CSE202 | DBMS

OUTSIDE EDGE

PROJECT REPORT

Group 63

Ramesh (2018081)

Suchet (2018105)

Aanya (2018208)

Adwit (2018276)

Ankit (2018381)

Description for the Mini-World:

The yearly IPL tournament is a professional Twenty20 cricket tournament where eight(sometimes a few more) battle it out to determine the ultimate dream team of cricketers from around the world.

However, what one fails to notice is the abundance of data and the need to store and manage it efficiently, there are over 60 matches spanned over two months, multiple days having two games a day.

The proposed Database Management system aims to handle the multitude of data like the details of the Teams, the players playing for the team, the owner of the team, their home ground, the officials of the matches, the fixtures between teams, and the game by match report for the entire duration of the Tournament. Besides this, the solution provides the admins of the IPL board and the chairman with a complete overview of the Tournament managing the salaries of all that are part of this Tournament, be it the officials, the players, the coaches or the respected cricket ground boards. The Database manages and updates the data. In it, as the Tournament progresses and adds to the record of the performances by individual players and the teams as a collective unit.

Stakeholders for the Mini-World and Assumptions for each:

1. **Teams**

   The Teams are an integral part of the System as the entire Tournament revolves around them.
2. **Managers**

   Most Teams have multiple managers and coaches, and their performance and wages are needed to be monitored, additionally each of these officials need to be able to view the details about the performance of each player.
3. **Owners**

   Most Teams have multiple owners with varying stakes in the respective teams, each of these owners need to be able to view the details about the performance of each player, the sponsors for their teams etc.
4. **Player Association/Board**

   The Player Association keeps track of the players, their record with IPL board (previous teams, current team, runs, etc.)
5. **Cricket Ground Committee**

   The IPL board also has a Cricket ground committee that manages stadiums, the hoardings for advertisements, and the availability of a stadium for a given match, whilst analyzing the sale of tickets for each stadium
6. **Players**

   A given player should be able to view his profile, and get the details of the Owners, coaches, and should be able to see his past track record.

7. **Auction Committee**

   The Auction Committee is also a part of the IPL board and is responsible for conducting the Auction for players for each season.

8. **Managing Board and Chairman**

   The Managing Board and Chairman have all the privileges and look after the entire function of the System for its smooth operation.

9. **Umpires**

   Umpires/ Officials report directly to the board and should be able to view their profiles, and the matches assigned to them.

10. **Sponsors**

    All teams have sponsors that naturally want their teams to perform exceptionally, and performance of an organization determines the amount of money(stake) a given company would provide as a sponsor

11. **Broadcasters**

    The broadcasters are an integral part of the System as different channels have different rights for broadcasting the matches and running advertisements for different teams.

Queries for each Stakeholder :

Team :

- Find the Teams who won maximum matches in the Tournament.
- Total points of a particular team (say Team “X”)
- Find captain of “X” team
- List of teams in Tournament whose matches got a draw
- Name of the team whose player won Orange/Purple cap.

Managers :

- Find the details of coach “X.”
- Find the details of coach of “X” Team
- Add/ Update Team Details
- Number of matches won by the team of “X” coach

Owners :

- Find a list of owners of team number “X.”
- Name of the owner who bought the most expensive player during IPL auctions
- Total money spent by Owner “x” to build his IPL team (Sum of money paid to buy all players of his team)
- Details of Team owned by Owner “O.”

Cricket Ground Committee:

- Find the name of the home ground of the team “X.”
- Find the name of Ground where Team “X” vs. Team” Y” match will be played

- The number of games played in Ground “X.”
- Find Location where Ground “X” is located
- Add new Ground
- Change/ Add new Homeground for team X to Ground Y

Player Association :

- The player who Won the orange/purple cap.
- List of Players of Team “X”
- The player who took maximum Wickets in the Tournament
- The player who scored maximum runs in the Tournament.
- Most Expensive Players
- Add Players to the Tournament
- Update Details of The Player

Indian Players/ Foreign Players :

- The total number of Indian/ Foreign players in Team “X.”
- Indian/ Foreign Player who scored Maximum Runs
- Most Expensive Indian/Foreign Player
- Find the contact details of the owner of the team for which player “X” plays
- Find the details of the sponsor of the team for which player “X” plays

Umpires:

- View his profile or someone else’s.
- List of umpires for a match no X
- Schedule of matches that he is the umpire of.

