1. Introduction

This project focuses on developing a University Sports Information System designed to organize and analyze data related to university athletic events, player performance, and community engagement.

Our database team consists of Sharath Ragila, Qixuan Chen, and Olivia Minowa.

The relational database is hosted on the Akira MySQL server under Sharath Ragila's account, in DB3.

2. Mission Statement and Mission Objectives

Mission Statement

The mission of this project is to create a centralized, interactive platform that captures, organizes, and shares comprehensive university sports data—including game schedules, player statistics, and fan interactions—to promote transparency, participation, and athletic excellence within the campus community.

Mission Objectives

- Maintain all university athletic schedules and game results.
- Record and update detailed player information and performance statistics. ● Store and manage user-generated content such as comments, reviews, and ratings. ● Support administrative oversight for coaches and team managers through searchable records. Allow students, fans, and staff to easily access and contribute to sports-related data.
- Ensure data accuracy and consistency through defined relational rules and constraints.

3. Relational Design Basics

The relational structure of our University Sports Information System follows proper database normalization and integrity standards through the creation of Primary Keys (PKs) and Foreign Keys (FKs) in every table. These elements ensure that data remains consistent, uniquely identifiable, and relationally connected across all entities in the database.

Primary Keys (PKs)

Each table in our database includes a unique Primary Key that distinguishes every record and prevents duplication. The PKs for each table are as follows:

- Users → UserId
- Sports → SportId

- Teams → TeamId
- Coaches → CoachId
- Players → PlayerId
- Games → GameId
- Ratings → RatingId
- Comments → CommentId
- StatTypes → StatTypeId
- Stats → StatId

Each of these primary keys is defined as an integer with AUTO_INCREMENT, ensuring every record receives a unique numeric identifier when inserted.

Foreign Keys (FKs)

Foreign Keys are correctly placed to establish relationships between entities and maintain referential integrity. These FKs define how tables interact with each other to model real-world relationships in the university sports environment.

- Teams.SportId → Sports.SportId (Each team is associated with one sport.) ● Coaches.TeamId → Teams.TeamId (Each coach manages one team.) ● Players.TeamId → Teams.TeamId (Each player belongs to one team.) ● Games.SportId → Sports.SportId (Each game belongs to one sport.) ● Games.HomeTeamId → Teams.TeamId and Games.AwayTeamId → Teams.TeamId (Each game connects two teams.)
- Ratings.UserId → Users.UserId (Each rating is submitted by one user.) ● Ratings.GameId → Games.GameId and Ratings.PlayerId → Players.PlayerId (Ratings link to specific games or players.)
- Comments.UserId → Users.UserId (Each comment belongs to one user.) ● Comments.GameId → Games.GameId, Comments.PlayerId → Players.PlayerId, Comments.ParentCommentId → Comments.CommentId (Comments link to games, players, or other comments.)
- StatTypes.SportId → Sports.SportId (Each statistic type belongs to one sport.) ● Stats.GameId → Games.GameId, Stats.PlayerId → Players.PlayerId, Stats.StatTypeId → StatTypes.StatTypeId (Each stat connects player performance to a specific game and stat category.)

Summary

Every table includes a properly defined Primary Key, and all Foreign Keys are correctly placed to model one-to-many and many-to-many relationships as appropriate.
This structure ensures:

- Entity integrity (through PKs),
- Referential integrity (through FKs),
- Logical normalization (through linking tables like Stats, Ratings, and Comments).

As a result, the relational design fully supports accurate data entry, reliable joins across entities, and efficient query performance while preserving data consistency across the entire database.

4. Conceptual Design

The conceptual design of our University Sports Information System illustrates how the database entities relate to one another using an Entity–Relationship Diagram (ERD). Each entity represents a key component of the system, such as users, players, games, or teams. The ERD displays table names, fields with consistent table prefixes, and identifies both primary keys (PK) and foreign keys (FK). Crow's-foot notation is used to show relationships between entities, along with optional or mandatory participation symbols and degree of participation using (min,max) labels.

