UCS310

Database Management System

Project

IPL Auction System

Submitted By:

CHIRAG JAWA	102203988
ARNAV GUPTA	102203749
APOORAV SINGH	102203503
EKASJOT SINGH	102203583

Submitted To:
Dr. Sanjeev Rao

Assistant Professor

INDEX

Introduction

Our project is an online simulation of an IPL (Indian Premier League) Auction System that is designed to enable the collection and management of player data. The IPL is a professional 20-20 cricket league in India, where players from around the world participate in a player auction to play for different franchise teams. In our project, we have modeled the system after this auction process, where teams bid on players to form their teams for the season.

The system we have created uses a database to store and manage player, team, and owner data. The player table includes a unique player_id, which is auto-incremented through the use of a Trigger. This ensures that each player in the system has a distinct identifier, which is essential for managing and tracking players throughout the auction process. Additionally, the player table includes several other attributes, such as the player's name, statistics, current value, the team they are playing for, and their designated player type (i.e., batsman, bowler, all-rounder). These attributes are essential for creating a detailed profile for each player in the system and for enabling teams to make informed decisions during the auction.

The team table is another crucial component of our system, as it stores information about the different teams in the league. This table includes a team_id, which is also auto-incremented to ensure that each team has a unique identifier. Additionally, the team table includes several other attributes, such as the team's name, balance, player count, and captain. The balance attribute represents the amount of money that the team has available for bidding on players during the auction. The player count attribute stores the number of players currently playing for the team, while the captain attribute identifies the captain of the team. These attributes are critical for enabling the system to track each team's progress throughout the season, and for providing teams with the information they need to make informed decisions during the auction.

To create a link between players and teams, we have implemented a foreign key constraint between the player and team tables. This constraint ensures that each player can only play for a single team, while a team can have multiple players playing for it. By linking players and teams in this way, we can track which players are currently playing for which teams, and we can also determine which teams have the most valuable players on their roster.

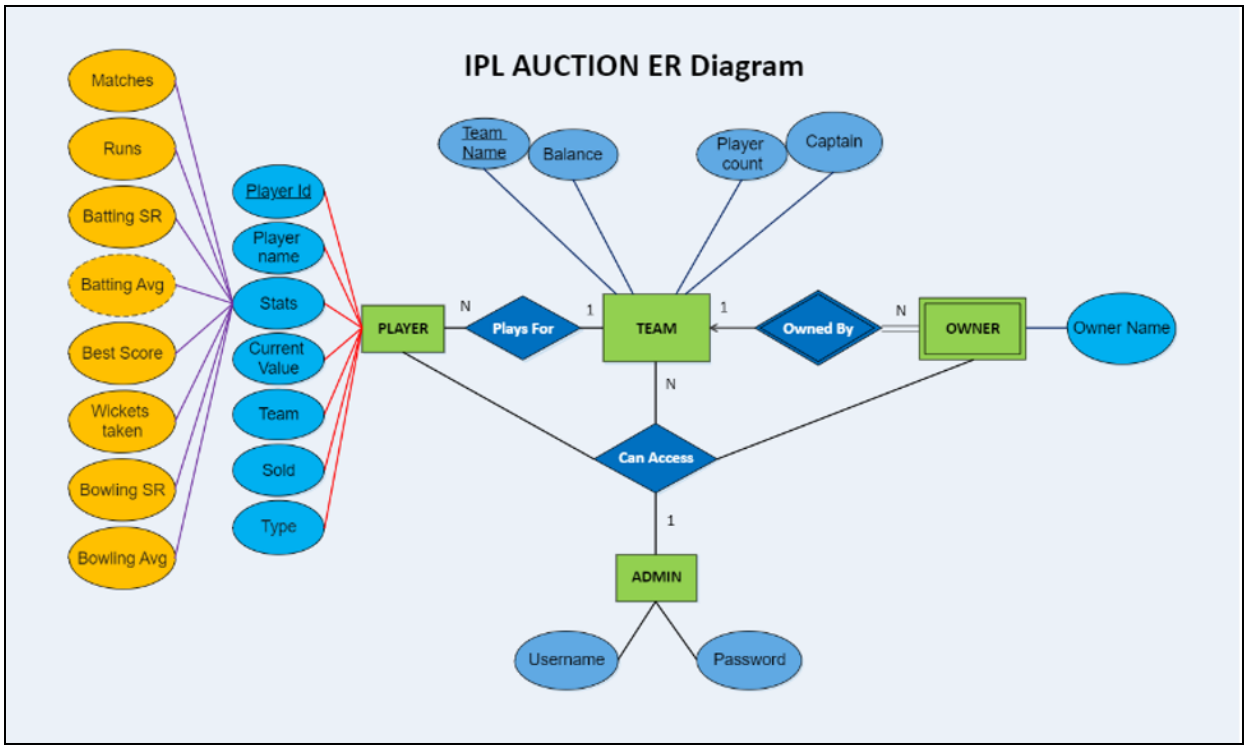
In addition to the player and team tables, we have also created an owners table to store information about the owners of each franchise team. The owners table includes the name of the owner, which is essential for identifying the person or organization that owns each franchise team. By creating an owners table, we can track which owners own which teams, and we can also determine which owners have the most valuable teams in the league.