Sponsors:

- Performance of the team of the sponsored team.
- Performance of individual players of the sponsored team.
- The number of players on the team.

Broadcasters:

- Info of the matches in the Tournament.
- Info of the commentators in a particular match.
- List of commentators in the Tournament.

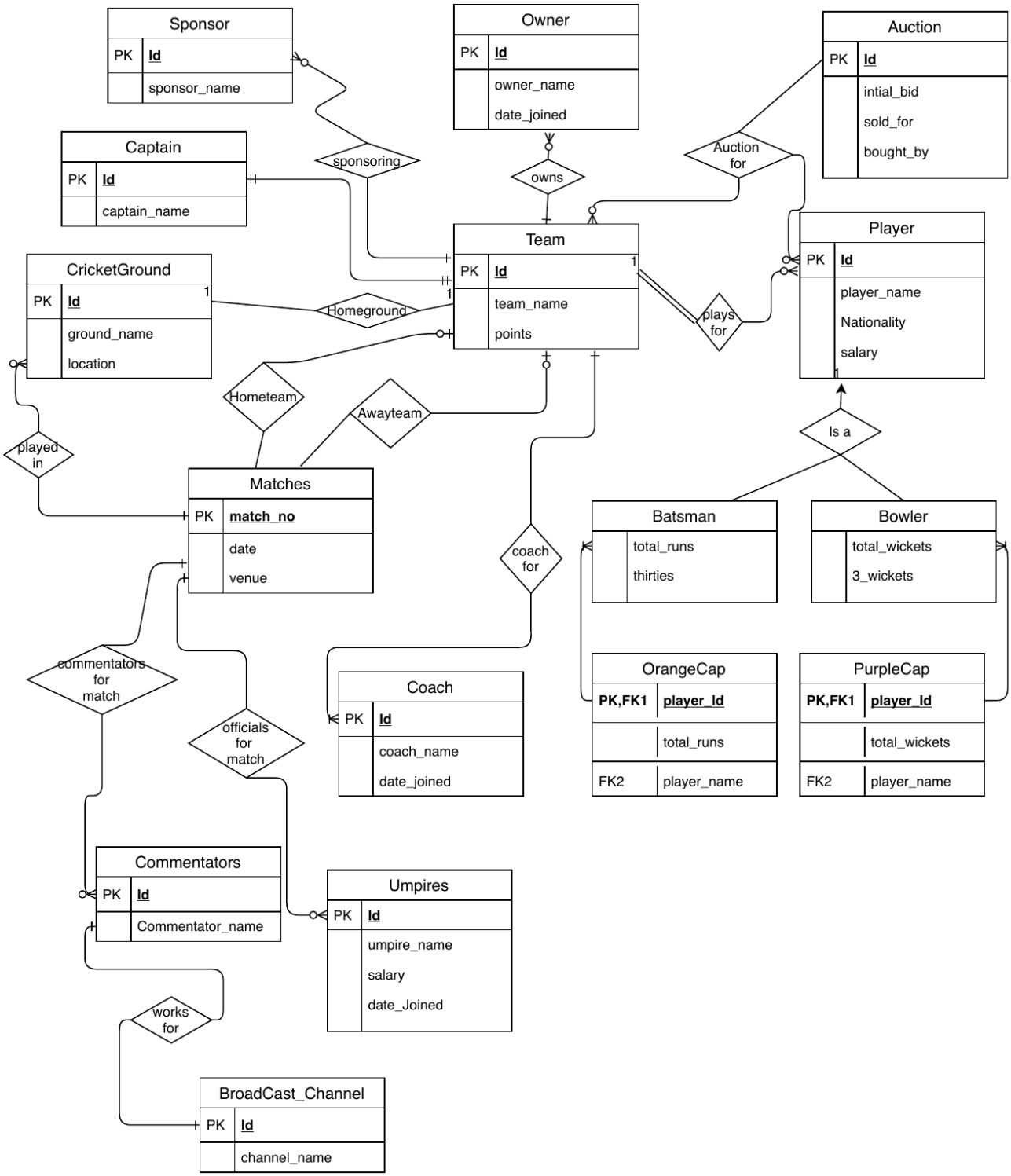
Auction committee:

- The base price of a player.
- Most expensive overseas player.
- Details of unsold players.
- Make the sale of Player X

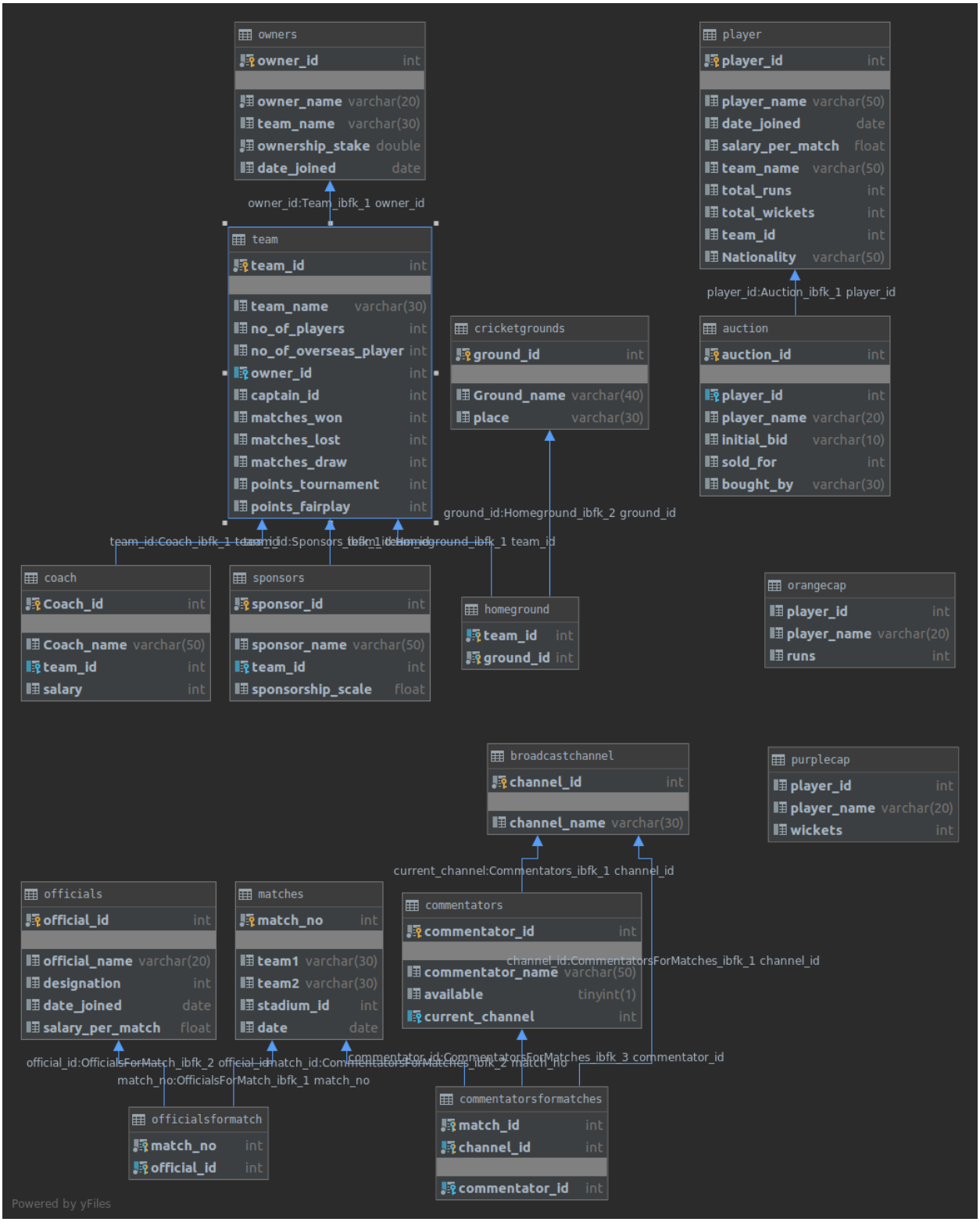
Relation Schema Description:

1. **MATCHES** (match_no int, team1 varchar, team2 varchar, stadium_id int, DATE date)
2. **OfficialsForMatch** (match_no int, official_id int)
3. **AUCTION** (auction_id int, player_id int **foreign key from Players [CASCADE]**, player_name int, initial_bid real, , sold_for real, bought_by varchar)
4. **UMPIRES** (official_id int, official_name varchar, designation int, date_joined date, salary_per_match float)
5. **PLAYERS** (player_id int, player_name varchar date_joined date salary_per_match float, team_id int, team_name varchar, total_runs int , total_wickets int, nationality varchar)
6. **TEAMS** (team_id int, team_name varchar, no_of_players int, owner_id int **foreign key from Owners [CASCADE]**, captain_id int, matches_won int, matches_lost int, matches_draw int, points_tournament int, points_fairplay int)
7. **OWNERS** (owner_id int, owner_name varchar, Team_id int,Ownership_stake real, date_joined date)
8. **MANAGERS** (Coach_id int, team_id int **foreign key from Teams [CASCADE]** coach_name varchar, salary real)
9. **GROUNDS** (ground_id int, Ground_name varchar, place varchar)
10. **HOMEGROUND** (team_id int **foreign key from TEAMS**, ground_id int **foreign key from GROUNDS**)
11. **SPONSORS** (Sponsor_id int, sponsor_name, team_id int **foreign key from Teams [RESTRICT]**, sponsorship scale float)
12. **BROADCASTERS** (Channel_id int, channel name varchar)
13. **CommentatorForMatch** (Match_id int **foreign key from MATCHES**, Channel_id int **foreign key from BROADCASTERS**, commentator_id int)
14. **COMMENTATORS** (commentator_id int, commentator_name varchar(50), available bool, current_channel int **foreign key from Broadcast_Channel[CASCADE]**)

ER Model for the Mini World:



Relation Schema:



Indices on Attributes:

*****All the primary keys for each relation are auto-incremented, and thus one does not need to enter the details again and again*****

All primary keys for all relations are, by default, represented as indices in SQL. Thus all primary keys for each of the tables are already being used as indices.

- Match_no for **Match**
- Auction_id for **Auction**
- Official_id for **Official**
- Player_id for **Player**
- Team_id for **Team**
- Owner_id for **Owner**
- Coach_id for **Manager**
- Sponsor_id for **Sponsor**
- Channel_id for **Broadcasters**
- Commentator_id for **Commentators**

Apart from the usual primary Key-based indices, we have created the following indices (B+ tree-based indices):

- Sold_for in **Auction**: Majority of the queries for the auction committee revolve around analyzing which players were sold for what price and since the Database for Auction is decidedly huge (on a real scale) creating an index based on selling price would benefit as one can get a list of the player with selling price greater than, less than or in a range fairly quickly. **(index1)**
- Place in **Cricket Grounds:** The majority of the time, we need to quickly find all the stadia in a particular region like North or South West, so for doing that, we can create an index based on the location of the cricket ground. **(index2)**
- Stadium_id in Matches **(index3),** To quickly find out which all matches are played at a given stadium more efficiently instead of traversing through the entire relation.
- Bought_by in Auction **(index4)** To quickly find out about all purchases made by a given team.
- Team_name in Players **(index5)**, For Grouping the Players of a team together
- Team_id in Sponsors **(index6),** For Grouping the Sponsors of a team together

Types of SQL commands used in the System:

1. **DDL – Data Definition Language**
    1. CREATE
    2. DROP
2. **DQL – Data Query Language**
    1. SELECT
3. **DML – Data Manipulation Language**
    1. INSERT
    2. UPDATE
    3. DELETE
4. **DCL – Data Control Language**
    1. GRANT
    2. REVOKE

Privileges and Roles:

We have defined a role for each of the stakeholders and have specific rights for each role after that,
The Roles and Privileges are as follows:

1. **Manager**

   For any team, the role of a Manager is to view the performance and other statistics related to the team, and possibly viewing the performance of other teams as well, release and update stats for each player, etc. thus they only have permission to **SELECT AND UPDATE on TEAMS, SELECT on MANAGERS**
2. **Owners**

   Most Teams have multiple owners with a different stake in the respective teams. Each of these owners needs to be able to view the details about the performance of each player, the sponsors for their teams, etc. Thus, they have permissions to **SELECT ON SPONSOR and TEAMS and Owners**
3. **Player Board**

   Player Association's responsibility is to maintain a record for each player, update their records and maintain consistency thus they have permission to **ALL ON PLAYER**
4. **Players**

   A given player should be able to view his profile, and get the details of the Owners, coaches, and should be able to see his past track record. They have permission only to **SELECT ON PLAYER, TEAMS**
5. **Cricket Ground Committee**

   The IPL board also has a Cricket ground committee that manages stadiums, and they have complete right to update and/or delete details about the stadia, permission to **ALL ON GROUND, and HOMEGROUND**
6. **Auction Committee**

The Auction Committee is also a part of the IPL board and is responsible for conducting the Auction for players for each season. This involves inserting, updating and/or deleting records for each sale made during the Auction, permission to **ALL ON AUCTION and PLAYER, and TEAMS**

7. **Managing Board and Chairman**
   The Managing Board has the complete authority over all the tables presented in the Mini-World and thus have permission to **ALL** on every relation. They have the power to **GRANT** permission to the various stakeholders.
8. **Umpires**
   Umpires/ Officials should be able to view their profiles, and the matches assigned to them. **SELECT on UMPIRES**
9. **Sponsors**
   All teams have sponsors that naturally want their teams to perform exceptionally, and performance of a team determines the amount of money(stake) a given company would provide as a sponsor **SELECT on SPONSORS and TEAMS**
10. **Broadcasters**
    The broadcasters are an integral part of the System as different channels have different rights for broadcasting the matches and running advertisements for different teams. **SELECT on Commentators, CommentatorForMatch and BROADCASTERS**

Normalization:

The given Database is represented in the Boyce Codd Normal Form (BCNF) as all functional dependencies have been removed, and there are no transitive dependencies as well. In each of the tables, the Left-hand side for every functional dependency is the Primary Key for that relation, thus satisfying the criteria for BCNF.

1 NF:
All Attributes are atomic thus DB is in 1 NF Form

2 NF:
Since In Every Relation the primary key is a single attribute, there cannot be any partial dependency, (There are two relations where this is not the case, however in those relations there are no partial dependencies as well)

3 NF:
There are no partial dependencies in any of the relations, several new relations where created to remove the transitive dependencies such as OfficialsForMatch. CommentatorForMatch etc.

BCNF:
Since each relation satisfies 3NF and also in every functional dependency the LHS is the primary key itself, the relation satisfies BCNF as wells

Triggers:

For insertion in the Database, several constraints have been imposed either implicitly (by referential integrity constraints) or otherwise like Adding data for a player does not take in values for the team for which he plays, which can only be set once the player has been auctioned to some team. A similar approach has been implemented for teams and owners, i.e., a team cannot exist without an owner, and each team can have exactly one owner at any given point of time.

Keeping the data consistent throughout, we added a handful of triggers that take care of all assumptions of the Mini World and also make the Database consistent for use.

