oracle no lo usa
DECIMAL(p, s)	Número decimal, precisión p (número de dígitos), escala s (número de decimales)
FLOAT(p)	Numérico aproximado, p precisión de mantisa
REAL	Igual que FLOAT, pero el SGBD define la precisión
DOUBLE PRECISION	REAL con mayor capacidad
BOOLEAN	0/1, TRUE/FALSE, Oracle no lo implementa
NO ESTÁNDAR	
NUMBER(p,s)	Propio de Oracle, transforma los enteros y decimales estándar a este tipo
TINYINT	Usado por MySQL entre otros, rango menor a SMALLINT

TEXTUALES	DESCRIPCIÓN
CHAR(n)	Cadena de caracteres de longitud fija n,
VARCHAR(n)	Cadena de caracteres de longitud variable, máximo n

Atención a las **fechas**. Suelen ser compatibles entre SGBD, pero no comparten el formato por defecto para mostrarse/introducirse como texto.

FECHA/HORA	DESCRIPCIÓN
DATE	Representa una fecha
TIME	Representa un tiempo con horas, minutos, segundos (milisegundos opcionales)
DATETIME	Representa una fecha y un tiempo
TIMESTAMP	Suele tener menor rango que DATETIME y tener en cuenta la zona horaria

Campos de fecha

- **MySQL**: **'(AA)AA-MM-DD'** Año en 2/4 cifras, mes y dia 2 cifras cada uno.
- **Oracle**: **'DD-MES-AA(AA)'** Dia 2 cifras, mes en 3 letras, año 2/4 cifras

El separador en ambos casos también puede ser '/'

Usar formato propio de cada SGBD para introducir fechas o usar funciones.

- **MySQL**: *STR_TO_DATE(cadena, formato)*
  https://www.w3schools.com/sql/func_mysql_str_to_date.asp
- **Oracle**: *TO_DATE(cadena, formato)*
  https://www.databasestar.com/oracle-to_date/

5. CREATE TABLE

```
CREATE TABLE [IF NOT EXISTS] [nombre_bd.]nombre_tabla (
nombre_columna tipo_datos [definición_columna]
[,nombre_columna tipo_datos [definición_columna]]…
[,restricción_tabla] [,restricción_tabla]… )

definición_columna:
[PRIMARY KEY] [NOT NULL] [UNIQUE] [DEFAULT expr] [AUTO_INCREMENT]

restricción_tabla:
[CONSTRAINT nombre_restricción]
{ CHECK (condición)
| {UNIQUE | PRIMARY KEY} (nombre_columna [,nombre_columna]…)
| FOREIGN KEY (nombre_columna [,nombre_columna]…)
  REFERENCES nombre_tabla (nombre_columna [,nombre_columna]…)
  [ON DELETE | ON UPDATE {CASCADE | SET NULL | SET DEFAULT}]}
```

Definimos la tabla con su nombre y una lista entre paréntesis con sus campos + su tipo de datos y las restricciones separados por comas.

Recuerda que una tabla solo puede tener una *PK*, no puedes incluir esta definición en 2 columnas.

MySQL considera *NOT NULL* definición de columna, no restricción.

Algunos SGBD incluyen una restricción *NOT NULL* en *UNIQUE*, MySQL no.

Las **definiciones adicionales de columna** pueden expresar **CP/UK simples**, pero no compuestas. **No pueden expresar FK**.

Restricciones de tabla para **claves/restricciones simples o compuestas**.

Restricciones de tabla

- Incluir siempre un nombre con *CONSTRAINT*.
- Pueden expresarse sobre un campo o un grupo de campos. Los grupos van entre paréntesis separados por comas.
- *FOREIGN KEY* sólo se puede poner aquí (en MySQL).
- Las restricciones *FOREIGN KEY* tienen 3 partes:
  - *FOREIGN KEY*(campo o campos) indica sobre qué campos se aplica.
  - *REFERENCES* tabla(campo o campos) a qué tabla y campos apunta la *FK*.
  - *ON DELETE/UPDATE*... para restricciones de integridad de la *FK*.
  - *CHECK* obliga a que se cumpla una condición, puede contener funciones, operadores lógicos, literales y campos de la propia tabla.

