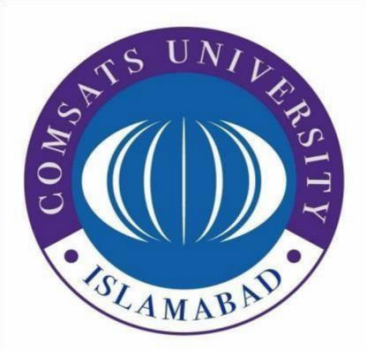
Subject: DataBase Systems

Project Documentation

Title: Tournament Management System



Submitted by: **Areeba Khan (FA22-BSE-008)**

Submitted to: **Ma'am Nighat Usman**

Department of Software Engineering

Comsats University Islamabad

Abbottabad Campus

Table of Contents

TOURNAMENT MANAGEMENT SYSTEM

1. Match Table

MatchID	TeamID1	TeamID2	VenueID	UmpireID
1	1	2	1	1
2	1	3	2	2
3	3	3	3	4
4	4	5	5	4
5	5	6	5	5

2. TicketDetails Table

TicketID	MatchID	SeatNumber	BuyerName
1	1	A1	John Doe
2	1	A2	Alice Smith
3	2	B1	Emma Watson
4	4	A1	Emma Watson
5	3	C1	Robert Downey

3. Tournament Table

TournamentID	Name	StartDate	EndDate	VenueID
1	ICC World Cup	2024-03-01	2024-04-01	1
2	IPL 2024	2024-05-01	2024-06-30	1
3	IPL 2024	2024-04-15	2024-06-31	3
4	T20 World Cup	2024-04-15	2024-09-01	4
5	Champions Trophy	2024-10-01	2024-06-31	5

4. Sponsorship Table

SponsorID	TeamID	Name	Amount	MatchID
1	1	Pepsi	5000000	1
2	2	Pepsi	5000000	1
3	2	Adidas	4000000	3
4	3	Nike	3500000	4
5	4	Puma	3000000	5

5. Player Table

Joe Root			
PlayerID	**TeamID**	**Name**	**Age**
1	1	Virat Kohli	34
2	3	Steve Smith	35
3	3	Joe Root	33
4	4	Virat Kohli	37
5	5	Kane Williamson	33

6. PlayerStats Table

PlayerStatsID	PlayerID	Runs	Wickets	Catches
1	1	500	20	15
2	2	600	30	18
3	3	500	25	10
4	4	700	30	20
5	5	550	28	18

7. Venue Table

VenueID	Name	City	Capacity
1	Eden Gardens	Kolkata	60000
2	MCG	Melbourne	55000
3	Lord's	London	30000
4	MCG	Mumbai	25000
5	The Oval	London	25000

8. Team Table

TeamID	Name	CoachID
1	India	1
2	Australia	2
3	England	3
4	Pakistan	4
5	New Zealand	5

9. Coach Table

CoachID	TeamID	Name	Experience
1	1	Ricky Ponting	10
2	3	Shane Warne	12
3	3	Shane Warne	12
4	4	Gary Kirsten	8
5	5	Justin Langer	7

10. TeamStats Table

TeamStatsID	Wins	MatchesPlayed	Losses	Points
1	10	15	5	20
2	12	16	4	24
3	8	14	5	16
4	8	14	6	16
5	14	18	4	28

11. Umpire Table

UmpireID	Name	Experience
1	Nitin Menon	12
2	Kumar Dharmasena	15
3	Aleem Dar	18
4	Marais Erasmus	15
5	Hassan ali	13

12. MatchTimeDetails Table

MatchID	StartTime	EndTime
1	10:00 AM	01:00 PM
2	11:00 AM	02:00 PM
3	10:00 AM	04:00 PM
4	02:00 PM	05:00 PM
5	03:00 PM	01:00 PM

MERGING AND SPLITTING

Merged Table: Match+Match details

1.Match details

MatchID	TeamID1	TeamID2	VenueID	UmpireID	StartTime	EndTime
1	1	2	1	1	10:00 AM	01:00 PM
2	1	3	2	2	11:00 AM	02:00 PM
3	3	3	3	4	10:00 AM	04:00 PM
4	4	5	5	4	02:00 PM	05:00 PM
5	5	6	5	5	03:00 PM	01:00 PM

Anomalies:

1. **Insert Anomaly**:
   - If you insert a match but forget to insert match details (like start time or umpire), or vice versa, the data will be incomplete.
2. **Update Anomaly**:
   - If match details (like start time) change, you will need to update every row for the match. Forgetting to do so will cause inconsistency in your data.
3. **Delete Anomaly**:
   - If you delete a match, both match and match details will be removed. If you only want to delete the match without removing its details, this will cause issues.