Finally, to provide secure access to the system, we have created an admin table that stores the usernames and passwords of the system administrators. By requiring administrators to log in with a username and password, we can ensure that only authorized users have access to the system.

Overall, our project is a comprehensive simulation of an IPL Auction System, which is designed to enable the collection and management of player, team, and owner data. By creating a detailed database schema that includes player, team, and owner tables, we can track each aspect of the auction process, from the bidding on players to the formation of teams and the management of player rosters throughout the season. Moreover, by implementing foreign key constraints and other data integrity measures, we can ensure that the data in our system is accurate, consistent, and reliable.

Data Model

ER Diagram:



FD and Normalization

F1: {Player_id} ➔ {Player_id, Player_name}

In table Player

F2: {Player_name} ➔{Player_name, Matches, Runs, Bating_SR, Batting_AVG, Best, Wickets, Bowling_SR, Bowling_AVG, Current_value, Team, Status, Player_type}

In table Player_details

F3: {Team_name} ➔{Team_name, Balance, Player count, Captain}

In table Teams

F4: {Username} ➔ {Username, Password}

In table Admin

F5: {Team_name, Owner} ➔{Team_name, Owner}

In table Owners

FD and Normalization

Normalization is the process of minimizing redundancy from a relation or set of relations. Redundancy in relation may cause insertion, deletion, and update anomalies. So, it helps to minimize the redundancy in relations. Normal forms are used to eliminate or reduce redundancy in database tables.

1. First Normal Form:

If a relation contains multi-valued attribute, then it violates first normal form or a relation is in first normal form if it does not contain any multi-valued attribute. A relation is in first normal form if every attribute in that relation is singled valued attribute.

Team_name	Balance	Player_count	Captain	Owners
RCB	9500	1	Virat Kohli	Prathmesh Mishra
KKR	9500	1	Shreyas Iyer	Juhi Chawla, Shah Rukh Khan, Jay Mehta

The table teams has a multivalued attribute owners hence it is not in 1NF
So we reduce the table to 1NF with combination of Team_name and Owners as Key attribute.

Team_name	Balance	Player_count	Captain	Owners
RCB	9500	1	Virat Kohli	Prathmesh Mishra
KKR	9500	1	Shreyas Iyer	Juhi Chawla
KKR	9500	1	Shreyas Iyer	Shah Rukh Khan
KKR	9500	1	Shreyas Iyer	Jay Mehta

ALL other tables are in 1NF.

2. Second Normal Form:

To be in second normal form, a relation must be in first normal form and relation must not contain any partial dependency. A relation is in 2NF if it has No Partial Dependency, i.e., no non-prime attribute (attributes which are not part of any candidate key) is dependent on any proper subset of any candidate key of the table. Partial Dependency – If the proper subset of candidate key determines non-prime attribute, it is called partial dependency.