Otras operaciones sobre tablas

Mostrar una lista con todas las tablas de una base de datos

```
SHOW tables;
```

Mostrar los campos con sus tipos y definiciones

```
DESC nombre_tabla;
```

Elimina por completo una tabla y su contenido, recuerda que no podrás recuperarla tras su eliminación sin una copia de seguridad.

```
DROP TABLE [IF EXISTS] nombre_tabla;
```

Ejemplo completo

Queremos crear las siguientes tablas de departamentos y empleados:

cod_dpt	nombre_dpt*	ubicacion

dni	nombre_emp*	especialidad	fecha_alta	dpt*

```
CREATE TABLE departamentos (
cod_dpt VARCHAR(4) PRIMARY KEY,
nombre_dpt VARCHAR(20) NOT NULL,
ubicacion VARCHAR(30) );

CREATE TABLE empleados (
dni VARCHAR(10) PRIMARY KEY,
nombre_emp VARCHAR(30) NOT NULL,
especialidad VARCHAR(20),
fecha_alta DATE,
dpt VARCHAR(4) NOT NULL,
CONSTRAINT emp_fk_dpt FOREIGN KEY(dpt) REFERENCES departamentos(cod_dpt) );
```

6. Scripts

Los scripts o guiones contienen órdenes que se quieren ejecutar. Pueden crearse en cualquier editor de texto y guardarse con extensión SQL.

Para ejecutar en MySQL

```
`SOURCE ruta\nombre_script`  --No es SQL, no acaba en ;
```

Por ejemplo si se tiene el script en C:\BD\scripts en Windows el comando sería *SOURCE C:\BD\scripts\prueba1.sql*.

En Oracle en vez de *SOURCE* se usa *@*.

Nuestro primer script basado en lo hecho hasta ahora, creamos la base de datos, marcamos que vamos a usarla y creamos las 2 tablas.

```
CREATE DATABASE IF NOT EXISTS prueba1
CHARACTER SET utf8mb4 COLLATE utf8mb4_spanish_ci;
use prueba1;
CREATE TABLE IF NOT EXISTS departamentos (
cod_dpt VARCHAR(4) PRIMARY KEY,
nombre_dpt VARCHAR(20) NOT NULL,
ubicacion VARCHAR(30) );
CREATE TABLE IF NOT EXISTS empleados (
dni VARCHAR(10) PRIMARY KEY,
nombre_emp VARCHAR(30) NOT NULL,
especialidad VARCHAR(20),
fecha_alta DATE,
dpt VARCHAR(4) NOT NULL,
CONSTRAINT emp_dpt_fk FOREIGN KEY(dpt) REFERENCES departamentos(cod_dpt) );
```

7. Restricciones de tabla

Nombre de restricción

- *Tres letras* para el nombre de la *tabla* y separamos con guion bajo.
- *Tres letras por columna* afectada por la restricción y separamos con guion bajo.
- *Dos letras* con la abreviatura del tipo de restricción. La abreviatura puede ser:
  - *nn. NOT NULL* (En MySQL no se considera restricción, no se usa)
  - *pk. PRIMARY KEY*
  - *uk. UNIQUE*
  - *fk. FOREIGN KEY*
  - *ck. CHECK* (validación)

Ejemplo: *emp_dpt_fk* indica que dpt en tabla empleados tiene una restricción *FK*.

UNIQUE

```
CREATE TABLE cliente(
dni VARCHAR(9) UNIQUE
);
ó
CREATE TABLE cliente(
dni VARCHAR(9),
CONSTRAINT cli_dni_uk UNIQUE(dni)
);

CREATE TABLE cita (
dni VARCHAR(9),
fecha DATE,
CONSTRAINT cit_dnifec_uk UNIQUE(dni, fecha)
);
```

PRIMARY KEY

```
CREATE TABLE cliente(
dni VARCHAR(9) PRIMARY KEY,
nombre VARCHAR(50) );
ó
CREATE TABLE cliente(
dni VARCHAR(9),
nombre VARCHAR(50),
CONSTRAINT cli_dni_pk PRIMARY KEY(dni) );

CREATE TABLE cita (
dni VARCHAR(9),
fecha DATE,
lugar VARCHAR(50),
CONSTRAINT cit_dnifec_pk PRIMARY KEY(dni, fecha) );
```

FOREIGN KEY

```
CREATE TABLE cliente(
dni VARCHAR(9) PRIMARY KEY,
nombre VARCHAR(50) );

CREATE TABLE cita (
dni VARCHAR(9),
fecha DATE,
lugar VARCHAR(50),
CONSTRAINT cit_dnifec_pk PRIMARY KEY(dni, fecha),
CONSTRAINT cit_dni_fk FOREIGN KEY(dni) REFERENCES cliente(dni) );
```

