Unidad de Trabajo 5
Tratamiento de los datos

Creación y edición de tablas

IES Palomeras Vallecas

Curso 2020/2021

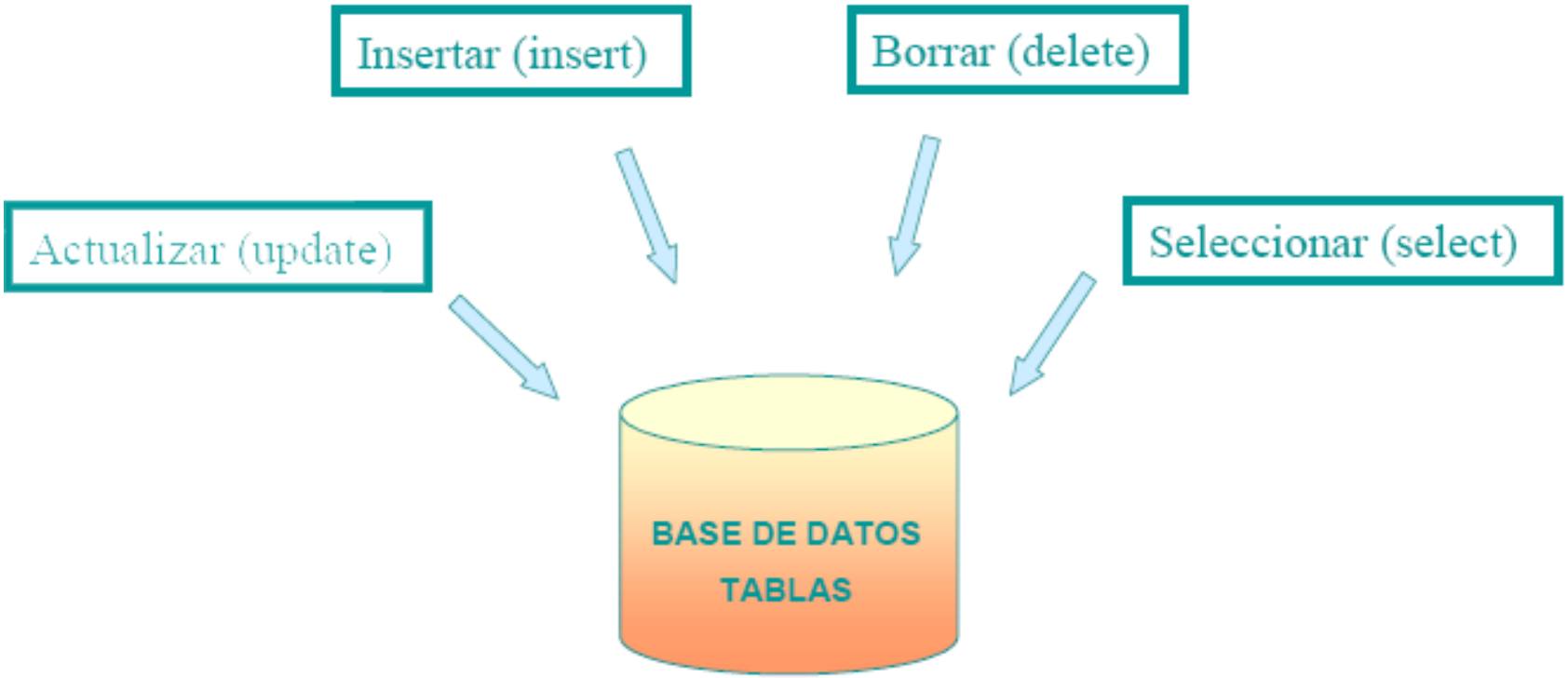
Profesor: Alberto Ruiz

Las tres caras de SQL

- **DML** (Lenguaje de Manipulación de Datos)
  - Realizar consultas, insertar, borrar o modificar datos
- **DDL** (Lenguaje de Definición de Datos)
  - Definición de dominios, tablas, vistas y restricciones de integridad
- **DCL** (Lenguaje de Control de Datos)
  - Gestionar privilegios, permisos, etc

