Creating the SQL Schema

Second Lab Session

From E-R to Database

Lesson Objectives

✅ Transform the **E-R** model into an **SQL schema**
✅ Define **tables, primary keys and foreign keys**
✅ Apply **integrity constraints**
✅ Create the SQL schema for the **University Library** and **Sports Swimming Pool**

From E-R to SQL Schema

- 📌 Each **entity** becomes a **table**
- 📌 **Attributes** become **columns**
- 📌 **Relationships** become **foreign keys**

Primary and Foreign Keys

📌 **Primary Key (`PRIMARY KEY`)** → Uniquely identifies each row
📌 **Foreign Key (`FOREIGN KEY`)** → Creates a link between two tables

University Library Database

📌 We create tables for:

✅ **Users**

✅ **Books**

✅ **Loans**

Library SQL Schema

```
CREATE DATABASE Library;
\c Library;

CREATE TABLE Users (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    user_type VARCHAR(20)
);
```

SQLite - Library Schema

Differences:

- `CREATE DATABASE` and `\c` do not exist in SQLite (a database is simply a file);
- `SERIAL` → `INTEGER PRIMARY KEY` (SQLite auto-increments it);
- add `PRAGMA foreign_keys = ON;` to enable foreign key enforcement.

```sql
PRAGMA foreign_keys = ON;

CREATE TABLE Users (
    id INTEGER PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    user_type VARCHAR(20)
);
```

Books Table

```
CREATE TABLE Books (
    id SERIAL PRIMARY KEY,
    title VARCHAR(200) NOT NULL,
    author VARCHAR(100) NOT NULL,
    genre VARCHAR(50),
    publication_year INT CHECK (publication_year > 0),
    available_copies INT CHECK (available_copies >= 0)
);
```

SQLite - Books Table

📌 **Difference:** `SERIAL` is not supported in SQLite. Use `INTEGER PRIMARY KEY` – SQLite automatically assigns an incrementing value.

```
CREATE TABLE Books (
    id INTEGER PRIMARY KEY,
    title VARCHAR(200) NOT NULL,
    author VARCHAR(100) NOT NULL,
    genre VARCHAR(50),
    publication_year INT CHECK (publication_year > 0),
    available_copies INT CHECK (available_copies >= 0)
);
```

Loans Table

```
CREATE TABLE Loans (
    id SERIAL PRIMARY KEY,
    user_id INT REFERENCES Users(id) ON DELETE CASCADE,
    book_id INT REFERENCES Books(id) ON DELETE CASCADE,
    loan_date DATE NOT NULL,
    return_date DATE,
    CONSTRAINT unique_loan UNIQUE (user_id, book_id, loan_date)
);
```

SQLite - Loans Table

📌 **Difference:** `SERIAL` → `INTEGER PRIMARY KEY`. Foreign key constraints (`ON DELETE CASCADE`) are only enforced if `PRAGMA foreign_keys = ON` has been set at the start of the session.

```
CREATE TABLE Loans (
    id INTEGER PRIMARY KEY,
    user_id INT REFERENCES Users(id) ON DELETE CASCADE,
    book_id INT REFERENCES Books(id) ON DELETE CASCADE,
    loan_date DATE NOT NULL,
    return_date DATE,
    CONSTRAINT unique_loan UNIQUE (user_id, book_id, loan_date)
);
```

Sports Swimming Pool Database

📌 We create tables for:

✅ **Customers**
✅ **Subscriptions**
✅ **Lanes**
✅ **Swimming Sessions**
✅ **Bookings**

Swimming Pool SQL Schema

```
CREATE DATABASE Pool;
\c Pool;

CREATE TABLE Customers (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    registration_date DATE NOT NULL,
    subscription_type VARCHAR(20));
```

SQLite - Pool Schema

📌 **Differences:** no `CREATE DATABASE` or `\c;` `SERIAL` → `INTEGER PRIMARY KEY;` add `PRAGMA foreign_keys = ON;`.

```
PRAGMA foreign_keys = ON;

CREATE TABLE Customers (
    id INTEGER PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    registration_date DATE NOT NULL,
    subscription_type VARCHAR(20)
);
```

