Bases de Datos y SQL

- Introducción a Base de Datos Relacionales
  - Definición, conceptos principales
  - Entidades y relaciones
- Lenguaje de consultas estructuradas SQL
  - Notación
  - Comandos
    - Select
    - Insert
    - Delete
    - Update
    - Join
- Modelo de Objetos y Modelo de Entidad Relación

Base de Datos Relacional
Definiciones

Base de Datos

Una **base de datos** es un conjunto organizado de datos que se almacena y se gestiona electrónicamente, permitiendo a los usuarios acceder, modificar y administrar esos datos de manera eficiente. Son esenciales para almacenar información de manera estructurada.
Existen varios tipos de bases de datos y motores que son ampliamente utilizados según las necesidades de las aplicaciones. Las más utilizadas son las bases de datos relacionales, pero también se usan las base de datos noSQL (MondoDB, CouchDB), de tiempo real (InFLUXDB), orientadas a objetos (ObjectDB), etc..

Base de Datos Relacional (RDBMS)

Organizan los datos en tablas que están relacionadas entre sí a través de claves. Utilizan el lenguaje SQL (Structured Query Language) para la gestión y consulta de datos.
Motores mas populares: MySQL, PostgreSQL, Oracle, Microsoft SQL Server, SQLite.

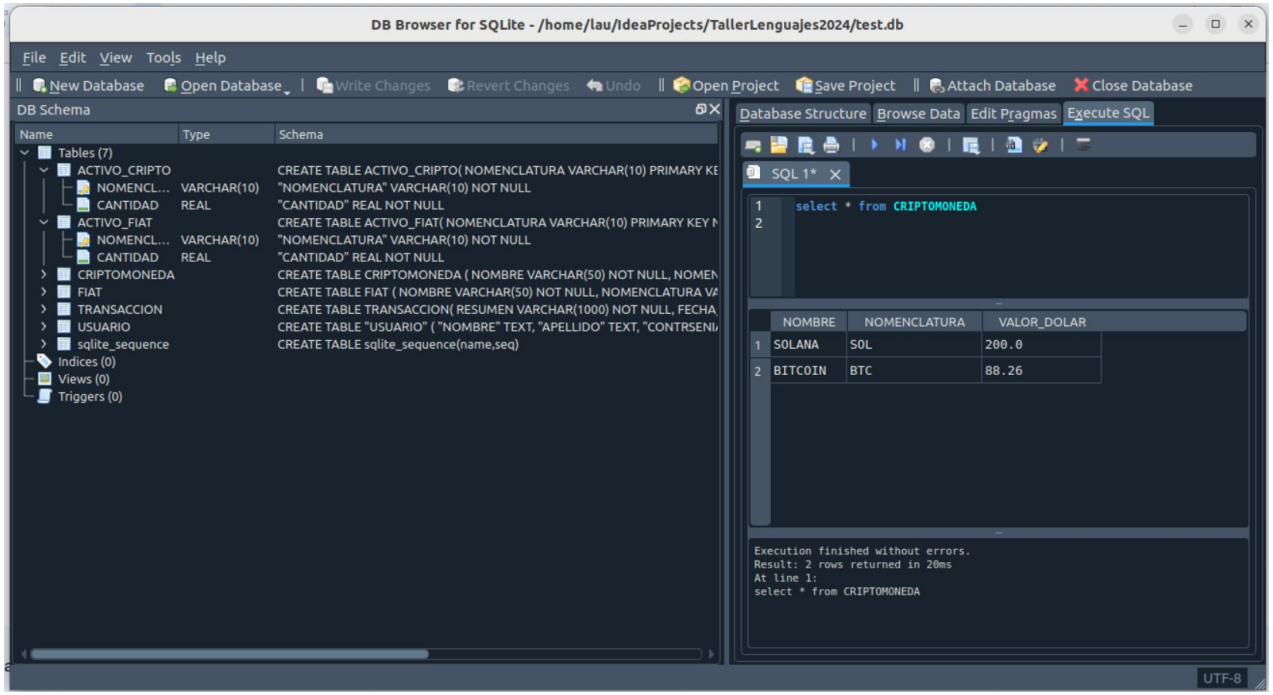
Entidad

Una entidad es una representación de un concepto que puede identificarse de manera única dentro de un sistema o dominio. Una entidad se representa dentro de una base de datos como una **tabla**. Una entidad puede representar **objetos concretos** (como una persona, un coche, un producto) o **conceptos abstractos** (como una transacción, una reserva, una factura).