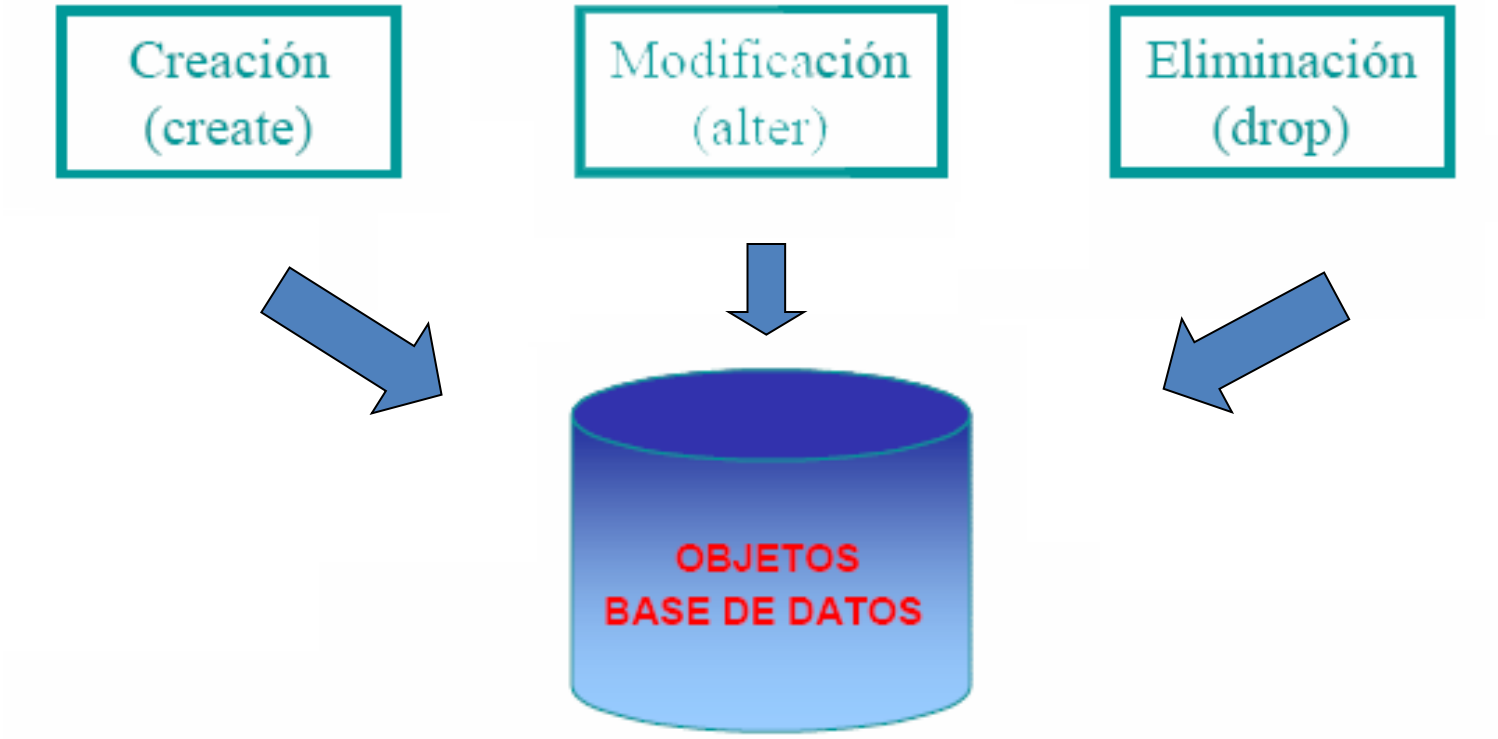
SQL como DML

- Comandos DML:



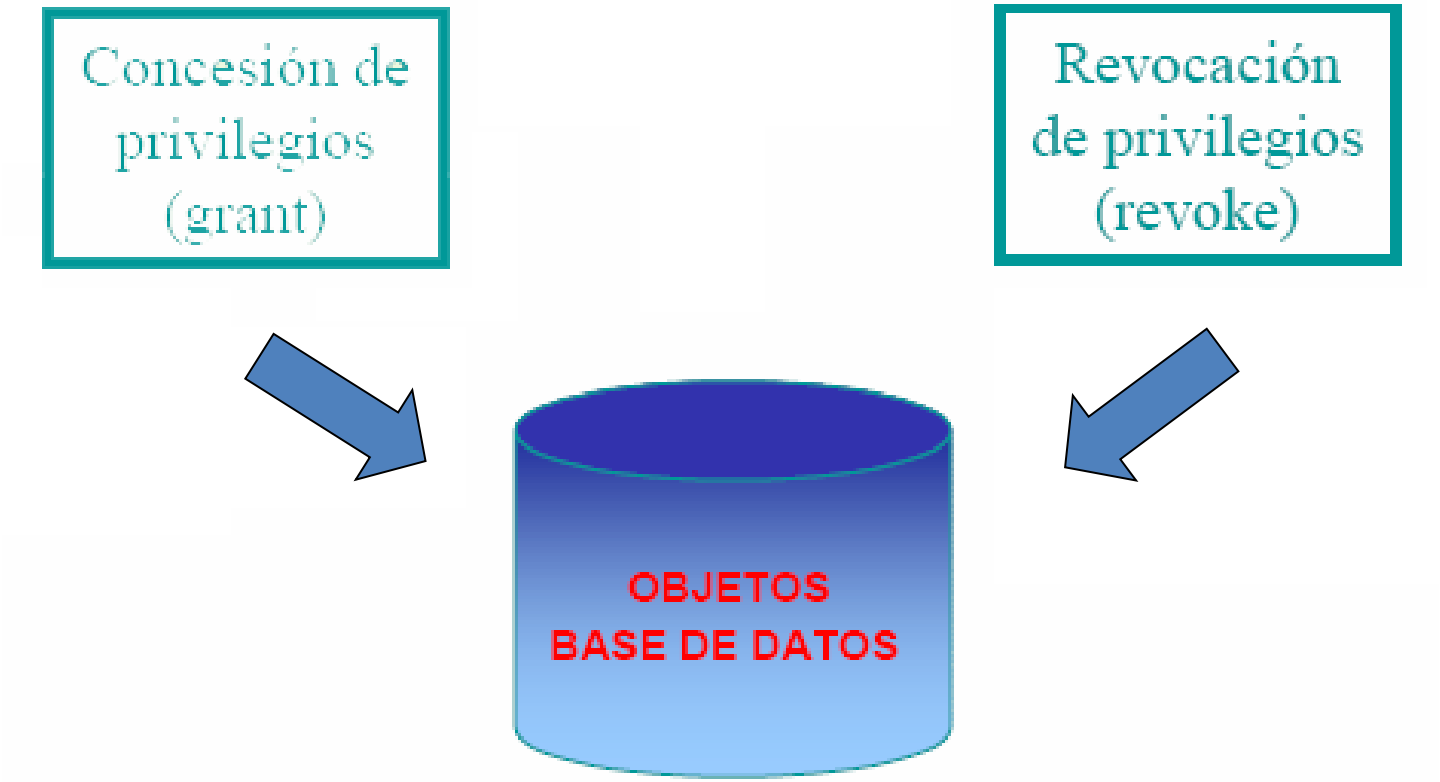
SQL como DDL

- Comandos DDL:



SQL como DCL

- Comandos DCL:



Creación de una base de datos

- La creación y eliminación de una base de datos se hace en MySQL con CREATE SCHEMA o CREATE DATABASE (da igual cuál usar), y con DROP

```
CREATE DATABASE Instituto;
```

- Si miras los scripts, es habitual poner condiciones para evitar fallos.

```
CREATE DATABASE IF NOT EXISTS Instituto;
DROP DATABASE IF EXISTS Instituto;
```

