Sports arena database

Group number 25

Members:
Daniela Botu, 001569618
Miruna Gherasim, 001718391
Avaneep Kamal: 000426590

Course name: Basics of database systems
Teacher: Iflaah Salman

Table of contents:

1. Introduction

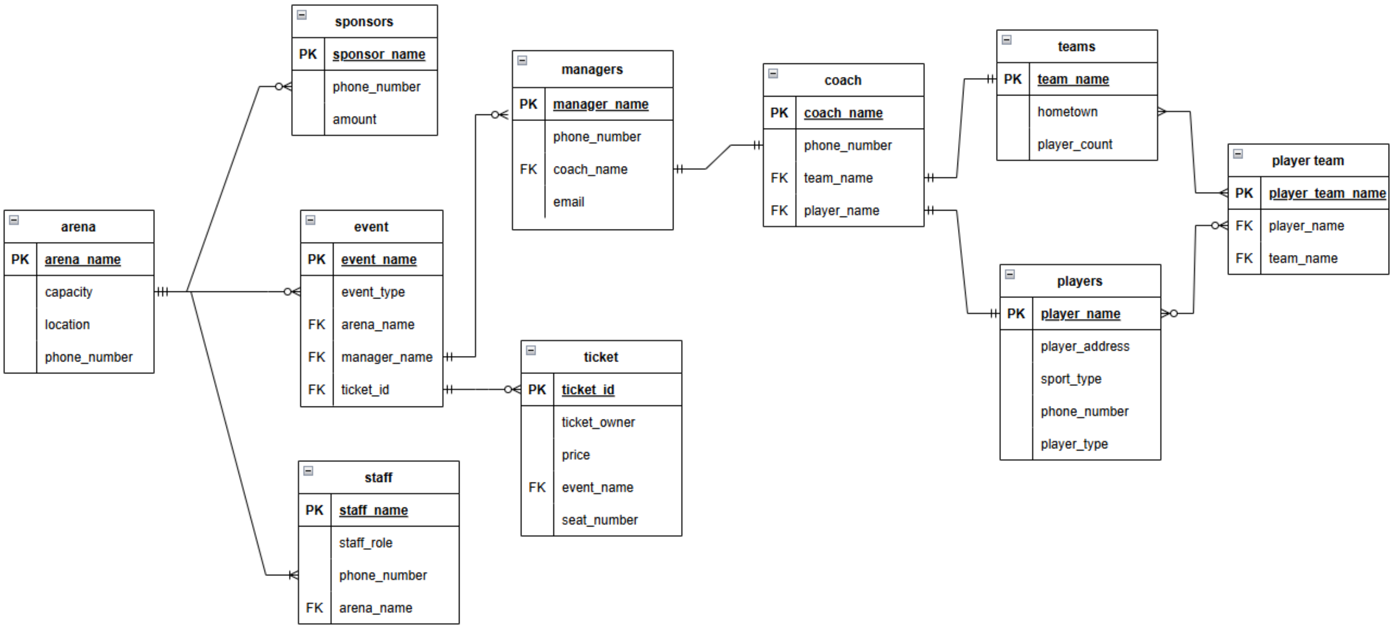
1.1 General description

The general idea of the management information system that we chose to develop, and its database is a sports arena, which mainly focuses on hosting competitions and trainings. The system itself contains the most general and important aspects when using an arena to host events, such as the audience, managers/organizers, staff, coaches, teams, players, and the divertissement, such as shops, that are open to the public.

1.2 Business rules

- The arena is the space where the competitions take place. It has a location, contact information, such as phone number, and a capacity of people that can fit in. The arena also has different sponsors that help organize different events. For each sponsor, their name, phone number and the amount they donated is tracked.
- In order for the competitions to be organized successfully, there are managers and staff that help with it. The staff works for the arena, and the number of staff has to be big in order for the event to run smoothly. For example, there can be one competition with "500" staff, and for each staff, their role and phone number are tracked. The managers work with the event and their number can vary. The event's name and type are tracked.
- For every event there is an audience watching. They secure their places through buying tickets, which tracks the price, the seat number and the event itself that is happening. In order for someone to buy a ticket, they have to register their name on it.
- A coach reports about his pupils situation, as well as gets information, from the manager. In order to do so, he registers his telephone number in the system. A coach can also have one team playing and one solo player in the competition, each of them playing some type of sport. If the player is not playing solo, he will be included in a team. A team's name, hometown, as well as the number of players in it, has to be tracked. There is an additional table showing each player and in what team they play.