1. addPlayer
2. deletePlayer
3. auction_change_team
4. auction_add_player
5. deleteTeam
6. addTeam
7. updateTeam

The design choice behind adding triggers instead of augmenting the Database to store the information in different tables was that the latter would involve taking a handful of JOINS for every operation for which a trigger was created. Since a JOIN operation is quite expensive, we decided to go with the earlier approach, and having created indices on team_name and team_id majority of these operations are quite efficient and also greatly reduces the storage requirements. Further, the choice was optimal since the frequency of addition of Players, teams, and sale of Players in Auction is quite less as compared to the frequency with which this data is queried so the System can afford a bit higher cost in terms of addition/ modification to the records than in terms of queries (which in turn would be higher if one uses multiple tables to store the data).

Relational Queries:

1) Total points of a particular team (say Team "X")

$\pi_{points\_tournament}(\sigma_{team\_name="KKR"}(\text{Team}))$

2) List of teams in tournament whose matches got a draw

$\pi_{team\_name}(\sigma_{matches\_draw <> 0}(\text{Team}))$

3) The player who Won the orange/purple cap

$\pi_{player\_name}(\text{OrangeCap}) \cup \pi_{player\_name}(\text{PlayerCap})$

4) List of Players of Team "X"

$\pi_{player\_name}(\sigma_{team\_name="KKR"}(\text{Player}))$

5) Find a list of owners of team number x,y,z

$\pi_{owner\_name}(\sigma_{team\_name="KKR" \wedge team\_name="RR"}(\text{Owners}))$

6) Owner of "X" team

$\pi_{owner\_name}(\sigma_{team\_name="KKR"}(\text{Owners}))$

7) Info of a player 'X'

$\sigma_{player\_name='X'}(\text{Players})$

8) Performance of players in the team sponsored by 'X'.

$\pi_{total\_runs,total\_wickets}(\sigma_{p.team\_id=s.team\_id \cap s.sponsor\_name='X'}(\text{Sponsor} \times \text{Players}))$

9) View profile of umpire 'X'

$\sigma_{designation='2' \cap official\_name='X'}(\text{Officials})$

10) Names of unsold players

$\pi_{player\_name}(\text{Players}) - \pi_{player\_name}(\text{Players} \bowtie \text{Auction})$

11) The total number of Indian players in Team X.

$\pi_{(no\_of\_players - no\_of\_overseas\_players)}(\sigma_{team\_name='X'}(\text{Team}))$

12) List of commentators in the tournament.

$\pi_{commentator\_name}(\text{Commentators})$

Embedded SQL queries:

Our project is implemented in python and uses the mysql.connector library for embedding SQL and python, the Queries for SQL remain the same and are passed to the function committing the changes.

Following is the code for the function:

```
def execute_read_query(connection, query):
    cursor = connection.cursor()
    result = None
    try:
        cursor.execute(query)
        result = cursor.fetchall()
        return result
    except Error as e:
        print(f"The error '{e}' occurred")
```