Entities and Attributes (with prefixes and keys)

Users
PK: UserId
Attributes: UserUsername, UserStudentId, UserEmail, UserType, UserLastLogin, UserCreateDate, UserDeleteDate

Sports
PK: SportId
Attributes: SportName

Teams
PK: TeamId
FK: SportId → Sports.SportId
Attributes: TeamName, TeamFoundedYear, TeamSchool, TeamDivision

Coaches
PK: CoachId
FK: TeamId → Teams.TeamId
Attributes: CoachName, CoachResume

Players
PK: PlayerId
FK: TeamId → Teams.TeamId
Attributes: PlayerName, PlayerStudentYear, PlayerGender, PlayerHeight, PlayerWeight, PlayerJerseyNumber, PlayerPosition

Games
PK: GameId
FK: SportId → Sports.SportId
FK: HomeTeamId → Teams.TeamId

FK: AwayTeamId → Teams.TeamId
Attributes: GameDate, GameVenue, GameStartTime, GameStatus, HomeFinalScore, AwayFinalScore

StatTypes
PK: StatTypeId
FK: SportId → Sports.SportId
Attributes: StatName, StatUnit

Stats
PK: StatId
FK: GameId → Games.GameId
FK: PlayerId → Players.PlayerId
FK: StatTypeId → StatTypes.StatTypeId
Attributes: StatTypeValue

Ratings
PK: RatingId
FK: UserId → Users.UserId
FK (nullable): GameId → Games.GameId
FK (nullable): PlayerId → Players.PlayerId
Attributes: RatingTime, RatingScore

Comments
PK: CommentId
FK: UserId → Users.UserId
FK (nullable): GameId → Games.GameId
FK (nullable): PlayerId → Players.PlayerId
FK (nullable, self-ref): ParentCommentId → Comments.CommentId Attributes: CommentTime, CommentContent, CommentLikes, CommentIsApproved, CommentIsDeleted, CommentIsInvalid

Relationship Descriptions (with Crow's-foot and (min,max) participation)

Each relationship below shows both directions of participation using crow's-foot notation symbols and (min,max) labels.

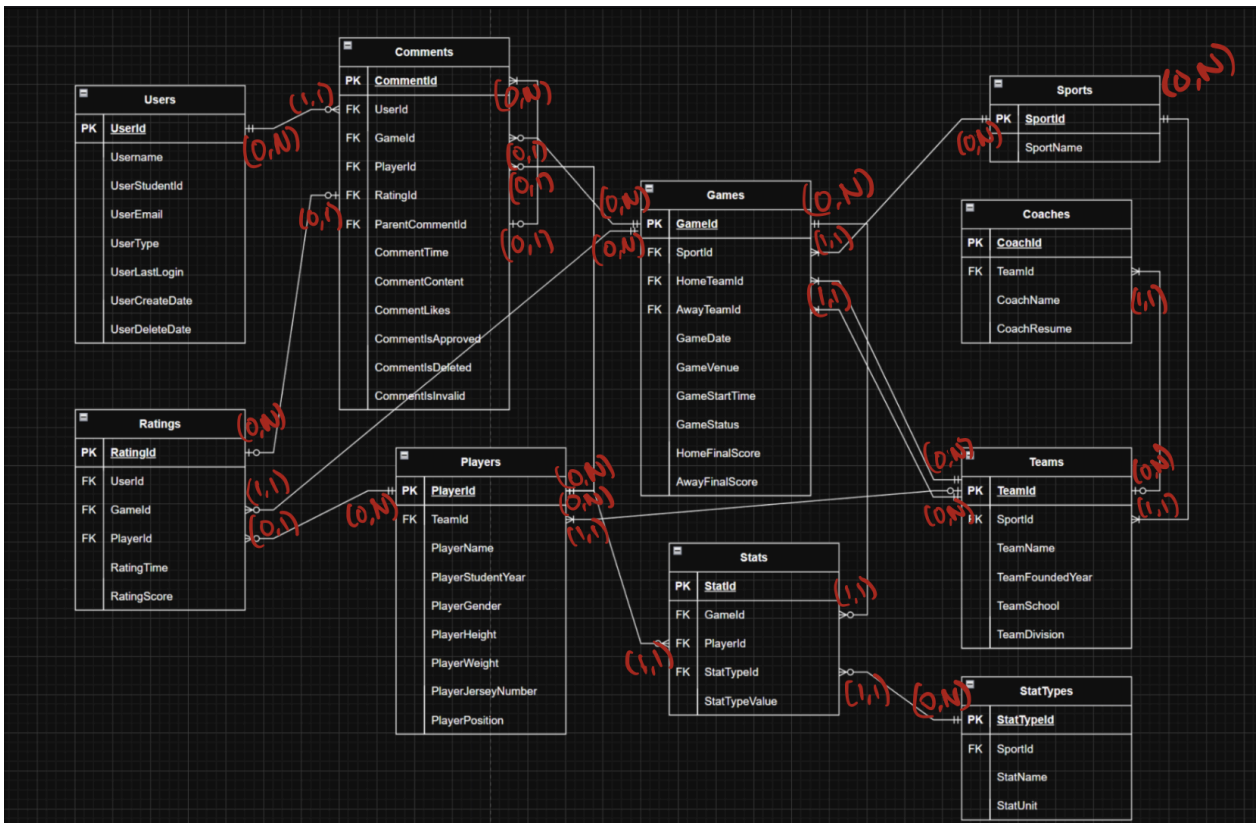
1. Sports → Teams
    - Sports to Teams: (0,N) — one sport can have many or no teams. ○ Teams to Sports: (1,1) — each team belongs to exactly one sport. ○ FK: Teams.SportId
2. Teams → Players
    - Teams to Players: (0,N) — a team can have many or no players. ○ Players to Teams: (1,1) — each player must belong to exactly one team. ○

