**Diseño Físico**

### Lenguaje de definición de datos
### Data Definition Language (DDL)

Permite crear y modificar la estructura de las tablas de la base de datos.

```
CREATE TABLE pet (
     name VARCHAR(20),
     owner VARCHAR(20),
     birth DATE
);
```

### Lenguaje de manipulación de base de datos
### Data Manipulation Language (DML)

Permite insertar, modificar y eliminar datos de la base de datos.

```
INSERT INTO Customers (CustomerName, ContactName)
VALUES ('Cardinal', 'Tom B. Erichsen);

UPDATE Customers
SET ContactName = 'Alfred Schmidt', City= 'Frankfurt'
WHERE CustomerID = 1;
```

### Lenguaje de consulta de datos
### Data Query Language (DQL)

Permite realizar consultas sobre los datos de la base de datos.

```
SELECT CustomerName, City FROM Customers;

SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country
ORDER BY COUNT(CustomerID) DESC;
```

## Lenguaje de control de datos
## Data Control Language (DCL)

Permite gestionar permisos de usuario y trabajar con transacciones.

```
GRANT CREATE TABLE TO juanito;
```

## Comandos SQL.

| Comando | Descripción |
|---|---|
| `show databases;` | Lista el nombre de las bases de datos disponibles. |
| `use db_name;` | Selecciona la base de datos *db_name*. |
| `show tables;` | Muestra las tablas de la base de datos seleccionada. |
| `describe table_name;` | Muestra el formato de los campos de la tabla *table_name*. |
| `show columns from table_name;` | Muestra las columnas, su formato y restricciones de la tabla *table_name*. |

## Tipos de datos

| Tipo | Descripción |
|---|---|
| `INTEGER` | Permite almacenar valores enteros entre -2147483648 y 2147483647. |
| `DECIMAL(p, s)` | Número decimal exacto donde *p* es el número de dígitos del número y *s* cuántos de esos dígitos forman parte de la parte decimal. |
| `DOUBLE` | Número decimal aproximado |
| `CHAR(n)` | Cadena de caracteres de longitud fija *n*. |
| `VARCHAR(n)` | Cadena de caracteres de longitud variable *n*. |

## Tipos de datos

| Tipo | Descripción |
|---|---|
| `DATE` | Utilizado para almacenar fechas en formato AAAA-MM-DD. |
| `TIME` | Contiene una hora en formato HH:MM:SS. |
| `DATETIME` | Almacena una fecha y hora en formato AAAA-MM-DD HH:MM:SS |
| `TIMESTAMP` | Incluye tanto la fecha como la hora y añade la zona horaria.<br>Por ejemplo: 2017-05-32 20:40:58 UTC |

## Valores por defecto

```
CREATE TABLE personaje (
    nombre VARCHAR(25),
    nivel INT DEFAULT 1
);
```

## Borrar tablas

```
DROP TABLE IF EXISTS NombreTabla;
```

Ejemplo:

```
DROP TABLE IF EXISTS proveedor;
```

### Restricciones

Una restricción es una condición de obligado cumplimiento para una o más columnas de la tabla.

Las restricciones que podemos aplicar son:

- Restricción de clave primaria: PRIMARY KEY.
- Restricción de valor no nulo: NOT NULL.
- Restricción de unicidad: UNIQUE.
- Restricción de clave ajena: FOREIGN KEY.
- Restricción de validación: CHECK.

### Restricciones de columna y tabla

Algunas restricciones se pueden definir a nivel de columna y/o tabla.

- Las restricciones de tabla pueden hacer referencia a múltiples columnas mientras que las restricciones de columna no.
- Si se trata de una restricción de tabla opcionalmente podemos asignar un nombre a la restricción.
- No todas las restricciones se pueden definir a nivel de columna o tabla.

