BLG 317E - Database Systems
CRN - 13508 / OG CRN
Fall 2025

Basketball League Management System

Team: *Foreign Key*

Team Members:
Celil Aslan (150210703)
Çağatay Dişli (040210081)
Emir Şahin (150220072)
Musa Can Turgut (150210918)
Talip Demir (040210514)

Instructors: Kadir Özlem, Ali Çakmak
Teaching Assistants:
Burcu Kartal, Elif Yıldırım, Selahaddin Şentop, Hacer Akıncı

Due Date: Friday, 10:00, 02/01/2026

Contents

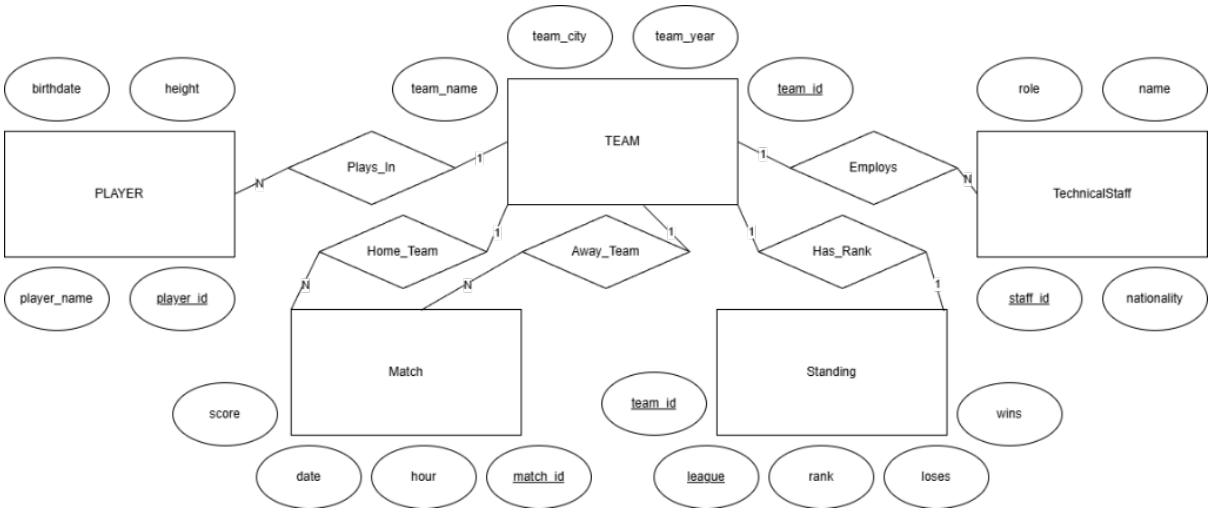
1 Technology Stack

The project relies on a robust and modern technology stack designed for scalability and maintainability:

- **Relational Database Management System (RDBMS):** PostgreSQL is used as the primary database for its advanced features, reliability, and support for complex queries.
- **Backend Development:** Python is the core programming language, utilizing the **Flask** micro-framework to handle HTTP requests, routing, and application logic.
- **Frontend Development:** The user interface is built using HTML5, CSS3, and JavaScript, integrated via Jinja2 templating engine provided by Flask.
- **Containerization:** Docker is employed to containerize the application and database, ensuring consistent environments across development and deployment.

2 Database Design

2.1 ER Diagram



2.2 ER Diagram Explanation

The database schema is designed to manage a basketball league system efficiently. The key entities and their relationships are:

1. **Users:** Stores authentication details for system administrators. Attributes include `id`, `username`, and `password_hash`.
2. **Teams:** Represents basketball teams. Attributes include `team_id`, `team_name`, `league`, `team_city`, and `team_year`. It also links to home venues (`saloon_name`, `saloon_capacity`).
3. **Players:** Contains information about athletes. Attributes include `player_id`, `player_name`, `player_height`, `player_birthdate`, `player_foot`, and `player_bio`. A **Many-to-One** relationship exists between Players and Teams.

4. **Matches:** Detailed records of games played, linked to Teams (Home/Away). Attributes include `match_id`, dates, scores, and locations.
5. **Technic_Roster:** Stores coaching staff data (`staff_id`, `technic_member_name`, `role`). It has a **Many-to-One** relationship with Teams.
6. **Standings:** Captures league rankings and performance metrics (wins, losses, points). It references Teams.