Team_name	Balance	Player_count	Captain	Owners
RCB	9500	1	Virat Kohli	Prathmesh Mishra
KKR	9500	1	Shreyas Iyer	Juhi Chawla
KKR	9500	1	Shreyas Iyer	Shah Rukh Khan
KKR	9500	1	Shreyas Iyer	Jay Mehta

Here ,

F: {Team_name, Owners} ➔ {Team_name, Balance, Player_count, Captain, Owners}

But,

F: {Owners} ➔ {Team_name, Balance, Player_count, Captain, Owners}

Hence, Partial dependency is present and table is not in 2NF.

So we now decompose the table into 2 tables.

Team_name	Balance	Player_count	Captain
RCB	9500	1	Virat Kohli
KKR	9500	1	Shreyas Iyer

Team_name	Owners
RCB	Prathmesh Mishra
KKR	Juhi Chawla
KKR	Shah Rukh Khan
KKR	Jay Mehta

All other tables are in 2NF.

3. Third Normal Form:

A relation will be in 3NF if it is in 2NF and does not contain any transitive partial dependency.

3NF is used to reduce the data duplication. It is also used to achieve data integrity.

If there is no transitive dependency for non-prime attributes, then the relation must be in third normal form.

A relation is in third normal form if it holds at least one of the following conditions for every non-trivial functional dependency X → Y.

X is a super key.

Y is a prime attribute, i.e., each element of Y is part of some candidate key.

Player_id	PLAYER_NAME	CURR_PRICE	Stats	TEAM_NAME	STATUS	PLAYER_TYPE
101	Virat Kohli	1500	1	RCB	Sold	Batsman
102	Shreyas Iyer	800	1	KKR	Sold	Batsman

Here,

F: {Player_id} ➔ {Player_name, curr_price, Stats, Team_name, Status, Player_type}
But,
F: {Player_name} ➔ {Curr_price, Stats, Team_name, Status, Player_type}
So, A non prime attribute is determining other non prime attributes
Hence , the table is not in 3NF.
So we decompose the above table into 2 tables:

PLAYER_NAME	CURR_PRICE	Stats	TEAM_NAME	STATUS	PLAYER_TYPE
Virat Kohli	1500	1	RCB	Sold	Batsman
Shreyas Iyer	800	1	KKR	Sold	Batsman

Player_id	PLAYER_NAME
101	Virat Kohli
102	Shreyas Iyer

Here the attribute creating transitive dependency is shifted to different table.
So table is now in 3NF.

All other tables are in 3NF

Table Creation

Admin

```
1  Create Table Admin(Username varchar(20) Primary Key,Password varchar(10) Not Null);
2  Describe Admin;
```

Table created.

TABLE ADMIN

Column	Null?	Type
USERNAME	NOT NULL	VARCHAR2(20)
PASSWORD	NOT NULL	VARCHAR2(10)

Player

```
1  Create Table Player(Player_id int Primary Key,Player_name Varchar(30) Not Null);
2  Describe Player;
```

Table created.

TABLE PLAYER

Column	Null?	Type
PLAYER_ID	NOT NULL	NUMBER
PLAYER_NAME	NOT NULL	VARCHAR2(30)

Player Details

```
create table player_details(Player_name Varchar(30) primary key,Current_price int, Matches int,Runs int,
    Batting_SR decimal(5,2), Batting_avg decimal(5,2),Best int,
    Wickets int,Bowling_SR decimal(4,2),Bowling_avg decimal(4,2),
    Team_name varchar(50) default Null,Status varchar(6) default 'Unsold');
Describe Player_details;
```

TABLE PLAYER_DETAILS

Column	Null?	Type
PLAYER_NAME	NOT NULL	VARCHAR2(30)
CURRENT_PRICE	-	NUMBER
MATCHES	-	NUMBER
RUNS	-	NUMBER
BATTING_SR	-	NUMBER(5,2)
BATTING_AVG	-	NUMBER(5,2)
BEST	-	NUMBER
WICKETS	-	NUMBER
BOWLING_SR	-	NUMBER(4,2)
BOWLING_AVG	-	NUMBER(4,2)
TEAM_NAME	-	VARCHAR2(50)
STATUS	-	VARCHAR2(6)

Team

```
1  create table team(Team_name Varchar(50) Primary Key, Balance int, Player_count int,Captain Varchar(30));
2  Describe team;
```

Table created.