Creación de una tabla

- Empezamos con un ejemplo sencillo sin indicar claves primarias, ajenas, comprobaciones…
- Por cada campo indicamos su tipo de datos y sus atributos

```
CREATE TABLE Alumnos
(
   NumeroMatricula INTEGER UNIQUE,
   Nombre VARCHAR(15) NOT NULL UNIQUE,
   FechaNacimiento DATE
);
```

Tipos de datos

Texto:

- VARCHAR(n): Pueden contener cualquier carácter.Longitud variable con un tamaño máximo de n bytes, siendo el límite para n de 65536. La ventaja de este tipo es que sólo consume en memoria el número de caracteres que realmente se use.
- CHAR(n): Pueden contener cualquier carácter

Tipos de datos

Fechas y horas:

- DATE: una fecha con formato 2020-12-31
- TIME: una hora con formato 21:50:05
- DATETIME: combinación de ambas con un espacio en medio: 2020-12-31 21:50:05
- YEAR: un año, como 2021

Tipos de datos

Números enteros:

- TINYINT: un número entero pequeño de un byte (8 bits), por tanto será entre -128 y 127
- SMALLINT: dos bytes (aproximadamente entre -32.000 y 32.000)
- MEDIUMINT: tres bytes (aproximadamente entre -8.000.000 y 8.000.000
- INT: cuatro bytes (aproximadamente entre -2.000.000.000 y 2.000.000.000
- BIGINT: ocho bytes (entre -2^{63} y 2^{63})

Tipos de datos

Números enteros:

- En todos estos casos, podemos marcar los campos como UNSIGNED (casilla UN en MySQL Workbench). Al hacer esto sólo se consideran números positivos y por tanto se duplican los posibles valores positivos que podemos almacenar:
  - por ejemplo un valor de tipo TINYINT sin signo podrá valer entre 0 y 255.

Tipos de datos

Números con decimales:

- FLOAT: número decimal de 4 bytes
- DOUBLE: número decimal de 8 bytes
- DECIMAL(5,2): este tipo sirve cuando queremos un número exacto de cifras decimales. Por ejemplo estos valores indican que se almacenan números de 5 dígitos y 2 de ellos son decimales, admitiendo únicamente valores del estilo 567,43. Este tipo se suele usar para valores monetarios.

  Ejemplo: DECIMAL (6,2) admitiría de -9999.99 a 9999.99

Tipos de datos

Números:

- Podemos marcar los campos como ZEROFILL (casilla ZF en MySQL Workbench).
- Se rellenan con ceros los caracteres no usados
- Ejemplo: en DECIMAL(5,1) 6,5 pasa a ser 0006,5
- *Aunque aparece en MySQL Workbench, es un atributo considerado obsoleto que terminará desapareciendo de MySQL*

Tipos de datos

Booleanos:

- Este tipo no existe como tal en MySQL: en su lugar se almacena como TINYINT, con valores de 1 (true) o 0 (false).
- Cuando rellenes valores en la tabla de un campo de este estilo, no podrás usar las palabras **true** o **false**: sin embargo estas palabras sí se podrán usar al utilizar sentencias SQL, y MySQL las traducirá automáticamente a 1 y 0.

Tipos de datos de texto

Booleanos:

Ejemplo de consulta:

```
SELECT *
FROM receta
WHERE aptoCelíacos = TRUE
```

Al igual que NULL, TRUE y FALSE se escriben <u>sin comillas</u>

Tipos de datos de texto

Archivos binarios:

BLOB = Binary Large Object

- Se trata de contenido que no es texto
- Podría ser una imagen, un documento PDF…)
- Se marca la casilla B (Binary) en MySQL Workbench
- No lo usaremos en este curso

Tipos de datos de texto

Enumerados:

```
color ENUM ('verde', 'rojo', 'amarillo', 'azul'),
género ENUM ('hombre', 'mujer', 'otro')
```

Admite únicamente un valor de los indicados.

