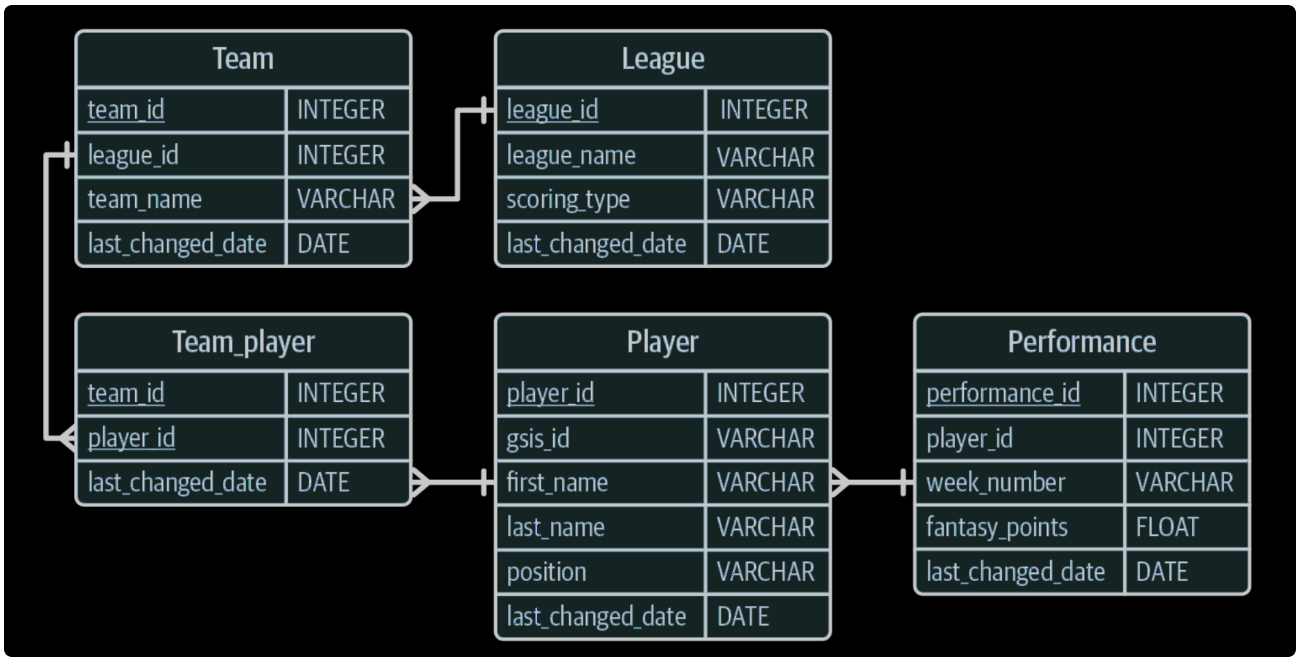
Chapter 03 - Creating Your Database

API Components

API Components

- Database
  - SQLite
    - You can later switch to PostgreSQL or MySql later on
- Database Classes
  - SQLAlchemy
    - Python database toolkit + ORM (Object Relational Mapping (ORM))
    - Jobs of SQLAlchemy:
      - Provide query access to database using Python, without using SQL query
      - It populates Python objects with the data from the source database without requiring any conversion of data types.
        - Keep consistency between Python objects type and database objects
        - Support variety of databases
        - Same Python code for different database
        - Create queries as prepared statements $\xrightarrow{\text{Which}}$ Prevent/Combat SQL injection
  - Handle Querying the Databases + Storing the Data
- API Controller
  - With FastAPI
  - Handle all of the API processing + other functions
- Data Transfer and Validation
  - Ensure that the API requests and responses have valid data + conform to their definitions
  - Pytest
    - Python Testing Library
    - Create Unit Test with it
      - Verify the parts of your code
    - Regression Test

Database



Create an empty sqlite database:

```
sqlite3 fantasy-data.db "VACUUM;"
```

Include External Identifiers For Your API

it make the life of data scientists easier, cause now they can understand data and combine them easier

Your API should support querying by last change date

It is a major time saver

- Primary Key is always unique, enable you to join tables
- Foreign Key enable tables to relate to each others, match the records of child table by parent table
  - More than one Foreign Key = Association Table
    - Enable Many-to-Many relationships
- Enforce foreign keys within sqlite
  - If your child tables record, don't match with parent record = error
    `PRAGMA foreign_keys = ON`

- Prepare import statement to recognize CSV
  `.mode csv`
- Check the `schema.sql` + `import.sql`

Accessing Data Through Python

- You could create a connection $\xrightarrow{\text{then}}$ Execute SQL Query

> **It will enable SQL Injection**
>
> > **Use ORM in this case SQLAlchemy**
> > Handle the Process of reading from database tables and create python objects from them

> **Regression Test**
>
> Finding what you broke when you updated your code or library
> More code coverage = More confidence about the code

Python Files and Their Functionality

- `crud.py`: Helper function to query the database
  - Contains Query Functions
  - CRUD $=$ Create, Write, Update, Delete
- `database.py`: Configures SQLAlchemy to use the SQLite database
  - Tasks that get perform in this file:
    - Create a database connection which points to SQLite database + has a correct setting
    - Create a parent class $\xrightarrow{\text{so?}}$ You use to define the python table classes
- `models.py`: Defines the SQLAlchemy classes related to the database tables
  - Python representation of your database data.
  - Tasks that get perform in this file:
    - Define SQLAlchemy classes to store information from database tables.

    - Describe the relationship between these tables so the Python code can access the related tables.
- `test_crud.py`: The PyTest file to unit-test your SQLAlchemy files

> **Pyright error:** sqlalchemy document