TABLE TEAM

Column	Null?	Type
TEAM_NAME	NOT NULL	VARCHAR2(50)
BALANCE	-	NUMBER
PLAYER_COUNT	-	NUMBER
CAPTAIN	-	VARCHAR2(30)

Owners

```
1  create table owners(Team_name varchar(50) references team(team_name),Owner_name varchar(30),Primary key(team_name,owner_name));
2  describe owners;
```

TABLE OWNERS

Column	Null?	Type
TEAM_NAME	NOT NULL	VARCHAR2(50)
OWNER_NAME	NOT NULL	VARCHAR2(30)

Insertion of Values

```
14  insert into admin values('Chirag jawa','102203988');
15  insert into admin values('apoorav', '102203503');
16  insert into admin values('Arnav', '102203749');
17  insert into admin values('Ekasjot', '102203583');
18
19
```

```
1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.
```

```
21  insert into team values('GT', 9500, 1, 'Shubman Gill');
22  insert into team values('LSG', 9500, 1, 'KL Rahul');
23  insert into team values('SRH', 9500, 1, 'Pat Cummins');
24  insert into team values('DC', 9500, 1, 'Rishab Pant');
25  insert into team values('MI', 9500, 1, 'Hardik Sharma');
26  insert into team values('PBKS', 9500, 1, 'Shikhar Dhawan');
27  insert into team values('RR', 9500, 1, 'Sanju Samson');
28  insert into team values('CSK', 9500, 1, 'MS Dhoni');
29  insert into team values('KKR', 9500,1, 'Shreyas Iyer');
```

```
1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.
```

```
insert into player_details values('Virat Kohli', 1500, 230,6903,129.61,36.52,113,4,62.75,92, 'RCB', 'Sold', 'Batsman');
insert into player_details values('MS Dhoni', 1200, 240, 5037,135.77,39.35,84,0,0,0, 'CSK', 'Sold', 'Wicket keeper');
insert into player_details values('Shubhnam Gill', 1500, 112, 2078,145.82,29.69,91,51,21.92, 31.61, 'GT', 'Sold', 'All rounder');
insert into player_details values('Shreyas Iyer',800,97,2335,134.66, 27.8,87,9,16.78,21.56, 'KKR', 'Sold', 'Batsman');
insert into player_details values('Pat Cummins', 260, 25, 548,137.87, 38.12,68,1,78, 99, 'SRH', 'Sold', 'Batsman');
insert into player_details values('Shikhar Dhawan',825,210,6476,126.96,35.78,106,4, 12, 16.5, 'PBKS', 'Sold', 'Batsman');
insert into player_details values('KL Rahul', 1700,116,4151,134.55,47.17,132,0,0,0,'LSG', 'Sold', 'Batsman');
insert into player_details values('Sanju Samson', 1400,145,3707,136.69, 28.96,119,0,0,0, 'RR', 'Sold', 'Wicket keeper');
insert into player_details values('Rishabh pant ',625,168,6166,139.63,42.23,126,0,0,0, 'DC', 'Sold', 'Batsman');
insert into player_details values('Hardik Pandya', 1600, 233,6058,130.22, 30.29, 109, 15,22.6,30.2, 'MI', 'Sold', 'Batsman');
```

```
1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.
```

```
insert into player values (101, 'Virat Kohli');
insert into player values (102, 'MS Dhoni');
insert into player values (103, 'Hardik Pandya');
insert into player values (110, 'KL Rahul');
insert into player values (104, 'Shikhar Dhawan');
insert into player values (105, 'Hardik Pandya');
insert into player values (106, 'Sanju Samson');
insert into player values (107, 'Pat Cummins');
insert into player values (108, 'Shubhnam Gill');
```

```
1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.
```

```
39 insert into owners values('KKR', 'Shah Rukh Khan');
40 insert into owners values('KKR', 'Juhi Chawla');
41 insert into owners values('KKR', 'Jay Mehta');
42 insert into owners values('RR', 'Manoj Badale');
43 insert into owners values('RR', 'Lachlan Murdoch');
44 insert into owners values ('RR', 'Shane Warne');
45 insert into owners values('SRH', 'Kalanithi Maran');
46 insert into owners values('CSK', 'N Srinivasan');
47 insert into owners values('GT', 'Siddharth Patel');
48 insert into owners values('LSG', 'Sanjeev Goenka');
49 insert into owners values('MI', 'Nita Ambani');
50 insert into owners values('MI', 'Aakash Ambani');
51 insert into owners values ('DC', 'Parth Jindal');
52 insert into owners values ('DC', 'Kiran Kumar Gandhi');
53 insert into owners values ('PBKS', 'Preity Zinta');
54 insert into owners values ('PBKS', 'Ness Wadia');
55 insert into owners values('PBKS', 'Mohit Burman');
56 insert into owners values('PBKS', 'Karan Paul');
```

```
1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.
```