Es muy útil para garantizar que el usuario rellena los datos correctamente

Tipos de datos de texto

Conjuntos:

```
idiomas SET ('francés', 'alemán', 'inglés')
```

Admite un conjunto (puede ser vacío) de valores separados por comas. Ejemplos válidos (**no pongas espacios entre los valores**):

'alemán'

'francés,alemán'

''

'francés,inglés,alemán'

Atributos de campo

- Después de cada campo puedes poner atributos que se refieran únicamente a ese campo:

```
CREATE TABLE Alumnos
(
   NumeroMatricula INT UNIQUE,
   Nombre VARCHAR(15) NOT NULL UNIQUE,
   FechaNacimiento DATE
);
```

Atributos de campo

- UNIQUE:
  - Restricción que impide duplicados (permite NULL)
- NOT NULL:
  - no se permite dejarlo vacío
- DEFAULT valor:
  - si no rellenamos el campo, tomará el valor 'valor' por defecto
- AUTO_INCREMENT:
  - si no rellenamos el valor, pondrá automáticamente el siguiente número libre. Muy útil para campos 'id' de clave primaria. Sólo se permite un campo así por tabla

Atributos de campo

```
CREATE TABLE Alumno
(
    id INT AUTO_INCREMENT,
    nombre VARCHAR(90) NOT NULL,
    edad TINYINT,
    ciudad VARCHAR(50) DEFAULT 'Madrid',
    idProfesor INT NOT NULL UNIQUE
);
```

Campos generados o calculados

```
Precio INT,
PrecioConIVA INT AS (Precio*1.21)
```

Son campos que no rellenamos nosotros, sino que se encarga de hacerlo MySQL a partir de una fórmula que depende de otros campos

Restricciones de campo

- La restricción de clave primaria se puede incluir junto al campo…

```
CREATE TABLE Alumno
(
   id INT AUTO_INCREMENT PRIMARY KEY,
   nombre VARCHAR(90) NOT NULL,
   edad TINYINT,
   ciudad VARCHAR(50) DEFAULT 'Madrid',
   idProfesor INT NOT NULL UNIQUE
);
```

Restricciones de tabla

- … Pero se suele incluir al final, una vez descritos todos los campos, en la sección de restricciones de tabla.

```
CREATE TABLE Alumno
(
    id INT AUTO_INCREMENT,
    nombre VARCHAR(90) NOT NULL,
    edad TINYINT,
    ciudad VARCHAR(50) DEFAULT 'Madrid',
    idProfesor INT NOT NULL UNIQUE,
    PRIMARY KEY (id)
);
```

Restricciones de campo VS tabla

- ¿Cuál es la diferencia?
  - En el ejemplo que hemos visto, ninguna…
  - Pero imagina que la clave primaria está compuesta por dos campos: en este caso es necesario expresarla como restricción de tabla.
  - Ejemplo: los campos de una tabla que proviene de una relación N:N (“pertenece” en Ciclos por ejemplo) son los dos clave primaria

```
CREATE TABLE Alumno
(
   nombre VARCHAR(90) NOT NULL,
   apellidos VARCHAR(90) NOT NULL,
   edad TINYINT,
   ciudad VARCHAR(50) DEFAULT 'Madrid',
   idProfesor INT NOT NULL UNIQUE,
   PRIMARY KEY (nombre, apellidos)
);
```

Restricciones de campo VS tabla

- Pasa lo mismo con UNIQUE
  - Imagina que quieres permitir que se repita un nombre, y también un apellido, pero no quieres que se repita la misma pareja nombre-apellido
  - No podrías poner UNIQUE en nombre ni tampoco en apellido: lo tendrías que poner como restricción de tabla

```
CREATE TABLE Alumno
(
   id INT AUTO_INCREMENT,
   nombre VARCHAR(90) NOT NULL,
   apellidos VARCHAR(90) NOT NULL,
   edad TINYINT,
   ciudad VARCHAR(50) DEFAULT 'Madrid',
   idProfesor INT NOT NULL UNIQUE,
   PRIMARY KEY (id),
   UNIQUE (nombre, apellidos)
);
```