FK: Players.TeamId

3. Teams → Coaches
    - Teams to Coaches: (0,N) — a team can have zero or many coaches.
    - Coaches to Teams: (1,1) — each coach belongs to exactly one team.
    - FK: Coaches.TeamId
4. Sports → Games
    - Sports to Games: (0,N) — a sport can include many or no games. ○ Games to Sports: (1,1) — each game belongs to exactly one sport. ○ FK: Games.SportId
5. Teams (Home) → Games
    - Teams to Games (as Home): (0,N) — a team can be the home team in many or no games.
    - Games to Teams (as Home): (1,1) — each game must have one home team.
    - FK: Games.HomeTeamId
6. Teams (Away) → Games
    - Teams to Games (as Away): (0,N) — a team can be the away team in many or no games.
    - Games to Teams (as Away): (1,1) — each game must have one away team.
    - FK: Games.AwayTeamId
7. Games → Stats
    - Games to Stats: (0,N) — a game can have many stat entries or none.
    - Stats to Games: (1,1) — each stat record must link to one game. ○ FK: Stats.GameId
8. Players → Stats
    - Players to Stats: (0,N) — a player can have many or no stat records.
    - Stats to Players: (1,1) — each stat record links to one player.
    - FK: Stats.PlayerId
9. StatTypes → Stats
    - StatTypes to Stats: (0,N) — a stat type can be used in many stat records.
    - Stats to StatTypes: (1,1) — each stat record uses one stat type.
    - FK: Stats.StatTypeId
10. Sports → StatTypes
    - Sports to StatTypes: (0,N) — a sport can define multiple stat types.
    - StatTypes to Sports: (1,1) — each stat type belongs to one sport. ○ FK: StatTypes.SportId
11. Users → Ratings
    - Users to Ratings: (0,N) — a user can leave multiple ratings.
    - Ratings to Users: (1,1) — each rating belongs to exactly one user.
    - FK: Ratings.UserId
12. Games → Ratings
    - Games to Ratings: (0,N) — a game can receive multiple ratings or none.
    - Ratings to Games: (0,1) — a rating may refer to one game or none. ○ FK: Ratings.GameId (nullable)
13. Players → Ratings

- Players to Ratings: (0,N) — a player can have many ratings or none.
- Ratings to Players: (0,1) — a rating may refer to a player or none.
- FK: Ratings.PlayerId (nullable)

14. Users → Comments
    - Users to Comments: (0,N) — a user can post multiple comments.
    - Comments to Users: (1,1) — each comment belongs to one user.
    - FK: Comments.UserId
15. Games → Comments (optional)
    - Games to Comments: (0,N) — a game can have multiple comments. ○ Comments to Games: (0,1) — a comment may refer to one game or none. ○ FK: Comments.GameId (nullable)
16. Players → Comments (optional)
    - Players to Comments: (0,N) — a player can have multiple comments. ○ Comments to Players: (0,1) — a comment may refer to one player or none. ○ FK: Comments.PlayerId (nullable)
17. Comments (Parent–Child)
    - Parent Comment to Child Comments: (0,N) — a comment can have multiple replies.
    - Child Comment to Parent Comment: (0,1) — a comment may reply to one parent or be top-level.
    - FK: Comments.ParentCommentId (nullable)



5. Logical Design

Excel file of data dictionary table is attached.

