Semester Project: Java Chess

Jamie Wright
CPSC 321, Fall 2024

1 Summary

I designed a chess program database that tracks info about user accounts, games, friendships, and displays it to the user on command. The program it utilizes is a Java chess program that has both a versus and AI mode. The program has a social component as users can send messages and view head-to-head records between each other.

2 Use Cases

2.1 Create Account--Creation

Creating an account with a specified username. User starts with 1000 elo and must work their way up.

2.2 Add Friend--Creation

Add a friend to a certain user’s account.

2.3 Insert Game--Creation

Insert a game into game table after completion. After game is inserted, records and elo are updated.

2.4 Find Friend--Search

Find a specific friend’s info to be displayed to the user.

2.5 Get Leaderboard--Analysis

Find the current leaders in elo.

2.6 Get All Time Leaderboard--Analysis

Find the all-time leaders in elo.

2.7 Remove Friend--Deletion

Remove a friend from an account, including messages.

2.8 Remove Account--Deletion

Remove a user’s account info. Friendships, messages, and games user partook in are also deleted.

2.9 Send Message--Creation

Send a message to another user.

2.10 Get Recent Games--Search

Find the most recent games played by a user.

2.11 Get Recent Messages--Search

Find the most recent messages between two users.

2.12 Get Friends--Search

Find info for all friends of a user.

2.13 Get Average Game Length--Analysis

Finds the average length in turns of all games played by a user.

2.14 Get Turns--Search

Find all turns for a specific game to be displayed to a user.

2.15 Get Head-To-Head--Search

Find the head-to-head record between two users.

2.16 Update Record--Update

Update record for a user after a game is played based on the result.

2.17 Update Head-To-Head--Update

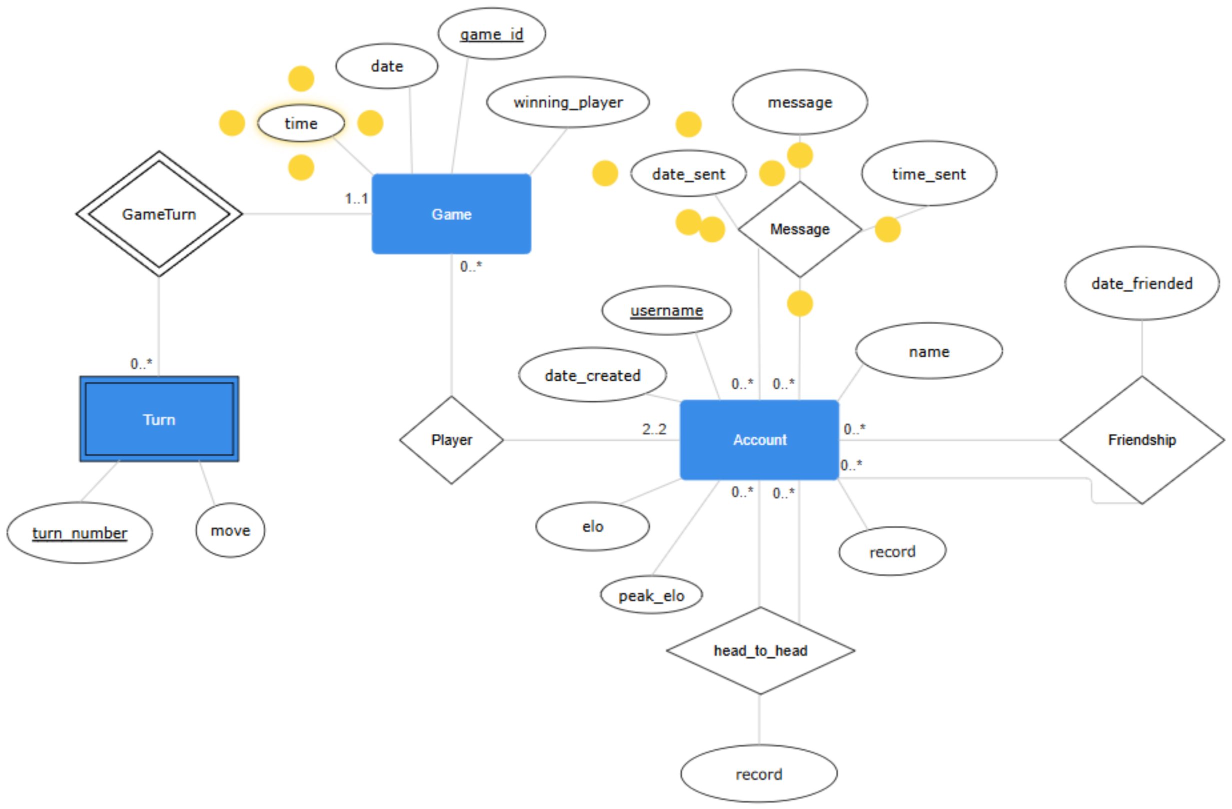
Update head-to-head between users after game based on the result.