Restricciones de comprobación

- CHECK permite asegurar ciertas normas para los valores admitidos en nuestros campos
- Las comprobaciones se limitan a campos de esta tabla: para comprobaciones multitabla es necesario ASSERTION, que no estudiaremos

```
CREATE TABLE Alumno
(
   id INT AUTO_INCREMENT,
   nombre VARCHAR(90) NOT NULL,
   apellido1 VARCHAR(30) NOT NULL,
   apellido2 VARCHAR(30) NOT NULL,
   edad TINYINT,
   códigoMatrícula VARCHAR(5),
   PRIMARY KEY (id),
   CHECK (edad>0 AND edad<120),
   CHECK (códigoMatrícula NOT LIKE "Z%"),
   CHECK (apellido1 != apellido2)
);
```

Restricción de clave ajena

- Es la restricción fundamental para garantizar la coherencia de los datos de las distintas tablas

```
CREATE TABLE Alumno
(
   id INT AUTO_INCREMENT,
   nombre VARCHAR(90) NOT NULL,
   edad TINYINT,
   ciudad VARCHAR(50) DEFAULT 'Madrid',
   idProfesor INT NOT NULL UNIQUE,
   PRIMARY KEY (id),
   FOREIGN KEY (idProfesor) REFERENCES Profesor(id)
);
```

Restricción de clave ajena

Indica que el campo es clave ajena, es decir, su valor se debe corresponder con alguno de los valores del campo original (id) en la tabla padre (Profesor)

FOREIGN KEY (idProfesor) REFERENCES Profesor(id);

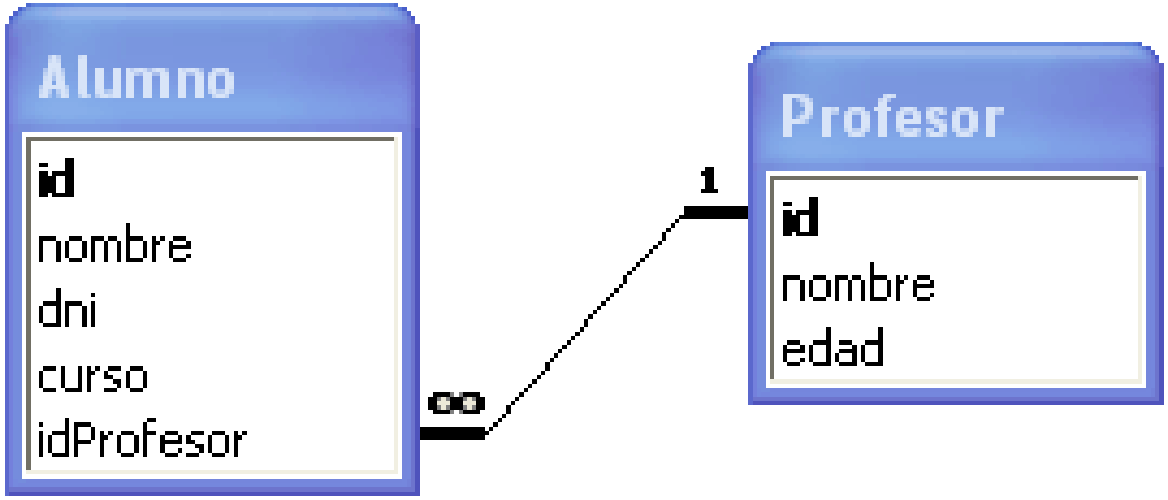
Restricciones de tabla

- FOREIGN KEY (idProfesor)
  REFERENCES Profesor(id) REGLAS

Las reglas indican el comportamiento del SGBD al ocurrir cosas que pongan en juego la integridad referencial:

- ON DELETE : qué hacer si se borra una clave primaria en la tabla Profesor
- ON UPDATE: qué hacer si se modifica una clave primaria en la tabla Profesor

Ejemplo



- Tabla Alumno: el campo “idProfesor” es clave ajena, es decir, es clave primaria en la tabla Profesor
- Se dice que Profesor es tabla padre de Alumno, ya que Alumno depende de ella

Recuerda el problema

- Si modifico o elimino “id”, la clave primaria de un profesor (o directamente elimino al profesor), ¿qué ocurre con los alumnos que tenían ese “id” como clave ajena?