Business Rules:

Rule 1 — Rating score must be 1–5

- Type: Attribute-oriented
- Rationale: Standardizes feedback and prevents out-of-range values.
- Statement: Ratings.RatingScore must be an integer in [1,5].
- Enforcement: DB CHECK (RatingScore BETWEEN 1 AND 5) + UI validation.
- Affected: Ratings.RatingScore.
- Example: 4 → allowed; 7 → rejected.

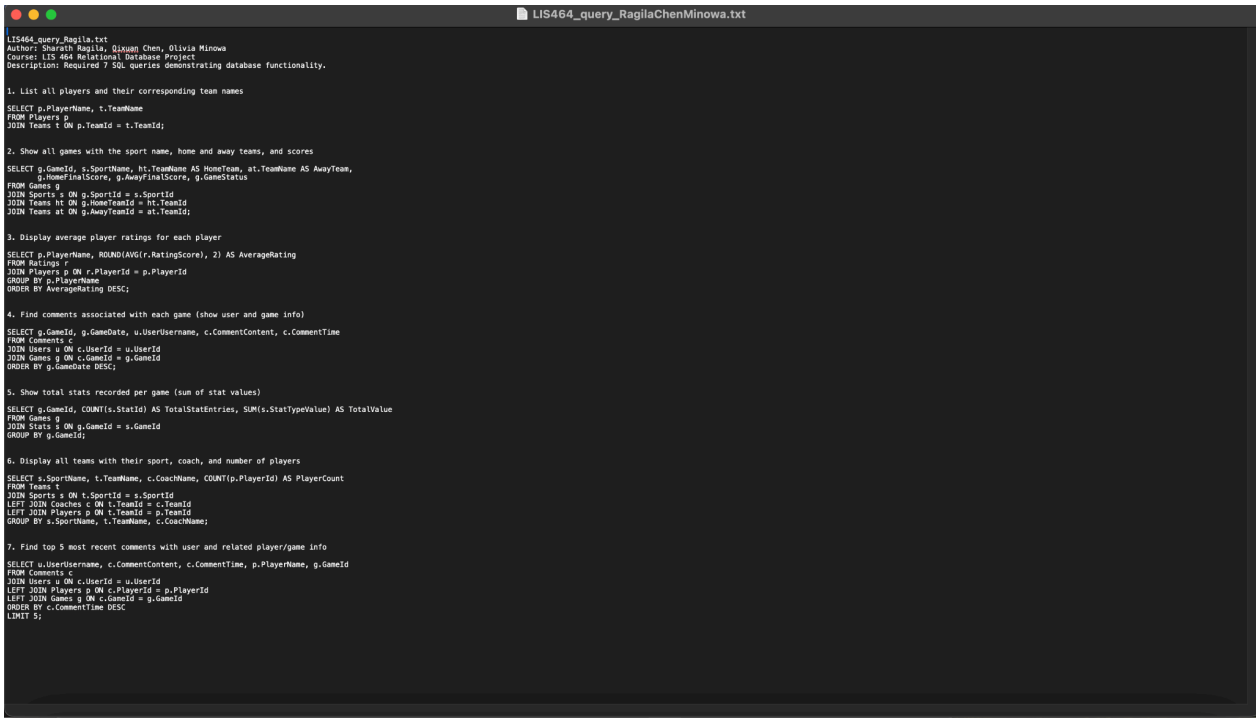
Rule 2 — Every Team belongs to exactly one Sport

- Type: Relationship-oriented
- Rationale: Ensures teams are sport-scoped; avoids orphan teams.
- Statement: Each Teams.TeamId must reference one valid Sports.SportId.
- Enforcement: FK Teams.SportId → Sports.SportId (NOT NULL, ON DELETE RESTRICT).
- Affected: Teams, Sports.
- Example: “UW Badgers M Basketball” cannot exist without SportId for “Basketball”.

Rule 3 — Each Player belongs to exactly one Team

- Type: Relationship-oriented
- Rationale: Maintains roster integrity for stats and schedules.
- Statement: Each Players.PlayerId must reference one valid Teams.TeamId.
- Enforcement: FK Players.TeamId → Teams.TeamId (NOT NULL, ON DELETE CASCADE).
- Affected: Players, Teams.
- Example: Inserting a player without a valid TeamId is rejected.

6. Queries



All queries located in the .txt file are attached.

2. SQL Implementation