Subscriptions Table

```
CREATE TABLE Subscriptions (
    id SERIAL PRIMARY KEY,
    type VARCHAR(20) UNIQUE NOT NULL,
    price DECIMAL(8,2) NOT NULL CHECK (price >= 0),
    duration INT CHECK (duration > 0)
);

ALTER TABLE Customers ADD COLUMN subscription_id INT
REFERENCES Subscriptions(id) ON DELETE SET NULL;
```

SQLite - Subscriptions Table

📌 **Differences:** `SERIAL` → `INTEGER PRIMARY KEY`. Note: `DECIMAL(8,2)` is accepted but SQLite stores it as a floating-point number – decimal precision is not enforced.

```
CREATE TABLE Subscriptions (
    id INTEGER PRIMARY KEY,
    type VARCHAR(20) UNIQUE NOT NULL,
    price DECIMAL(8,2) NOT NULL CHECK (price >= 0),
    duration INT CHECK (duration > 0)
);

ALTER TABLE Customers ADD COLUMN subscription_id INT
REFERENCES Subscriptions(id) ON DELETE SET NULL;
```

Lanes Table

```
CREATE TABLE Lanes (
    id SERIAL PRIMARY KEY,
    lane_number INT UNIQUE NOT NULL CHECK (lane_number > 0),
    depth DECIMAL(4,2) CHECK (depth > 0),
    width DECIMAL(4,2) CHECK (width > 0)
);
```

SQLite - Lanes Table

📌 **Difference:** SERIAL → INTEGER PRIMARY KEY.

```
CREATE TABLE Lanes (
    id INTEGER PRIMARY KEY,
    lane_number INT UNIQUE NOT NULL CHECK (lane_number > 0),
    depth DECIMAL(4,2) CHECK (depth > 0),
    width DECIMAL(4,2) CHECK (width > 0)
);
```

Instructors Table

```
CREATE TABLE Instructors (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    specialization VARCHAR(50)
);
```

SQLite - Instructors Table

📌 **Difference:** SERIAL → INTEGER PRIMARY KEY.

```
CREATE TABLE Instructors (
    id INTEGER PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    specialization VARCHAR(50)
);
```

Swimming Sessions Table

```
CREATE TABLE Sessions (
    id SERIAL PRIMARY KEY,
    time TIME NOT NULL,
    level VARCHAR(20),
    max_participants INT CHECK (max_participants > 0),
    instructor_id INT REFERENCES Instructors(id) ON DELETE SET NULL
);
```

SQLite - Sessions Table

📌 **Differences:** `SERIAL` → `INTEGER PRIMARY KEY`; `time` is a built-in SQLite function name – renamed to `start_time` to avoid conflicts; `TIME` is not a native SQLite type, values are stored as TEXT.

```
CREATE TABLE Sessions (
    id INTEGER PRIMARY KEY,
    start_time TEXT NOT NULL,
    level VARCHAR(20),
    max_participants INT CHECK (max_participants > 0),
    instructor_id INT REFERENCES Instructors(id) ON DELETE SET NULL
);
```

Bookings Table

```
CREATE TABLE Bookings (
    id SERIAL PRIMARY KEY,
    customer_id INT REFERENCES Customers(id) ON DELETE CASCADE,
    session_id INT REFERENCES Sessions(id) ON DELETE CASCADE,
    lane_id INT REFERENCES Lanes(id) ON DELETE CASCADE,
    booking_date DATE NOT NULL,
    UNIQUE (session_id, lane_id, booking_date)
);
```

SQLite - Bookings Table

📌 **Difference:** `SERIAL` → `INTEGER PRIMARY KEY`. Remember that `ON DELETE CASCADE` requires `PRAGMA foreign_keys = ON;`.

```
CREATE TABLE Bookings (
    id INTEGER PRIMARY KEY,
    customer_id INT REFERENCES Customers(id) ON DELETE CASCADE,
    session_id INT REFERENCES Sessions(id) ON DELETE CASCADE,
    lane_id INT REFERENCES Lanes(id) ON DELETE CASCADE,
    booking_date DATE NOT NULL,
    UNIQUE (session_id, lane_id, booking_date)
);
```

Lesson Summary

✅ **Translated the E-R model** into an **SQL schema**
✅ Defined **tables, primary keys and foreign keys**
✅ Applied **integrity constraints**
✅ SQL schema ready to **insert and query data**