Alumno : Tabla

id	nombre	dni	curso	idProfesor
1	Andrés	123434	4	1
2	Pablo	2342	3	1
3	Laura	34235	3	1
4	Ana	5452434	2	3
5	Luisa	342523	3	2
6	Andrea	4543523	2	3
7	Julio	2342243	4	2

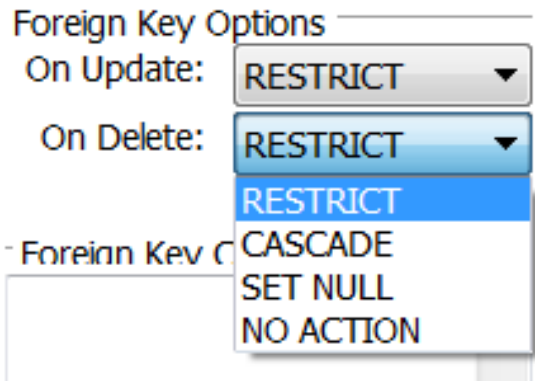
Profesor : Tabla

id	nombre	edad
1	Juan	50
2	Pedro	45
3	Sergio	30

ON DELETE / ON UPDATE

3 acciones posibles para ON DELETE y para ON UPDATE:

- RESTRICT o NO ACTION
- SET NULL
- CASCADE



ON DELETE / ON UPDATE

- RESTRICT o NO ACTION: no se permite [borrar / modificar] la clave primaria si hay claves ajenas referenciándola

- No permitimos tocar el “id” del profesor si existen alumnos que lo referencien en su campo “idProfesor”

ON DELETE / ON UPDATE

- SET NULL: se permite [borrar / modificar] la clave primaria y automáticamente se asigna valor NULL a la/s columna/s que forman parte de la clave ajena. Esto NO funcionará si estas columnas están marcadas como NOT NULL

Profesor : Tabla

id	nombre	edad
40	Juan	50
2	Pedro	45
3	Sergio	30

Alumno : Tabla

id	nombre	dni	curso	idProfesor
1	Andrés	123434	4	
2	Pablo	2342	3	
3	Laura	34235	3	
4	Ana	5452434	2	3
5	Luisa	342523	3	2
6	Andrea	4543523	2	3
7	Julio	2342243	4	2

ON DELETE / ON UPDATE

- CASCADE: se permite [borrar / modificar] la clave primaria y automáticamente se

  [borra/ modifica] la clave ajena que la referencia

Profesor : Tabla

id	nombre	edad
40	Juan	50
2	Pedro	45
3	Sergio	30

Alumno : Tabla

id	nombre	dni	curso	idProfesor
1	Andrés	123434	4	40
2	Pablo	2342	3	40
3	Laura	34235	3	40
4	Ana	5452434	2	3
5	Luisa	342523	3	2
6	Andrea	4543523	2	3
7	Julio	2342243	4	2

ON DELETE / ON UPDATE

- Las dos cláusulas son opcionales, puede ponerse una regla de borrado, una regla de actualización, ninguna regla o las dos reglas.
- Si no se pone nada, por defecto se entenderá NO ACTION / RESTRICT

```
FOREIGN KEY (idProfesor)
     REFERENCES Profesor(id)
     ON DELETE CASCADE
     ON UPDATE SET NULL
```

Nombrar restricciones

```
PRIMARY KEY (id)
UNIQUE (nombre, apellidos)
CHECK (edad>0 AND edad<120)
FOREIGN KEY (idProfesor) REFERENCES Profesor(id) ON
   DELETE CASCADE ON UPDATE SET NULL
```