Instalación del Gestor de Base de Datos - SQLite

SQLite es una DB portable, es decir no necesitamos tener una servidos, ponemos el jar en el proyecto y listo Trae un browser, con una GUI que nos ayuda a administrar la BD

La página oficial es **`https://www.sqlite.org/`**



UBUNTU
sudo apt install sqlite3
sudo apt install sqllitebrowser

Base de Datos Relacional
Definiciones

Entidad

Una entidad es una representación de un concepto que puede identificarse de manera única dentro de un sistema o dominio. Una entidad se representa dentro de una base de datos como una **tabla**. Una entidad puede representar **objetos concretos** (como una persona, un coche, un producto) o **conceptos abstractos** (como una transacción, una reserva, una factura).

Una entidad tiene atributos (propiedades). Dentro de las propiedades puede tener:

- clave primaria (primary key-pk). atributo o conjunto de atributos distingue a cada instancia de la entidad dentro del sistema
- clave foránea (foreign key-fk): atributo en una entidad que se refiere a la clave primaria de otra entidad, creando la relación entre ambas.

Tabla contry

```
SELECT * FROM country;
```

id	name	population	confederation_id
1	Argentina	45.808.747	3
2	Spain	47.265.321	1
3	Mexico	115.296.767	2
4	Colombia	47.846.160	3
5	Costa Rica	4.586.353	2

Tabla confederations

```
SELECT * FROM confederation;
```

id	initials	name
1	UEFA	Unión de Asociaciones Europeas de Fútbol
2	CONCACAF	Confederación de Norteamérica, Centroamérica
3	CONMEBOL	Confederación Sudamericana de Fútbol
4	CAF	Confederación Africana de Fútbol

SQL

Lenguaje de consultas estructuradas

SQL es un lenguaje estándar de programación. Nos permite guardar la información y recuperarla de manera relativamente fácil. Que sea estándar significa que todos los motores de base de datos o gestores lo deben manejar.

SQL es un lenguaje de consultas estructuradas que permite realizar muchas funcionalidades como:

- Crear DB
- Crear, modificar y borrar tablas
- Insertar, borrar datos en una tabla

Existen muchos comandos pero los más usados son los que nos permiten hacer operaciones CRUD (Create, Read, Update y Delete)

- INSERT → Create
- SELECT → Read
- UPDATE → Update
- DELETE → Delete
- JOIN → Une datos de dos tablas

SQL
Sentencia CREATE DB y CREATE TABLE

El comando CREATE DB crea una base de datos, con el nombre especificado. A partir de ahí se pueden empezar a crear las tablas.

```
CREATE DATABASE 'wallet'
```

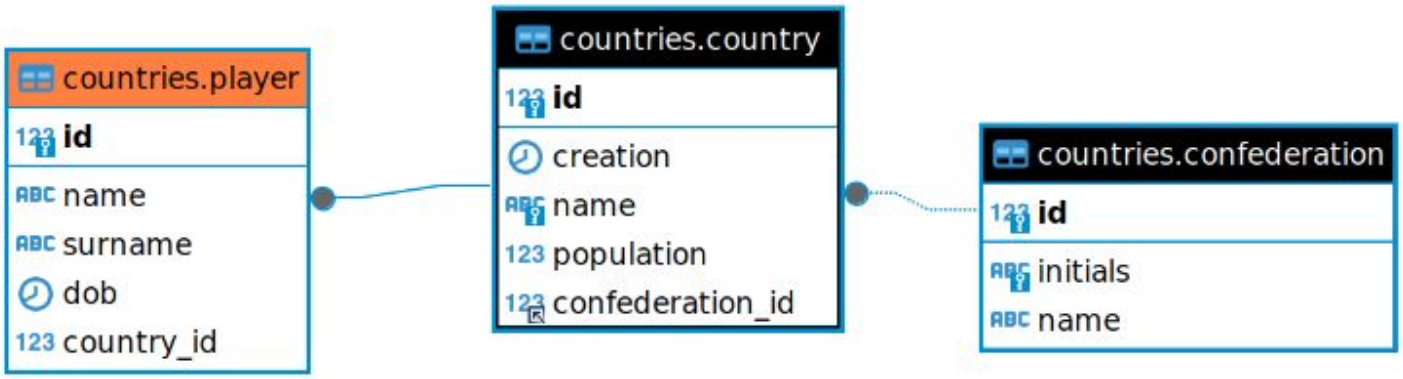
El comando CREATE TABLE crea una tabla en la base de datos en uso.

```
CREATE TABLE 'moneda' (
 'nomenclatura' varchar(10) NOT NULL,
 'nombre' varchar(50) NOT NULL,
 'valor_dolar' real NOT NULL,
 'volatilidad' real DEFAULT NULL,
 'stock_real' real DEFAULT NULL,
 'tipo' char(1) NOT NULL,
 PRIMARY KEY ('nomenclatura'));
```