Rest of the Embedded SQL queries are present in the file named embedded.py submitted along with the final submission, while a few have been mentioned below:

```
def get_rollup(connection):
        cursor = connection.cursor(dictionary=True)
        result = None
        try:
                cursor.execute("SELECT Nationality, team_name,SUM(total_runs)
'Runs' ,SUM(total_wickets) 'Wicket' FROM player GROUP BY
Nationality,team_name WITH ROLLUP")
                result = cursor.fetchall()
                return result
        except Error as e:
                print(f"The error '{e}' occurred")
```

```
def get_highest_bid_team(connection):
        cursor = connection.cursor(dictionary=True)
        result = None
        try:
                cursor.execute("SELECT bought_by 'Team', player_name,sold_for,
RANK() OVER (PARTITION BY bought_by ORDER BY sold_for DESC )
'Highest_Purchase' FROM auction")
                result = cursor.fetchall()
                return result
        except Error as e:
                print(f"The error '{e}' occurred")
```

```
def get_avg_stats_team(connection):
        cursor = connection.cursor(dictionary=True)
        result = None
        try:
                cursor.execute("SELECT player_name, team_name, total_runs,
AVG(total_runs) OVER(PARTITION BY team_name) as average_runs_of_team,
total_wickets, AVG(total_wickets) OVER(PARTITION BY team_name) as
average_wickets_of_team from player")
```

```
                result = cursor.fetchall()
                return result
        except Error as e:
                print(f"The error '{e}' occurred")
```

```
def get_unsold_players(connection):
        cursor = connection.cursor(dictionary=True)
        result = None
        try:
                cursor.execute("SELECT player.* FROM player LEFT JOIN auction
ON (player.player_id = auction.player_id) WHERE auction.player_id IS NULL")
                result = cursor.fetchall()
                return result
        except Error as e:
                print(f"The error '{e}' occurred")
```

```
def get_base_price_for_players(connection, player_name):
        cursor = connection.cursor(dictionary=True)
        result = None
        try:
                cursor.execute("SELECT player_name,initial_bid, sold_for from
auction WHERE player_name=\"" + player_name + "\"")
                result = cursor.fetchall()
                return result
        except Error as e:
                print(f"The error '{e}' occurred")
```

```
def get_commentators_of_match(connection, match_no):
        cursor = connection.cursor(dictionary=True)
        result = None
        try:
                cursor.execute("SELECT commentator_name FROM (commentators
NATURAL JOIN commentatorsformatches CM) WHERE CM.match_id="+ str(match_no))
                result = cursor.fetchall()
                return result
        except Error as e:
                print(f"The error '{e}' occurred")
```

```
def get_teams_sponsored_by(connection, sponsor_name):
        cursor = connection.cursor(dictionary=True)
        result = None
        try:
                cursor.execute("select team_name from Team where(team_id) in
(select team_id from Sponsors where sponsor_name=\"" + sponsor_name + "\")")
                result = cursor.fetchall()
                return result
        except Error as e:
                print(f"The error '{e}' occurred")
```

```
def get_most_expensive_players(connection):
        cursor = connection.cursor(dictionary=True)
```

```
        result = None
        try:
                cursor.execute("SELECT player_name, sold_for, bought_by from
Auction ORDER BY sold_for DESC LIMIT 3;")
                result = cursor.fetchall()
                return result
        except Error as e:
                print(f"The error '{e}' occurred")
```

Advanced Aggregation Functions Used:

- Windowing
- Roll Up
- Rank

Events Created:

- upLoads- This event occurs every 1 minute (for simplicity purposes, otherwise should occur every 24 hours) that updates the PurpleCap and OrangeCap relations with the leading wicket takers and run scorers.

Innovative Features implemented:

- Fully Functioning Web App with profile selection based on whether the user is a player, Team representative, IPL Board member, etc.
- The App also gives a summary of the season so far, displaying the leaderboards.
- The App displays the performance of players by a scatter plot measured by the total runs and wickets team wise.
- Defining Roles for each stakeholder and correspondingly laying out the privileges for them.
- The App further strengthens the strictness with which the privileges assigned to each stakeholder are imposed by the use of additional Key given out to each representative for a stakeholder (For simplicity and ease of use the Key is simply the Role Name) based on which a given stakeholder can/ cannot access the portal for inserting/deleting or modifying the Database.
- Use of events that ensure periodic check of the database (for orange cap and purple cap) and backup the database thereby preventing any loss of data
- An additional backup script that backs up the database, **DATABASE RECOVERY**

Work Division:

1. **Ramesh:** Designed the ER Diagram.

2. **Suchet:** Designed the Relational schema of the Database and Normalized it to the BCNF, further defined the roles of the various stakeholders in the Mini World, and correspondingly charted out the Privileges they should have while interacting with the Database. Designed the Web App Portal for the Mini-World and helped in some of the embedded SQL queries for the Database

3. **Aanya:** Writing SQL queries for the stakeholders ,writing relational queries, helped in populating the database.

4. **Adwit:** Implementation of database on MySQL server, inserting triggers, writing embedded MySQL queries. Helped in normalizing, populating database, performing integrity checks, documentation and supervision of project workflow.

5. **Ankit:** Writing SQL queries for half of the stakeholders ,writing relational queries , Optimizing Database, helped in some designing part of our schema ,helped in populating the database