Attributes:

user_id

username

email

password

roster_id

date_created

player_id

first_name

last_name

position

height

weight

team_id

draft_year

team_name

city

arena

championships_won

year_established

game_id

date

location

home_team_id

away_team_id

season

wins

losses

playoff_appearances

salary

points

assists

rebounds

steals

blocks

turnovers

Functional Dependencies

Users

Minimal Basis

`user_id` → `email`

`email` → `username`

`username` → `user_id`

`user_id` → `password`

Don’t need these FD’s for minimal basis

user_id → username

username → email

username → password

email → user_id

email → password

// User_id, username, and email should all be unique and CANNOT be duplicated. Password however can be the same between these unique values.

Players

Minimal Basis

player_id → first_name

player_id → last_name

player_id → position

player_id → height

player_id → weight

player_id → team_id

player_id → draft_year

// Primary Key covers all other attributes

Teams

Minimal Basis

team_id → team_name

team_id → city

team_id → arena

team_id → championships_won

team_id → year_established

team_id → wins

team_id → losses

team_id → playoff_appearances

// Primary Key covers all other attributes

Games

Minimal Basis

game_id → date

game_id → location

game_id → home_team_id

game_id → away_team_id

// Primary Key covers all other attributes

PlayerStats

Minimal Basis

(player_id, game_id) → points

(player_id, game_id) → assists

(player_id, game_id) → rebounds

(player_id, game_id) → steals

(player_id, game_id) → blocks

(player_id, game_id) → turnovers

// Primary Key covers all other attributes

Roster

Minimal Basis

roster_id → user_id

roster_id → date_created

// Primary Key covers all other attributes

Logical Design:

Users(

user_id: INT [PK],

email: VARCHAR(255),

username: VARCHAR(255),

password: VARCHAR(255)

)

Rosters(

roster_id: INT [PK],

user_id: INT [FK to User.user_id],

date_created: DATETIME

)

RosterPlayers(

player_id: INT [PK] [FK to Player.player_id]

roster_id: INT [PK] [FK to Roster.roster_id],

)

Players(

player_id: INT [PK],

first_name: VARCHAR(255),

last_name: VARCHAR(255),

position: VARCHAR(255),

height: DECIMAL,

weight: DECIMAL,

team_id: INT [FK to Team.team_id],

draft_year: INT,

salary: DECIMAL(15, 2)

)

Teams(

team_id: INT [PK],

team_name: VARCHAR(255),

city: VARCHAR(255),

arena: VARCHAR(255),

championships_won: INT,

year_established: INT

wins: INT,

losses: INT,

playoff_appearances: INT

)

Games(

game_id: INTEGER [PK],

date: DATETIME,

location: VARCHAR,

home_team_id: INTEGER [FK to Team.team_id],

away_team_id: INTEGER [FK to Team.team_id]

)

PlayerStats(

player_id: INT [PK] [FK to Player.player_id],

game_id: INT [PK] [FK to Game.game_id],

points: INT,

assists: INT,

rebounds: INT,

steals: INT,

blocks: INT,

turnovers: INT

)

Entities and Assumptions

Users

- **Attributes:**
  - user_id (Primary Key)
  - username
  - email
  - password
- **Assumptions:**
  - Represents an individual who interacts with the system.
  - Users can create and manage multiple rosters.
  - Each user has unique credentials (username and email are unique) for authentication.
- **Justification for Entity:**
  - Modeled as an entity because users have multiple attributes and participate in relationships with other entities.
  - Managing user data separately allows for secure authentication and personalized experiences.

Rosters

- **Attributes:**
  - roster_id (Primary Key)
  - user_id (Foreign Key to User)
  - date_created
- **Assumptions:**
  - Represents a collection of players selected by a user.
  - Each roster is created and owned by one user.
  - Rosters can be updated or deleted by the owning user.
- **Justification for Entity:**
  - Modeled as an entity to encapsulate the roster's attributes and relationships.
  - Allows tracking of when rosters were created and by whom.