SQL

Sentencia CREATE DB y CREATE TABLE

Para los ejemplos que veremos en esta clase, consideremos las siguientes tres tablas y sus relaciones:



En este ejemplo la tabla **`country`** tiene un atributo **`id`** como su clave primaria y **`confederation_id`** como una clave foránea (apunta a la clave primaria de otra tabla)

```
CREATE TABLE 'country' (
 'id' bigint NOT NULL AUTO_INCREMENT,
 'creation' datetime NOT NULL,
 'name' varchar(255) NOT NULL,
 'population' int NOT NULL,
 `confederation_id` int NOT NULL,
 PRIMARY KEY ('id'),
 FOREIGN KEY ('confederation_id') REFERENCES
'confederation' ('id'));
```

```
CREATE TABLE `confederation` (
  'id' int NOT NULL AUTO_INCREMENT,
  'initials' varchar(20) NOT NULL,
  'name' varchar(100) DEFAULT NULL,
 PRIMARY KEY (`id`));
```

SQL
Sentencia SELECT

El comando SELECT recupera datos desde una base de datos y puede utilizar diferentes cláusulas. Los datos retornados por el SELECT son almacenados en una tabla en memoria y son llamados **result-set**. Algunos ejemplos:

```
SELECT * FROM country;
```

Tabla **country**

id	creation	name	population	confederation_id
1	2018-10-06 18:10:48.000	Argentina	45.808.747	3
2	2018-10-06 18:10:48.000	Spain	47.265.321	1
3	2018-10-06 18:10:48.000	Francia	47.846.160	1
4	2018-10-06 18:10:48.000	Colombia	51.609.000	3
5	2018-10-06 18:10:48.000	Costa Rica	5.129.910	2

```
SELECT name, population FROM country WHERE confederation_id=3;
```

```
SELECT name, population FROM country
WHERE confederation_id=3 AND population>46000000;
```

name	population
Colombia	51.609.000

```
SELECT * FROM country ORDER BY name;
```

SQL
Sentencia INSERT

El comando INSERT es utilizado para agregar un registro en una tabla.

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...)
```

Tabla **player**

id	name	surname	dob
1	Leonel	Messi	1988-11-06 00:00:00.000
2	Angel	Di Maria	1989-02-02 00:00:00.000
3	Kylian	Mbappé	1988-11-06 00:00:00.000
4	Cristiano	Ronaldo	1988-11-06 00:00:00.000

```
INSERT INTO player (name, surname, dob)
VALUES('Marc', 'Cucurella', '1997-10-06 18:10:48.000');
```

¿Por qué no insertamos nada en el campo **id**?
Como la columna **id** se configuró para que sea **auto-increment** será generada automáticamente cada vez que se inserte un registro en la tabla.

Se pueden insertar solo algunos campos. Prestar atención con el orden!

SQL
Sentencia INSERT y valores NULL

¿Qué es un valor NULL?
Un campo con un valor NULL es un campo sin valor. Esto se puede dar cuando insertamos un registro y no mandamos un valor para cada uno de los campos.

Si la tabla tiene un campo que es **optional**, es posible insertar un nuevo registro o actualizar un registro existente sin agregar un valor a ese campo. En este caso, será salvado con valor NULL.

¿Cómo testear por valores NULL?
No se pueden usar los operadores 0 o <>, se debe usar usar alguno de los operadores `IS NULL` o `IS NOT NULL`. La sintaxis es así:

```
SELECT column_names FROM table_name
WHERE column_name IS NOT NULL/IS NULL;
```

Devuelve el `name` y `surname` del jugador que no tiene cargada la fecha de nacimiento.

```
SELECT name, surname
FROM player WHERE dob IS NULL;
```

id	name	surname	dob
1	Leonel	Messi	1988-11-06 00:00:00.000
2	Angel	Di Maria	1989-02-02 00:00:00.000
3	Kylian	Mbappé	[NULL]
4	Cristiano	Ronaldo	1988-11-06 00:00:00.000
5	Marc	Cucurella	1997-10-06 18:10:48.000

SQL
Sentencia UPDATE

El comando UPDATE es utilizado para actualizar los datos de registros existentes.

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

Note: Hay que ser cuidadosos al actualizar datos en una tabla. Notar que el UPDATE tiene la cláusula WHERE. Esta cláusula WHERE especifica cuál/es registro/s será/n actualizado/s. Si se omite el WHERE, se actualizarán todos los registros de la tabla!

Esta sentencia actualiza la fecha de nacimiento de Mbappé que está en NULL.