Si más adelante queremos quitar una restricción (CONSTRAINT en inglés), ¿cómo lo hacemos sin tener que rehacer la tabla entera? La solución es darles nombre:

```
CONSTRAINT pk PRIMARY KEY (id)
CONSTRAINT nombreÚnico UNIQUE (nombre, apellidos)
CONSTRAINT chk_edad CHECK (edad>0 AND edad<120)
CONSTRAINT fk_profe FOREIGN KEY (idProfesor) REFERENCES
   Profesor(id) ON DELETE CASCADE ON UPDATE SET NULL
```

Borrado de tablas

- Para eliminar definitivamente una tabla o tablas:

```
DROP tabla;
DROP tabla1, tabla2, tabla3;
```

- No podemos borrar una tabla si tiene tablas hijas, es decir, si hay claves ajenas referenciándola

Modificación de tablas

ALTER TABLE nombreTabla MODIFICACIÓN;

Posibles modificaciones:

- Añadir nuevo campo (ADD COLUMN)
- Eliminar campo existente (DROP COLUMN)
- Cambiar tipo de un campo (MODIFY COLUMN)
- Cambiar el nombre de la tabla (RENAME)
- Añadir restricción (ADD CONSTRAINT)
- Eliminar restricción (DROP)

Añadir campo

```
ALTER TABLE Alumno ADD COLUMN edad INT;
```

- Si añades más de un campo utiliza paréntesis:

```
ALTER TABLE Alumno ADD COLUMN (
    becado CHAR(1) NOT NULL,
    lugarNacimiento VARCHAR(50));
```

- Si queremos que se añada en una posición determinada:

```
ALTER TABLE Profesor ADD COLUMN segSocial INT
  AFTER Nombre;
```

Eliminar campo

```
DROP COLUMN nombreCampo;
```

```
ALTER TABLE Profesor DROP COLUMN segSocial;
```

```
ALTER TABLE Alumno DROP COLUMN dni;
```

Cambiar tipo de un campo

```
MODIFY COLUMN nombreCampo TIPO;
```

```
ALTER TABLE Profesor MODIFY COLUMN
  segSocial VARCHAR(200);

ALTER TABLE Alumno MODIFY COLUMN sexo
  VARCHAR(5);
```

Cambiar nombre de tabla

```
RENAME nuevoNombreTabla;
```

```
ALTER TABLE Profesor RENAME Docente;
```

Añadir restricción

```
ADD CONSTRAINT nombreRestricción RESTRICCIÓN
```

```
ALTER TABLE Profesor ADD CONSTRAINT pk PRIMARY
  KEY (id);
```

```
ALTER TABLE Alumno ADD CONSTRAINT forKeyProfe
  FOREIGN KEY (idProfesor) REFERENCES
  Profesor(id);
```

```
ALTER TABLE Alumno ADD CONSTRAINT chk_edad
  CHECK (edad>9)
```

Eliminar restricción

```
ALTER TABLE Profesor DROP PRIMARY KEY;

ALTER TABLE Alumno DROP FOREIGN KEY forKeyProfe;

ALTER TABLE Alumno DROP INDEX nombreÚnico;
```

Las restricciones UNIQUE generan un índice o INDEX (una especie de resumen ordenado de los valores para comprobar si hay valores repetidos). Para borrarlas hay que poner DROP INDEX, y para encontrarlas en MySQL Workbench hay que buscarlas en la sección Indexes.

```
ALTER TABLE Alumno DROP CHECK chk_edad;
```

Nota: esto último te puede dar un aviso de error en MySQL Workbench porque las cláusulas CHECK sólo están disponibles desde la versión 8.0.16: sin embargo si lo ejecutas funcionará

Un caso especial

```
SELECT Nombre, Apellidos
INTO ClientesFavoritos
FROM Cliente
WHERE Saldo > 10000 ;
```

- Se crea una nueva tabla de nombre 'ClientesFavoritos' con los campos Nombre y Apellidos y se rellena con los registros que cumplan la condición
- Estamos usando SQL como DDL en un SELECT