Clave compuesta, supongamos una tabla con una *FK* referenciando a *cita*:

```
CONSTRAINT X_fk FOREIGN KEY(dni,fecha) REFERENCES cita(dni,fecha)
```

FOREIGN KEY CON ON DELETE/UPDATE

ON DELETE/UPDATE: Se puede poner ninguna una o las dos

```
CONSTRAINT cit_dni_fk FOREIGN KEY(dni) REFERENCES cliente(dni) ON DELETE SET NULL ON UPDATE CASCADE
```

- *ON {DELETE | UPDATE} SET NULL*. Coloca nulos en los registros donde aparece en la clave ajena el valor borrado/modificado en la otra tabla.
- *ON {DELETE | UPDATE} CASCADE*. Borra todos los registros donde aparece en la clave ajena el valor borrado/modificado en la otra tabla.
- *ON {DELETE | UPDATE} SET DEFAULT*. Coloca el valor por defecto en los registros donde aparece en la clave ajena el valor borrado/modificado en la otra tabla. Debe existir un DEFAULT en la creación de la tabla para esos campos.
- *ON {DELETE | UPDATE} {NO ACTION | RESTRICT}*. Salta un error que impide borrar/modificar el registro donde estaba la clave principal. Valor por defecto, si no se pone nada actúa así.

CHECK

Puede condicionar más de un campo de la tabla pero nunca a otros campos fuera de la misma. Además se pueden añadir varias restricciones a un mismo campo.

La condición puede ser cualquier expresión válida que de como resultado *TRUE/FALSE*. Puede contener funciones, campos, literales o operadores lógicos (*AND*, *OR* y *NOT*). Atención al comparador *IN*, permite comprobar si un valor está contenido en un grupo de opciones definidas dentro de un paréntesis y separadas por comas.

```
CREATE TABLE cita (
dni VARCHAR(9),
fecha DATE,
lugar VARCHAR(50),
precio DECIMAL(6,2),
CONSTRAINT cit_prec_ck CHECK(precio>0 AND precio<1000),
CONSTRAINT cit_lug_ck CHECK(lugar IN ('A1','A2','B1','B2')) );
```

8. ALTER TABLE

```
ALTER TABLE nombre_tabla opción_alter [, opción_alter]…

opción_alter:
{ RENAME nombre_tabla_nuevo
| RENAME COLUMN nombre_columna TO nuevo_nombre
| ADD restricción_tabla
| ADD nombre_columna tipo_datos [definición_columna]
  [{FIRST | AFTER nombre_columna}]
| DROP {nombre_columna | CONSTRAINT nombre_restricción}
| DROP PRIMARY KEY
| MODIFY nombre_columna tipo_datos [definición_columna]
  [{FIRST | AFTER nombre_columna}] }
```

Vamos a ver cada caso, deshaciendo los cambios cada vez para volver al estado original.

Cambiar de nombre una tabla

RENAME tras el nombre de la tabla para cambiar el nombre de la misma.

```
ALTER TABLE nombre_tabla RENAME nombre_nuevo

ALTER TABLE empleados RENAME empleado;
ALTER TABLE empleado RENAME empleados;
```

Cambiar de nombre una columna

Añadir *COLUMN* a *RENAME* y poner un *TO* entre el nombre viejo y el nuevo

```
ALTER TABLE nombre_tabla RENAME COLUMN nombre_columna TO nuevo_nombre;

ALTER TABLE empleados RENAME COLUMN dni TO nif;
ALTER TABLE empleados RENAME COLUMN nif TO dni;
```

Añadir o quitar restricciones

ADD CONSTRAINT seguido de la definición de la restricción igual que cuando se crea con *CREATE TABLE*.

Eliminar restricciones con *DROP CONSTRAINT* seguido de su nombre.

```
ALTER TABLE nombre_tabla
ADD CONSTRAINT restricción_tabla

ALTER TABLE nombre_tabla DROP CONSTRAINT nombre_restricción

ALTER TABLE empleados ADD CONSTRAINT emp_dni_uk UNIQUE(dni);
ALTER TABLE empleados DROP CONSTRAINT emp_dni_uk;
```

Añadir *UNIQUE* a una *PRIMARY KEY* es redundante, aunque lo permita.