Splitting the table

Match Table

MatchID	TeamID1	TeamID2	VenueID	UmpireID
1	1	2	1	1
2	1	3	2	2
3	3	3	3	4
4	4	5	5	4
5	5	6	5	5

MatchTimeDetails Table

MatchID	StartTime	EndTime
1	10:00 AM	01:00 PM
2	11:00 AM	02:00 PM

3	10:00 AM	04:00 PM
4	02:00 PM	05:00 PM
5	03:00 PM	01:00 PM

Match + MatchDetails: **Lossless Decomposition**

2.Merged Table: Player+Player stats

PlayerDetails

PlayerID	TeamID	Name	Age	Runs	Wickets	Catches	PlayerStatsID
1	1	Virat Kohli	34	500	20	15	1
2	3	Steve Smith	35	600	30	18	2
3	3	Joe Root	33	500	25	10	3
4	4	Virat Kohli	37	700	30	20	4
5	5	Kane Williamson	33	550	28	18	5

Anamolies

 Insert Anomaly:

- Must insert both player and stats together. If only one is inserted, it creates incomplete data.

 Update Anomaly:

- Changing player details (e.g., age) requires updating every stat row. Forgetting this causes inconsistency.

 Delete Anomaly:

- Deleting a player removes both details and stats. If you only want to delete stats, this causes data loss.

Solution

Splitting the table

Player Table

Joe Root			
PlayerID	**TeamID**	**Name**	**Age**
1	1	Virat Kohli	34
2	3	Steve Smith	35
3	3	Joe Root	33
4	4	Virat Kohli	37
5	5	Kane Williamson	33

PlayerStats Table

PlayerStatsID	**PlayerID**	**Runs**	**Wickets**	**Catches**
1	1	500	20	15
2	2	600	30	18
3	3	500	25	10
4	4	700	30	20
5	5	550	28	18

Player + PlayerStats: **Lossless Decomposition**

3. Merged Table: team+team stats

TeamDetails

TeamID	**Name**	**CoachID**	**Wins**	**MatchesPlayed**	**Losses**	**Points**	**TeamStatsID**
1	India	1	10	15	5	20	1
2	Australia	2	12	16	4	24	2
3	England	3	8	14	5	16	3
4	Pakistan	4	8	14	6	16	4
5	New Zealand	5	14	18	4	28	5

Anamoliess

 Insert Anomaly: Must insert both team and stats together. Otherwise, incomplete data is created.

 Update Anomaly: Changing team details requires updating stats. Failing to do so causes inconsistency.

 Delete Anomaly: Deleting a team removes both details and stats. If only stats need deletion, this causes issues.

Solution

Splitting the table

Team Table

TeamID	Name	CoachID
1	India	1
2	Australia	2
3	England	3
4	Pakistan	4
5	New Zealand	5

TeamStats Table

TeamStatsID	Wins	MatchesPlayed	Losses	Points
1	10	15	5	20
2	12	16	4	24
3	8	14	5	16
4	8	14	6	16
5	14	18	4	28

Team + TeamStats: **Lossless Decomposition**

4.Merged Table: Venue +Tournament

VenueTournamentDetails

VenueID	Name	City	Capacity	TournamentID	TournamentName	StartDate	EndDate
1	Eden Gardens	Kolkata	60000	1	ICC World Cup	2024-03-01	2024-04-01
2	MCG	Melbourne	55000	2	IPL 2024	2024-05-01	2024-06-30
3	Lord's	London	30000	3	IPL 2024	2024-04-15	2024-06-31
4	MCG	Mumbai	25000	4	T20 World Cup	2024-04-15	2024-09-01
5	The Oval	London	25000	5	Champions Trophy	2024-10-01	2024-06-31

Anamolies

 Insert Anomaly: If a venue is added without a tournament (or vice versa), it creates incomplete data.

 Update Anomaly: Changing tournament details requires updating all rows for the same venue. Failure to do so causes inconsistency.

 Delete Anomaly: Deleting a venue removes both the venue and the tournament. If only one needs deletion, this causes issues.

