Group Project 2024

Databases and Information Systems

Department of Computer Science
University of Copenhagen

Oliver Fontaine Raaschou <vns328@alumni.ku.dk>
Shatin Nguyen<hlv332@alumni.ku.dk>

9 June, 23:59

1 E/R-diagram

Below is a E/R diagram we used in our project.

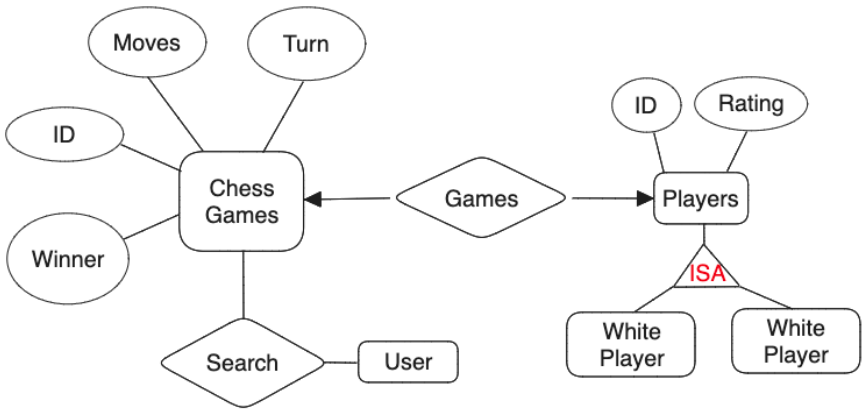

Figure 1: E/R-diagram

2 How to compile

Here is a step-by-step for how to compile our website locally. You need to have Python, Flask and a program to create a database. We used PostgreSQL to create our database, but you can use whatever you prefer. We will not cover how to setup Python, Flask and PostgreSQL and the steps below require that Python, Flask and PostgreSQL is installed.

1. Start with downloading file: `handin.zip` and unzip the folder.
2. `Create a database`: In PostgreSQL start creating database typing:

   CREATE DATABASE chessproject;

3. After creating your database use our `script.sql` to create a `TABLE`. In the script you need to change the location in `FROM` to the folder where `games.csv` is located. When that is done use this command to run the script in a terminal:

   psql -d chessproject -U *username* -f .../*script.sql*

   You need to use the location for the script in the end of the command. If it is success the terminal will output `COPY 19112` or something alike.
4. In `app.py` you need to change *user* and *password* to the username and password your PostgreSQL is setup with.
5. To run the website you can use the following command in the terminal.

   `python app.py`

   If you have Python 3 you might need to type `python3 app.py`. The server is now running can be visited at the following webpage:

   http://127.0.0.1:5000

6. Now you have the options to either Search, Delete, Update, view the dataset or import a game.

3 What can our website do

We have added functionality for the following things: m

3.1 SQL

3.1.1 INSERT

We did not create functionality to ensure the integrity of our database, so when inserting a game you have to make sure that you follow the format below. It is essentially just filling out a HTML form so should be intuitive enough.

```
id VARCHAR(100) (will be generated randomly)
"rated VARCHAR(100)
created_at VARCHAR(100)
last_move_at VARCHAR(100) (can be empty)
turns INT (can be empty)
victory_status VARCHAR(50) (can be empty)
winner VARCHAR(50)
increment_code VARCHAR(50) (can be empty)
white_id VARCHAR(100)
white_rating INT
black_id VARCHAR(100)
black_rating INT
moves TEXT
opening_eco VARCHAR(100)
opening_name VARCHAR(100)
opening_ply INT  (can be empty)
```

3.1.2 UPDATE

For update you can only set a new winner but it will update the database with it. It requires you to have the Game ID before hand so keep that in mind

```
"UPDATE chess_games SET winner = %s WHERE id = %s"
```

3.1.3 SELECT

In most of our queries we are using `SELECT`. Examples:

```
SELECT EXISTS(SELECT 1 FROM chess_games WHERE id = %s)
SELECT id, winner, white_id, black_id, white_rating, black_rating, opening_name
SELECT id, winner, white_id, black_id, white_rating, black_rating, opening_name
```

3.1.4 DELETE

Similar to update we have made functionality so you can delete a game from the database. It also requires you to have the game_id beforehand.

```
DELETE FROM chess_games WHERE id = %s
```

3.2 REGEX

For regular expressions we have added some functionality for some filters to ensure that when searching for games they fit certain criteria that matches our database.
Code:

```
winner_pattern = re.compile["white|black"]
game_id_pattern = re.compile["^[a-zA-Z0-9]{8}$"]
moves_pattern = re.compile["^(\s*(?:[PNBRQK]?[a-h]?[1-8]?[x-]?[a-h][1-8][+#]?|C


def filterWinner(winner):
    return bool(winner_pattern.match(winner)


def filterGameID(id):
    return bool(game_id_pattern.match(id)

def filterMoves(moves):
    return bool(moves_pattern.match(moves):
```