```
UPDATE countries.Player
SET dob = '1986-12-02 00:00:00.000'
WHERE id = 3;
```

id	name	surname	dob
1	Leonel	Messi	1988-11-06 00:00:00.000
2	Angel	Di Maria	1989-02-02 00:00:00.000
3	Kylian	Mbappé	[NULL]
4	Cristiano	Ronaldo	1988-11-06 00:00:00.000
5	Marc	Cucurella	1997-10-06 18:10:48.000

id	name	surname	dob
1	Leonel	Messi	1988-11-06 00:00:00.000
2	Angel	Di Maria	1989-02-02 00:00:00.000
3	Kylian	Mbappé	1986-12-02 00:00:00.000
4	Cristiano	Ronaldo	1988-11-06 00:00:00.000
5	Marc	Cucurella	1997-10-06 18:10:48.000

SQL
Sentencia DELETE

El comando DELETE borra registros existentes en una tabla de la base de datos.

```
DELETE FROM table_name WHERE condition;
```

Note: Hay que ser cuidadosos cuando se utilizar el comando DELETE. Notar que el DELETE tiene la cláusula WHERE. Esta cláusula WHERE especifica cuál/es registro/s será/n borrado/s definitivamente de la tabla. Si se omite el WHERE, se borrarán todos los registros de la tabla!

Esta sentencia borra el jugador de name = 'Pepe' de la tabla de la izquierda:

```
DELETE FROM player WHERE name='Pepe'
```

id	name	surname	dob	country_id
1	Leonel	Messi	1988-11-06 00:00:00.000	1
2	Angel	Di Maria	1989-02-02 00:00:00.000	1
3	Kylian	Mbappé	1986-12-02 00:00:00.000	3
4	Cristiano	Ronaldo	1988-11-06 00:00:00.000	6
5	Marc	Cucurella	1997-10-06 18:10:48.000	2
6	Pepe	Cucurella	1997-10-06 18:10:48.000	2

id	name	surname
1	Leonel	Messi
2	Angel	Di Maria
3	Kylian	Mbappé
4	Cristiano	Ronaldo
5	Marc	Cucurella

```
SELECT id, name, surname
FROM player
```

SQL
Operador LIKE y comodines

El operador `LIKE` se usa en conjunción con el signo % para recuperar datos que cumplan con un patrón. El signo porcentaje, % representa uno o múltiples caracteres.

```
SELECT column1, column2, ...
FROM table_name
WHERE columnN LIKE pattern;
```

Devuelve todos los jugadores que su nombre contenga una letra o

```
SELECT * FROM countries.Player
WHERE name LIKE '%o%';
```

id	name	surname	dob	country_id
1	Leonel	Messi	1988-11-06 00:00:00.000	1
4	Cristiano	Ronaldo	1988-11-06 00:00:00.000	6

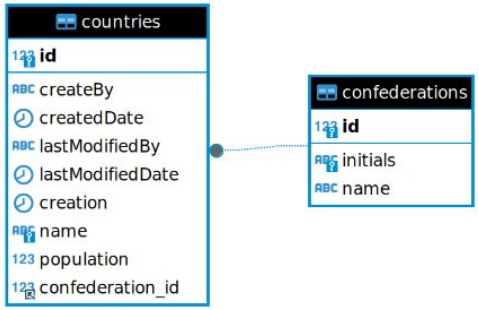
Devuelve todos los jugadores cuyos nombres finalicen con o