2. ERD



3. Relational model

arena (arena_name, capacity, location, phone_number)
sponsors (sponsor_name, phone_number, amount)
event (event_name, event_type, arena_name(FK), manager_name(FK), ticket_id(FK))
managers (manager_name, phone_number, coach_name(FK), email)
staff (staff_name, staff_role, phone_number, arena_name(FK))
ticket (ticket_id, ticket_owner, price, event_name(FK), seat_number)
coach (coach_name, phone_number, team_name(FK), player_name(FK))
teams (team_name, hometown, player_count)
player team (player_team_name, player_name(FK), team_name(FK))
players (player_name, player_address, sport_type, phone_number, player_type)

4. Improvements made to the DB structure

Our initial idea was to keep the database functional, while still maintaining diversity and data integrity.
Firstly, our design included multiple entities, which not only increased the project complexity, but it also increased the possibility of failing to develop it further. During our meeting, we analyzed every entity relationship, and we came to the conclusion of reducing the number of entities, removing three entities – shops, owner, and audience, that didn't have anything to do with our idea of a sports arena where events could be hosted, and we added the "event" and "ticket" entities to simplify associations. The restructuring of the ERD, relational model and finally, database structure made our project manageable and every entity had a real meaning and purpose. The ERD and

relational model helped while doing the database, and it was easier to understand the relations between each entity.
As part of our database optimization, we noticed that in our first iteration the entity “arena” had three foreign keys that were unnecessary. These foreign keys - “sponsor_name”, “event_name”, and “staff_name” were initially included in our ERD, but did not have any functionality, and they led us to have inefficiencies while writing the queries. By eliminating them, we made the database structure easier for us to understand and handle.
Furthermore, in order to have the database more structured, we opted for adding the entity “player_team”, so that we could focus more on implementing them into our system and not the logistics that go through having managers and coaches take care of teams that were formed from more players, and also have individual players managed by the coaches and managers.
With these changes, we finally concluded our attempt to optimize the database structure and worked more on implementing our knowledge in the system we created.

5. Brief description of the program developed to interact with the database

5.1 Programming language chosen

The programming language chosen for the project is Java, with PostgreSQL connection, and Swing framework for the GUI.

5.2 List of some of the SQL queries implemented in the program.

To insert:

```
INSERT INTO Players (player_name, player_address, sport_type, phone_number,
player_type) VALUES (?, ?, ?, ?, ?);
INSERT INTO Teams (team_name, hometown, player_count) VALUES (?, ?, ?);
INSERT INTO Coach (coach_name, phone_number, team_name, player_name)
VALUES (?, ?, ?, ?);
INSERT INTO Managers (manager_name, phone_number, coach_name, email)
VALUES (?, ?, ?, ?);
INSERT INTO Ticket (ticket_id, ticket_owner, price, event_name, seat_number)
VALUES (?, ?, ?, ?, ?);
```

Get events from an arena:

```
SELECT * FROM Event WHERE arena_name = ?;
```

Get staff working at an arena:

```
SELECT * FROM Staff WHERE arena_name = ?;
```

Get tickets from an event:

```
SELECT * FROM Ticket WHERE event_name = ?;
```

Get ticket info with event info:

```
SELECT t.ticket_id, t.ticket_owner, t.price, t.seat_number, e.event_name,
```

e.event_type FROM Ticket t JOIN Event e ON t.event_name = e.event_name WHERE t.ticket_id = ?;

Number of players in a team:
SELECT t.team_name, COUNT(pt.player_name) AS player_count FROM Teams t LEFT JOIN Player_Team pt ON t.team_name = pt.team_name GROUP BY t.team_name;

Update a manager for a specific event:
UPDATE Event SET manager_name = ? WHERE event_name = ?;

Give a manager a different coach:
UPDATE Managers SET coach_name = ? WHERE manager_name = ?;

Get total money sponsored:
SELECT SUM(amount) AS total_sponsorship FROM Sponsors;

6. No AI tools were used to make the theorectial part of the project. GitHub Co-Pilot was used for small parts of the Java code which are not relevant for the grading part, like writing the output text, naming the objects, and modifying the size of the components in the GUI.