PL/SQL Procedures

The PL/SQL stored procedure or simply a procedure is a PL/SQL block which performs one or more specific tasks. It is just like procedures in other programming languages.

The procedure contains a header and a body.

- Header: The header contains the name of the procedure and the parameters or variables passed to the procedure.
- Body: The body contains a declaration section, execution section and exception section similar to a general PL/SQL block.

Procedure to Print Team details

```
58  CREATE or REPLACE procedure print_team(t_name team.team_name%type)
59    AS
60       r_team team%rowtype;
61  BEGIN
62       SELECT * INTO r_team FROM team WHERE team_name = t_name;
63       dbms_output.put_line('Team is ' ||r_team.team_name);
64  dbms_output.put_line(' Having balance ' ||r_team.balance);
65  dbms_output.put_line(' captain is ' ||r_team.captain);
66  END;
67
```

```
Procedure created.
```

```
68  begin
69  print_team('SRH');
70  end;
71
```

```
Statement processed.
Team is SRH
 Having balance 9500
 Number of players are 1
 captain is Pat Cummins
```

Procedure to Print Player

```
Create or replace procedure print_player(p_name player_details.player_name%type)
IS
    r_player player_details%rowtype;

BEGIN
SELECT *INTO r_player FROM player_details WHERE player_name = p_name;
dbms_output.put_line('Player is' ||r_player.player_name);
dbms_output.put_line('no of matches played' ||r_player.Matches);
dbms_output.put_line('Total Runs' || r_player.Runs); dbms_output.put_line('Batting Avg' ||r_player.Batting_Avg);
dbms_output.put_line('Highest score' ||r_player.Best);
dbms_output.put_line('Batting Strike Rate' ||r_player.Batting_SR);
dbms_output.put_line('No. of wickets taken' ||r_player.Wickets);
dbms_output.put_line('Bowling Strike Rate '||r_player.Bowling_SR);
dbms_output.put_line('Bowling Avg' ||r_player.Bowling_Avg);
dbms_output.put_line('Status' || r_player.Status);
dbms_output.put_line('Player type' ||r_player.Player_type);
END;
```

```
Procedure created.
```

```
exec print_player('Virat Kohli');
```

```
Statement processed.
Player isVirat Kohli
no of matches played230
Total Runs6903
Batting Avg36.52
Highest score113
Batting Strike Rate129.61
No. of wickets taken4
Bowling Strike Rate 62.75
Bowling Avg92
StatusSold
Player typeBatsman
```

Procedure to Sell Player

```
CREATE or REPLACE procedure sell_player(p_name player_details.player_name%type)
AS
    t player_details.team_name%type;
BEGIN
    update player_details set status='Sold' where player_name=p_name;
if SQL%NOTFOUND then
    dbms_output.put_line('Player not present in auction');
else
    select team_name into t from player_details where player_name=p_name;
    dbms_output.put_line('Player succesfully sold to '|| t);
end if;
END;
```

```
Procedure created.
```