### PRIMARY KEY

Autor (<u>código</u>, nombre)

```
CREATE TABLE autor (
     codigo INTEGER PRIMARY KEY,
     nombre VARCHAR(50)
);
```

## NOT NULL

Alumno (dni, nombre, nota, fecha_nacimiento)
NN

```
CREATE TABLE Alumno (
     dni CHAR(9) PRIMARY KEY,
     nombre VARCHAR(50),
     nota DECIMAL(3,1),
     fecha_nacimiento DATE NOT NULL
);
```

**Restricción de columna**

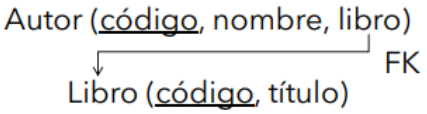
## UNIQUE

Autor (código, nombre, libro)
UK

```
CREATE TABLE autor (
     codigo INTEGER,
     nombre VARCHAR(50),
     libro VARCHAR(50) UNIQUE
);
```

**Restricción de columna**

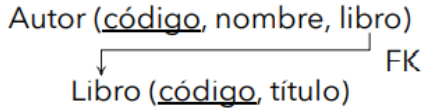
## FOREIGN KEY




A la hora de crear la clave ajena hay que tener en cuenta que:

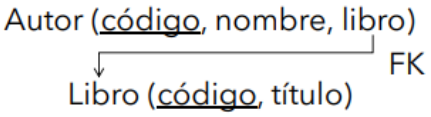
- El tipo de la columna que es clave ajena debe ser del mismo tipo que la clave primaria de la tabla a la que referencia.
- La columna referenciada debe ser clave primaria.

## FOREIGN KEY



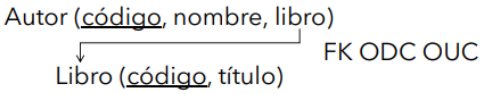

```
CREATE TABLE autor (
     codigo INTEGER PRIMARY KEY,
     nombre VARCHAR(50),
     libro INTEGER,
     CONSTRAINT autor_libro_fk FOREIGN KEY (libro)
     REFERENCES libro (codigo)
);
```

## FOREIGN KEY




```
CREATE TABLE autor (
     codigo INTEGER PRIMARY KEY,
     nombre VARCHAR(50),
     libro INTEGER,
     FOREIGN KEY (libro) REFERENCES libro (codigo)
);
```

## FOREIGN KEY




```
CREATE TABLE autor (
   codigo INTEGER PRIMARY KEY,
   nombre VARCHAR(50),
   libro INTEGER,
   FOREIGN KEY (libro) REFERENCES libro (codigo)
        ON DELETE CASCADE ON UPDATE CASCADE
);
```

⚠ La restricción FOREIGN KEY solo se puede definir como restricción de tabla

## CHECK

Son restricciones que establecen una condición que deben cumplir los datos de una columna.

- La condición se expresa entre paréntesis y debe devolver un resultado booleano (*true* o *false*).
- Podemos utilizar los operadores lógicos y relacionales:
  >, <, >=, !=, <=, =, IS NULL, IS NOT NULL, AND, NOT, OR.
- Un campo puede tener varias restricciones CHECK.

## CHECK

```
CREATE TABLE ingresos (
     cod INT PRIMARY KEY,
     concepto VARCHAR (40) NOT NULL,
     importe INT,
     CONSTRAINT ingresos_importe_ck
                CHECK (importe > 0)
);
```

## ALTER TABLE

Sentencia que nos permite:

- Modificar el nombre de una tabla.
- Añadir, borrar o modificar las columnas.
- Añadir y eliminar restricciones de la tabla.

## ALTER TABLE

Añadir columnas a una tabla.

```
ALTER TABLE nombreTabla ADD COLUMN nombreColumna
definicionColumna;
```

Ejemplo:

```
ALTER TABLE persona ADD COLUMN nombre VARCHAR(50);
```

## ALTER TABLE

Añadir una **restricción de tabla**.

```
ALTER TABLE nombreTabla ADD [CONSTRAINT] restriccion;
```

Ejemplos:

```
ALTER TABLE persona ADD PRIMARY KEY(id);
ALTER TABLE persona ADD CONSTRAINT PRIMARY KEY(id);
ALTER TABLE persona ADD CONSTRAINT persona_id_pk PRIMARY
KEY(id);
```

## AUTO_INCREMENT

Genera automáticamente valores numéricos secuenciales cada vez que se inserta una fila.

Por ejemplo:

```
CREATE TABLE autor (
    codigo INTEGER PRIMARY KEY AUTO_INCREMENT,
    nombre VARCHAR(50)
);
```

Al insertar varias filas:

```
INSERT INTO autor VALUES(DEFAULT, 'Juan');
INSERT INTO autor VALUES(DEFAULT, 'Andrea');
```