3 Individual Contributions

The project was a collaborative effort with specific responsibilities assigned to each member:

Member	Contribution
Celil Aslan	Focused on the **Matches Module**, handling complex relationships between multiple tables. Implemented extensive **Multi-table JOINs** and aggregations to present match data effectively. Designed sophisticated filtering logic and utilized SQL **Set Operations** like UNION and EXCEPT to generate comprehensive match analytics.
Çağatay Dişli	Responsible for the end-to-end development of the **Players Module**. This involved implementing a robust CRUD system and a dynamic search interface with pagination. A key contribution was the design of complex analytical queries on the 'Advanced Stats' page, utilizing **Correlated Subqueries** and **Nested Queries** to extract deep insights, such as identifying players performing above their team's average.
Emir Şahin	Led the **UI/UX design**, integrating Bootstrap to ensure a responsive and professional interface throughout the application. Implemented the secure **Authentication system**, handling user sessions and password hashing, and performed comprehensive system testing to ensure stability.
Musa Can Turgut	Managed the **Teams Module**, creating user-friendly interfaces for adding, editing, and removing team data. Implemented season-based filtering mechanisms and ensured the correct management of relational data between teams, their home stadiums, and the technical coaching staff.
Talip Demir	Acted as the underlying **Database Architect**, designing the initial ER Diagram and the relational schema. Ensured the database adhered to **Third Normal Form (3NF)** standards to reduce redundancy and defined critical integrity constraints and auxiliary tables like Standings.

4 Dataset and Configuration

4.1 Dataset Outlines

The system is initialized with real-world basketball data imported from CSV files located in the `tables/` directory:

- **team_data.csv:** Basic information for all participating teams.
- **player_data.csv:** Roster details including physical attributes and team affiliations.
- **matches.csv:** Historical match results and schedules.
- **technic_roster.csv:** Coaching staff information.
- **standings.csv:** Current league standings and rankings.

4.2 Connection Details

The application connects to the PostgreSQL database using the `psycopg2` library. The database configuration (host, port, user, password) is managed via environment variables and a dedicated `db_api` module. Use of environment variables within Docker Compose guarantees that connection parameters are securely handled and consistent across containers.

5 Website and Functionalities

5.1 Operational Overview

The web application serves as a comprehensive management tool for basketball leagues. It allows users (administrators and viewers) to browse, search, and manage data related to players, teams, and matches. Access to modification features (Create, Update, Delete) is protected via a secure Login/Register system.

5.2 Navigation and Interface

The application features a responsive navigation bar providing access to the main modules:

1. **Authentication:** Users can register for an account and log in to access administrative features.
2. **Players Module:**
    - **Browsing:** A paginated table listing all players with options to filter by Team and League.
    - **Search & Sort:** Users can search players by name and sort by Name, Age, or Height.
    - **CRUD:** Authenticated users can Add, Edit, and Delete player records.

- **Advanced Stats:** A special page listing interesting statistics, such as players taller than their team average.

3. **Teams Module:** Displays team details, filtering options by season, and administrative capabilities to manage team data.

4. **Matches Module:** A powerful interface to view match schedules, results, and filter games by various criteria.

6 Additional Topics

6.1 Docker Containerization

The entire application is containerized using Docker, allowing for easy deployment. A `docker-compose.yml` file orchestrates two services: the **Flask Web App** and the **PostgreSQL Database**. This setup ensures that the development environment matches production.

6.2 Performance Optimization

To ensure fast query response times, specific indexes were created in the database initialization script (`init_db.py`). Indexes on frequently queried columns such as `team_id`, `match_date`, and `player_name` significantly improve search and filter performance.

7 Difficulties and Solutions

During the development process, the team encountered and resolved several challenges:

- **Data Quality and Type Conversion:** The source CSV data contained inconsistent formats (e.g., "190 cm" vs "1.90m", dates in different formats). *Solution:* We implemented robust regex-based cleaning functions in Python and SQL to normalize data during imports.

- **Complex SQL Queries:** Implementing queries like "Players taller than team average" required referencing the same table multiple times. *Solution:* We utilized **Correlated Subqueries** and carefully designed the logic to ensure accuracy without sacrificing performance.

- **Docker Database Connection Timing:** The web application often started before the database was fully ready to accept connections. *Solution:* We implemented a retry mechanism that pauses the application startup until the PostgreSQL service is responsive.