Procedure to add new player to Player_details

```
CREATE OR REPLACE PROCEDURE add_player(
    p_name IN player_details.player_name%TYPE,p_price IN player_details.current_price%TYPE,p_matches IN player_details.matches%TYPE,p_runs IN player_details.run
    p_sr IN player_details.batting_sr%TYPE, p_avg IN player_details.batting_avg%TYPE, p_b IN player_details.best%TYPE,w IN player_details.wickets%TYPE,
    p_bsr IN player_details.bowling_sr%TYPE,p_bavg IN player_details.bowling_avg%TYPE,p_type IN player_details.player_type%TYPE
) AS
BEGIN
    INSERT INTO player_details (
        player_name, current_price, matches, runs, batting_sr, batting_avg, best, wickets, bowling_sr, bowling_avg, player_type)
    VALUES (
        p_name,p_price,p_matches,p_runs,p_sr,p_avg,p_b,w,p_bsr,p_bavg,p_type
    );
END;
```

```
Procedure created.
```

Procedure to bid for a Player

```
CREATE OR REPLACE PROCEDURE register_bid(
    p_name IN player_details.player_name%TYPE,
    t_name IN player_details.team_name%TYPE,
    p IN player_details.current_price%TYPE
) AS
BEGIN
    UPDATE player_details
    SET team_name = t_name, current_price = current_price - p
    WHERE player_name = p_name;

    IF SQL%NOTFOUND THEN
        DBMS_OUTPUT.PUT_LINE('Player not present in the auction');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Bid Registered Successfully');
    END IF;
END;
```

```
Procedure created.
```

PL/SQL Trigger

Trigger is invoked by Oracle engine automatically whenever a specified event occurs.Trigger is stored into database and invoked repeatedly, when specific condition match.

Triggers are stored programs, which are automatically executed or fired when some event occurs.

Triggers are written to be executed in response to any of the following events.

- A database manipulation (DML) statement (DELETE, INSERT, or UPDATE).
- A database definition (DDL) statement (CREATE, ALTER, or DROP).
- A database operation (SERVERERROR, LOGON, LOGOFF, STARTUP, or SHUTDOWN).

Triggers could be defined on the table, view, schema, or database with which the event is associated.

Trigger to automatically add players in Players table and assign them Player_id

```
135
136 CREATE OR REPLACE TRIGGER add_player
137 AFTER INSERT ON player_details
138 FOR EACH ROW
139 DECLARE
140     n player.player_id%TYPE;
141 BEGIN
142     SELECT MAX(player_id) INTO n FROM player;
143     n := n + 1;
144     INSERT INTO player (player_id, player_name) VALUES (n, :NEW.player_name);
145 END;
146
```

```
Trigger created.
```

Resize Code Editor

Here we use the procedure **add_player** to add details of a new player to **player_details** table. The trigger then invokes and automatically adds the player to **player** table with an ID.

Trigger to automatically increase player_count and deduct balance from team table when a player is sold to a team

```
CREATE OR REPLACE TRIGGER team_edit
AFTER UPDATE OF status ON player_details
FOR EACH ROW
BEGIN
    IF :new.status = 'Sold' THEN
        UPDATE team
        SET balance = balance - :new.current_price,
            player_count = player_count + 1
        WHERE team_name = :new.team_name;
    END IF;
END;
```

```
Trigger created.
```

Here we use the procedure **sell_player** to change the status of a player to sold in the details table. The trigger then invokes and automatically increase the **player_count** of the team to which the player is sold and reduce the balance of the team with the price of the player.

CONCLUSION

In conclusion, the IPL auction management system developed using SQL and PL/SQL provides an efficient and effective solution for managing the complex process of the IPL auction. The system enables easy storage, retrieval, and analysis of data related to the auction, including player details, team information, and auction prices.

Through the use of SQL queries and PL/SQL procedures and triggers, the project has demonstrated the power of these tools in extracting meaningful insights from the data. The queries and procedures developed in the project have enabled the identification of the highest-paid players, the most expensive franchises and stats of the players.

In addition to its analytical capabilities, the IPL auction management system also provides a user-friendly interface for easy data entry and management. The system has been designed to be scalable, allowing for easy expansion as the IPL grows and evolves.

Overall, the IPL auction management system developed in this project showcases the potential of DBMS and SQL/PLSQL for effective management of complex data-intensive processes. The project demonstrates the importance of these tools in modern-day data-driven decision-making and provides a foundation for future work in this area.