2.18 Update Elo--Update

Update elo for a user after a game is played based on the result.

3 Logical Design

3.1 Entity-Relationship Diagram



The Account entity has three relationships with itself. The first is a friendship between two users, the second is the overall record between two players, and the third is a message between two users. The Turn entity is a weak entity to game, and its key is made up of the game_id and turn_number.

3.2 Relational Schema

Account(<u>username</u>, first_name, last_name, date_created, elo, peak_elo, record)

Game(<u>game_id</u>, player_one, player_two, num_turns, date, time, white_won)
Foreign Key(player_one) References Account(username)
Foreign Key(player_two) References Account(username)

Turn(<u>game_id, turn_number</u>, move)
Foreign Key(game_id) References Game(game_id)

Friendship(<u>user_one, user_two</u>, date_friended)
Foreign Key(user_one) References Account(username)
Foreign Key(user_two) References Account(username)

Head_To_Head(user_one, user_two, record)
Foreign Key(user_one) References Account(username)

Foreign Key(user_two) References Account(username)

Message(<u>user_one, user_two, date_sent, time_sent, message</u>)
Foreign Key(user_one) References Account(username)
Foreign Key(user_two) References Account(username)

4 Use-Case SQL Statements

For each use case, give the corresponding SQL statements that you developed. Each SQL statement should support the use case. Be sure to describe each query (one sentence) and state the corresponding parameters (which you can denote as ? or %s, depending on how you implemented them in your applications). Organize this section like section 3 (each use case as a separate subsection).

4.1 Create Account—Creation

Simple insert into account.

INSERT INTO account VALUES (?, ?, ?, ?, 0, 0, ’00-00-00’)

Parameters: Username, first and last name, date created, elo and peak of 0, and a record of 00-00-00

4.2 Add Friend--Creation

Simple insert into friendship.

INSERT INTO friendship VALUES (?, ?, ?)

Parameters: Friend’s username, date, and time.

4.3 Insert Game

Simple insert into game.

INSERT INTO game VALUES (?, ?, ?, ?, ?, ?)

Parameters: Game id, both players, game result, date and time played.

4.4 Find Friend--Search

Returns all info from account for a specific username.

SELECT * FROM account WHERE username = ?

Parameters: Friend’s username.

4.5 Get Leaderboard—Analysis

Gets top 10 elo users by dense ranking over elo column.

```
SELECT username, elo DENSE_RANK() OVER (ORDER BY elo DESC) AS rank account
ORDER BY elo DESC
LIMIT 10
```

Parameters: None.

4.6 Get All Time Leaderboard--Analysis

Gets top 10 elo users of all time by dense ranking over peak_elo column.

```
SELECT username, peak_elo DENSE_RANK() OVER (ORDER BY peak_elo DESC) AS rank account
ORDER BY peak_elo DESC
LIMIT 10
```

Parameters: None.

4.7 Remove Friend—Deletion

Removes friendship row where user and friend or in either user spot.

```
DELETE FROM friendship
WHERE user_one = ? AND user_two = ? OR user_two = ? AND user_one = ?
```

Parameters: User_one, user_two, user_two again, user_one again.