Añadir o quitar columnas

Mismo procedimiento pero cambiando la palabra *CONSTRAINT* por *COLUMN* (opcional). MySQL tiene la opción adicional propia de añadir esta columna en un lugar determinado, o la primera (*FIRST*) o detrás de otra columna existente (*AFTER nombre_columna*).

```
ALTER TABLE nombre_tabla
ADD  definición_columna [{FIRST | AFTER nombre_columna}]

ALTER TABLE nombre_tabla DROP nombre_columna

ALTER TABLE empleados ADD  salario DOUBLE AFTER nombre_emp;
ALTER TABLE empleados DROP salario;
```

Eliminar PRIMARY KEY

Existe una opción específica *DROP PRIMARY KEY* para eliminar las claves primarias, tanto si están definidas como restricción de tabla como en la propia columna.

```
ALTER TABLE nombre_tabla DROP PRIMARY KEY
```

Esto **no elimina los campos**, solo su condición de clave primaria.

Modificar columnas

MODIFY seguido de un *nombre_columna* existente, su *tipo de datos* y sus *definicion_columna*. No solo las nuevas, todas las que se quieran tener. Funciona como un DROP+ADD más que como una modificación. Aún así, **en MySQL no anula una definición de PRIMARY KEY**. En MySQL también admite FIRST/AFTER.

```
ALTER TABLE nombre_tabla
MODIFY definición_columna [{FIRST | AFTER nombre_columna}]

ALTER TABLE empleados MODIFY dni VARCHAR(9) AFTER nombre_emp;
ALTER TABLE empleados MODIFY dni VARCHAR(10) FIRST;
```

Hay que tener en cuenta que si las tablas tienen datos, hay muchos cambios que no pueden realizarse si entran en contradicción con los datos existentes, como cambiar el tipo de valores o disminuir el tamaño. Las ampliación suele funcionar.

Modificar restricciones

Aunque otros SGBD sí que incorporan una opción para *MODIFY CONSTRAINT*, MySQL no la contempla.

Se puede, no obstante, borrar la restricción (*DROP CONSTRAINT*) y añadirla(*ADD CONSTRAINT*) con la nueva definición con las opciones anteriormente vistas.

```
SHOW CREATE TABLE nombre_tabla
```

Muestra la orden completa para recrear una tabla en su estado actual. Permite ver detalles y nombres de todos los campos y restricciones. **Úsalo si necesitas conocer el nombre de una restricción**.

Desactivar restricciones

En ocasiones es conveniente desactivar alguna restricción para poder rellenar los datos de una tabla y posteriormente reactivarla. Sobre todo cuando 2 tablas tienen claves ajenas cruzadas.

MySQL controla **todas las FOREIGN KEYS** mediante una variable interna, *FOREIGN_KEY_CHECKS*. 1 restricciones activadas, 0 desactivadas.

Al reactivarlas no las comprueba de nuevo, es una operación con mucho riesgo

```
-- Desactivar comprobación de claves ajenas
SET FOREIGN_KEY_CHECKS = 0;
-- Ahora se pueden romper las claves ajenas hasta que no recuperemos el valor 1.
SET FOREIGN_KEY_CHECKS = 1;
```

9. Borrado de tablas

```
DROP TABLE [IF EXISTS] nombre_tabla;
```

- Desaparecen todos los datos junto con la tabla.
- Sólo se pueden eliminar las tablas sobre las que tenemos permiso de borrado.
- En MySQL no se pueden borrar tablas que estén referenciadas en claves foráneas para no romper la integridad de los datos. Oracle permite añadir *CASCADE CONSTRAINTS* para que las restricciones que referencien esta tabla se borren y pueda ser eliminada.
- No existe ninguna petición de confirmación y el cambio es irreversible, excepto que se dispongan de copias de seguridad de la base de datos entera.

Borrar solo el contenido de una tabla

Oracle y MySQL disponen de una orden no estándar para eliminar definitivamente todos los datos contenidos dentro de una tabla, pero no la estructura de la tabla en sí.

Incluso se borra del archivo de datos el espacio ocupado por la tabla.

```
TRUNCATE TABLE nombre_tabla
```

10. Bibliografía

- 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
- JavaTPoint. Difference between MySQL and Oracle. https://www.javatpoint.com/mysql-vs-oracle
- W3Schools. MySQL Tutorial. https://www.w3schools.com/mysql/
- GURU99. Tutorial de MySQL para principiantes Aprende en 7 días. https://guru99.es/sql/
- Adam McGurk.How to change a foreign key constraint in MySQL https://dev.to/mcgurkadam/how-to-change-a-foreign-key-constraint-in-mysql-1cma
- Sqlines. MySQL - SET FOREIGN_KEY_CHECKS. http://www.sqlines.com/mysql/set_foreign_key_checks