```
SELECT * FROM countries.Player
WHERE name LIKE '%o';
```

id	name	surname	dob	country_id
4	Cristiano	Ronaldo	1988-11-06 00:00:00.000	6

SQL
JOIN

La cláusula `JOIN` es usada para combinar filas desde dos o más tablas, usando columnas comunes.



SELECT * FROM countries

	id	creation	name	population	confederation_id
1	1	2018-10-06 18:10:48.000	Argentina	45.808.747	3
2	2	2018-10-06 18:10:48.000	Spain	47.265.321	1
3	3	2018-10-06 18:10:48.000	Mexico	115.296.767	2
4	4	2018-10-06 18:10:48.000	Colombia	47.846.160	3
5	5	2018-10-06 18:10:48.000	Costa Rica	4.586.353	2

SELECT * FROM confederations

	id	initials	name
1	1	UEFA	Unión de Asociaciones Europeas de Fútbol
2	2	CONCACAF	Confederación de Norteamérica, Centroamérica y el Caribe de Fútbol
3	3	CONMEBOL	Confederación Sudamericana de Fútbol
4	4	CAF	Confederación Africana de Fútbol

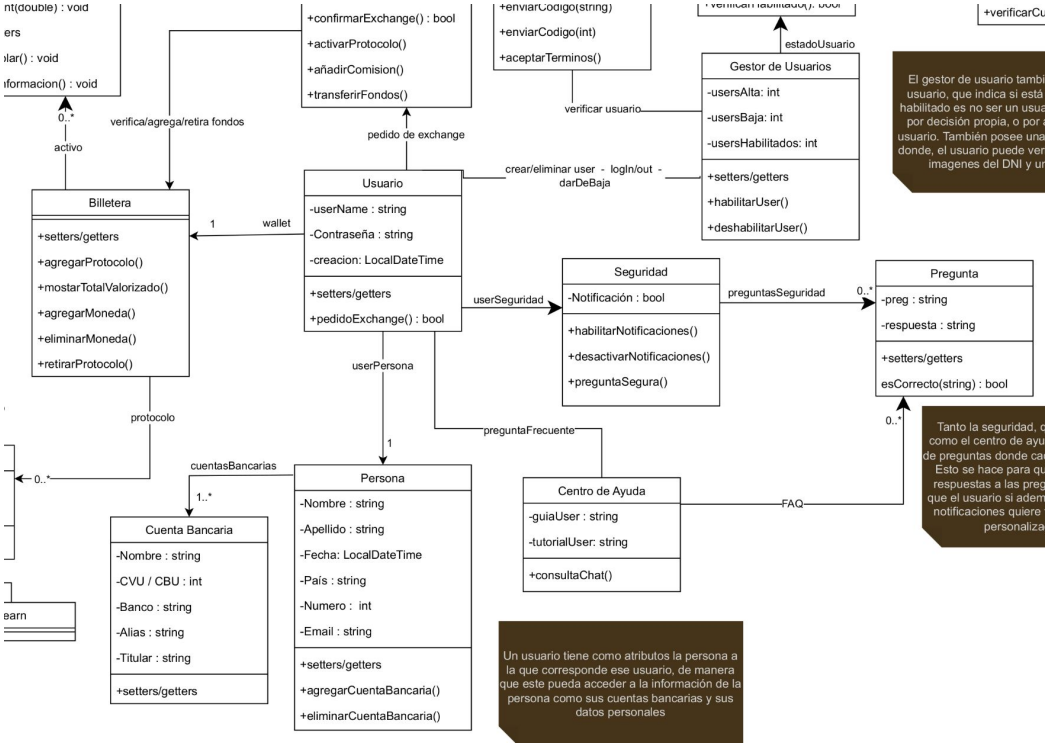
```
SELECT confederations.id, confederations.initials, confederations.name,
countries.id, countries.name, countries.population
FROM confederations
INNER JOIN countries ON confederations.id=countries.confederation_id
```

SELECT confederations.id, confederati

	id	initials	name	id	name	population
1	1	UEFA	Unión de Asociaciones Europeas de Fútbol	1	Norway	5.136.700
2	1	UEFA	Unión de Asociaciones Europeas de Fútbol	2	Spain	47.265.321
3	2	CONCACAF	Confederación de Norteamérica, Centroamérica y el Caribe de Fútbol	3	Mexico	115.296.767
4	3	CONMEBOL	Confederación Sudamericana de Fútbol	4	Colombia	47.846.160
5	2	CONCACAF	Confederación de Norteamérica, Centroamérica y el Caribe de Fútbol	5	Costa Rica	4.586.353

Modelo de Objetos y Modelo de Entidad Relación

Tener en cuenta que no todos los objetos definidos en el Modelo de Objetos del ENTREGABLE 1 serán persistidos en la base de datos.
Dentro del dominio de una aplicación existen múltiples objetos que no se persisten como:
Objetos que brindan servicios, los DAO, los objetos de interfaz de usuario, clases auxiliares, etc. Acá se ve un fragmento de un UML de un grupo:



Hay clases como **Gestor de Usuarios** o **Seguridad** por mencionar alguna de las clases seguramente no se persistirán en la base de datos. Estas clases son parte de la lógica de negocios, brindarán servicios pero no se persisten.