4.8 Remove Account—Deletion

Simple delete from account.

```
DELETE FROM account WHERE username = ?
```

Parameters: Currently singed into account.

4.9 Send Message—Creation

Simple insert into message.

```
INSERT INTO message VALUES (?, ?, ?, ?, ?, ?)
```

Parameters: Both users, date and time sent, message, and which user sent the message.

4.10 Get Recent Games--Search

Finds all games by unioning games where user is white with games where user is black. Orders by date and time and limits to 20 most recent games. Uses a subquery to find all games first in order to make sure limit works correctly across union.

```
SELECT opponent, date, time, game_id
FROM (
        SELECT black_player AS opponent, date, time, game_id
        FROM account JOIN game ON (? = white_player)
        UNION
        SELECT white_player AS opponent, date, time, game_id
        FROM account JOIN game ON (? = black_player)) AS games
ORDER BY date DESC, time DESC
LIMIT 10
```

Parameters: Active user both times.

4.11 Get Recent Messages—Search

Finds all recent messages between two users, ordered by date and time with a limit of 20 most recent messages.

```
SELECT user_sent, message, date_sent
FROM message
WHERE user_one = ? AND user_two = ? OR user_one = ? AND user_two = ?
ORDER BY date_sent DESC, time_sent DESC
LIMIT 20
```

Parameters: Active user and friend, then friend and active user.

4.12 Get Friends—Search

Finds all friends of a user by unioning cases where user is user_one with cases where user is user_two. Ordered by date with a limit of 20 most recent friends.

```
SELECT user_two AS friend, date_friended
FROM friendship
WHERE user_one = ?
UNION
SELECT user_one AS friend, date_friended
FROM friendship
WHERE user_two = ?
ORDER BY date_friended DESC
LIMIT 20
```

Parameters: Both are active user.

4.13 Get Average Game Length--Analysis
Finds average game length for a user. Subqueries to find length of all games for a user by joining account, game, and turn, then uses that data to get the average.

```
SELECT AVG(turns) AS turns
FROM
        //subquery
        (SELECT COUNT(*) AS turns
        FROM account JOIN game ON (username = white_player OR username = black_player)
        JOIN turn USING (game_id)
        WHERE username = ?
        GROUP BY game_id) AS turn_count
```

Parameters: Active user.

4.14 Get Turns—Search

Finds all moves made in a specific game.

```
SELECT move, turn_number
FROM turn JOIN game USING (game_id)
WHERE game_id = ?
ORDER by turn_number ASC
```

Parameters: Game id to be looked at.

4.15 Get Head-To-Head—Search

Finds head-to-head record between two users.

```
SELECT record
From head_to_head
WHERE user_one = ? AND user_two = ?
OR user_one = ? AND user_two = ?
```

Parameters: Active user, friend, friend again, active user again.

4.16 Update Record--Update

Simple update for record in account table.

```
UPDATE account SET record = ? WHERE username = ?
```

Parameters: New record and user that just finished game.

4.17 Update Head-To-Head--Update

Simple update of head-to-head record between two users.

UPDATE head_to_head SET record = ? WHERE user_one = ? OR user_two = ?

Parameters: New head-to-head record and two users who just finished game.

4.18 Update Elo—Update

Simple update of elo in account.

UPDATE account SET elo = ? WHERE username = ?

Parameters: New elo and user who just finished game.

5 Applications

I already had a functioning chess program written in Java that I used to implement this project. I did, however, have to develop a fully robust GUI using Java Swing in order to implement the project. I also needed to make major augmentations to the project such as in the introduction of specific users/players.

6 Conclusions

I accomplished the building of a chess database that stores info about chess games for many accounts. One of the biggest challenges was making sure all the tables worked together (ensuring foreign keys were respected, etc…) Another large challenge was time management, as alongside other finals and projects I found it difficult to accomplish everything I would have liked to. I learned through working on this project that it is extremely valuable to list a plan and use cases for a project before building it, as I believe that saved me much time in the creation process. If I had more time for the project, I would make an improved GUI (in terms of aesthetic) and password-protected accounts top priorities.