RosterPlayers

- **Attributes:**
  - roster_id (Foreign Key to Roster)
  - player_id (Foreign Key to Player)
  - **Primary Key:** (roster_id, player_id)
- **Assumptions:**

    - Created to represent the many-to-many relationship between Rosters and Players.
    - Each record associates a player with a roster.
- **Justification for Entity:**
    - Necessary to implement the many-to-many relationship.
    - Allows querying which players are in which rosters and vice versa.

Players

- **Attributes:**
    - player_id (Primary Key)
    - first_name
    - last_name
    - position
    - height
    - weight
    - team_id (Foreign Key to Team)
    - draft_year
    - salary
- **Assumptions:**
    - Represents an individual athlete in the league.
    - Each player belongs to one team at any given time.
    - Players have personal and career-related attributes.
- **Justification for Entity:**
    - Modeled as an entity due to its numerous attributes and participation in multiple relationships (e.g., with Team, Roster, PlayerStats).

Teams

- **Attributes:**
    - team_id (Primary Key)
    - team_name
    - city
    - arena
    - championships_won
    - Year_established
    - wins
    - losses
    - playoff_appearances
- **Assumptions:**
    - Represents a professional sports team.

    - Teams have multiple players and participate in games.
    - Historical data like championships won is tracked.
- **Justification for Entity:**
    - Modeled as an entity to store team-specific information and to relate with other entities like Player and Game.

Games

- **Attributes:**
    - game_id (Primary Key)
    - date
    - location
    - home_team_id (Foreign Key to Team)
    - away_team_id (Foreign Key to Team)
- **Assumptions:**
    - Represents a scheduled match between two teams.
    - Each game occurs at a specific date and location.
    - Distinguishes between home and away teams.
- **Justification for Entity:**
    - Modeled as an entity to capture game details and facilitate relationships with PlayerStats and Team.

PlayerStats

- **Attributes:**
    - player_id (Foreign Key to Player)
    - game_id (Foreign Key to Game)
    - points
    - assists
    - rebounds
    - steals
    - blocks
    - turnovers
    - **Primary Key:** (player_id, game_id)
- **Assumptions:**
    - Represents the performance statistics of a player in a specific game.
    - Multiple statistical attributes are recorded per game.
- **Justification for Entity:**
    - Modeled as an entity to track performance over time and to support detailed statistical analysis.

Cardinality of Relationships

Users to Rosters

Type: One-to-Many

Explanation:

- **One** User can create **many** Rosters.
- Each Roster is created by **one** User.

Assumptions:

- Users may manage multiple rosters for different purposes or leagues.
- Rosters cannot exist without an associated user.

Rosters to RosterPlayers

Type: Many-to-One

Explanation:

- **Many** Rosters can include **many** Players.
- **Many** Players can be part of **many** Rosters.

Assumptions:

- Players can be selected by multiple users for their rosters.
- Rosters are customizable collections of players.

RosterPlayers to Players

Type: One to One

Explanation:

- **One** RosterPlayer is **One** Player.
- **Vice versa** holds true.

Assumptions:

- Players can be selected by multiple users for their rosters.
- Rosters are customizable collections of players.

Players to Teams

Type: Many-to-One

Explanation:

- **Many** Players belong to **one** Team.
- Each Player is associated with **one** Team at a time.

Assumptions:

- Players cannot play for multiple teams at once.
- Team rosters consist of multiple players.

Teams to Games

Type: Two-to-Many [For both Home and Away teams]

Explanation:

- **One** Team can be the home team in **many** Games.
- **One** Team can be the away team in **many** Games.
- Each Game has **one** Home Team and **one** Away Team.

Assumptions:

- Teams participate in multiple games.
- Each game involves exactly two teams.

Players to PlayerStats

Type: One-to-One

Explanation:

- **One** Player can have **One** PlayerStats records.
- **Vice versa** holds true.

Assumptions:

- Players accumulate statistics over multiple games.

- Each statistical record corresponds to a specific game.

Games to PlayerStats

Type: One-to-Many

Explanation:

- **One** Game can have **many** PlayerStats records.
- Each PlayerStats record is for **one** Game.

Assumptions:

- Multiple players participate in a game, each generating statistics.
- Allows aggregation of player performances per game.