Solution

Splitting the tables

Venue Table

VenueID	Name	City	Capacity
1	Eden Gardens	Kolkata	60000
2	MCG	Melbourne	55000
3	Lord's	London	30000
4	MCG	Mumbai	25000
5	The Oval	London	25000

Tournament Table

TournamentID	Name	StartDate	EndDate	VenueID
1	ICC World Cup	2024-03-01	2024-04-01	1
2	IPL 2024	2024-05-01	2024-06-30	1
3	IPL 2024	2024-04-15	2024-06-31	3
4	T20 World Cup	2024-04-15	2024-09-01	4
5	Champions Trophy	2024-10-01	2024-06-31	5

Venue + Tournament: **Lossless Decomposition**

5.(Merged Table) match+sponsorship

SponsorshipDetails

SponsorshipID	Match ID	Team ID	SponsorName	Amount	Venue ID	Umpire ID	StartTime	EndTime
1	1	1	Pepsi	5000000	1	1	10:00 AM	01:00 PM
2	1	2	Pepsi	5000000	1	1	10:00 AM	01:00 PM
3	2	2	Adidas	4000000	2	2	11:00 AM	02:00 PM
4	3	3	Nike	3500000	3	4	10:00 AM	04:00 PM
5	4	4	Puma	3000000	5	4	02:00 PM	05:00 PM

Explanation:

- **Insert Anomaly**: If a sponsorship or match is inserted without the corresponding match/sponsorship, it will lead to incomplete data.
- **Update Anomaly**: If sponsorship or match details change, every relevant row needs to be updated, which could lead to inconsistencies.
- **Delete Anomaly**: Deleting a match or sponsorship causes loss of relevant data.

Solution

Splitting the table

Match Table

MatchID	TeamID1	TeamID2	VenueID	UmpireID
1	1	2	1	1
2	1	3	2	2
3	3	3	3	4
4	4	5	5	4
5	5	6	5	5

Sponsorship Table

SponsorID	TeamID	Name	Amount	MatchID
1	1	Pepsi	5000000	1
2	2	Pepsi	5000000	1
3	2	Adidas	4000000	3
4	3	Nike	3500000	4
5	4	Puma	3000000	5

match+sponsorship: **Lossless Decomposition**

NOMALIZATION UPTO 3.5

1. Match Table

MatchID	TeamID1	TeamID2	VenueID	UmpireID
1	1	2	1	1
2	1	3	2	2
3	3	3	3	4
4	4	5	5	4
5	5	6	5	5

□ Functional Dependency: MatchID → TeamID1, TeamID2, VenueID, UmpireID

□ Candidate Key: MatchID

 Prime Attribute: MatchID

 Non-prime Attributes: TeamID1, TeamID2, VenueID, UmpireIDStep 4: Normalization Process

1st Normal Form (1NF)

- **Analysis**: The table already has atomic values (each cell contains a single value), so it's in **1NF**.

2nd Normal Form (2NF)

Analysis:

- There are no partial dependencies because **MatchID** is the only candidate key, and all non-prime attributes (TeamID1, TeamID2, VenueID, UmpireID) depend entirely on **MatchID**.
- **Conclusion**: The table is in **2NF**.

3rd Normal Form (3NF)

- **Analysis**:
  - There are no transitive dependencies because none of the non-prime attributes (TeamID1, TeamID2, VenueID, UmpireID) depend on each other.
  - **Conclusion**: The table is in **3NF**.

3.5 Normal Form (3.5NF)

- **Analysis**: The table has no derived dependencies because each non-prime attribute is directly dependent on the primary key (MatchID).
- **Conclusion**: The table is in **3.5NF**.

2.TicketDetails Table

TicketID	MatchID	SeatNumber	BuyerName
1	1	A1	John Doe
2	1	A2	Alice Smith
3	2	B1	Emma Watson
4	4	A1	Emma Watson
5	3	C1	Robert Downey

TicketDetails Table:

Columns: TicketID, MatchID, SeatNumber, BuyerName

Step 1: Functional Dependencies (FDs)

- **TicketID → MatchID, SeatNumber, BuyerName**
  (TicketID determines the other attributes).

Step 2: Candidate Key

- The **Candidate Key** is **TicketID** because it uniquely identifies each record.

Step 3: Prime and Non-prime Attributes

- **Prime Attribute**: **TicketID (**
- **Non-prime Attributes**: **MatchID, SeatNumber, BuyerName Step 4: Normalization Process**

1st Normal Form (1NF)

- The table is in **1NF** because each column contains atomic values (no repeating groups).

2nd Normal Form (2NF)

- The table is in **1NF**.
- **Analysis**:
  - **TicketID** is the candidate key, and all non-prime attributes (**MatchID, SeatNumber, BuyerName**) are fully dependent on **TicketID**.
  - There is **no partial dependency** because **TicketID** is the only candidate key.
- **Conclusion**: The table is in **2NF**.

3rd Normal Form (3NF)

- The table is in **2NF**.
- **Analysis**:
  - There are no transitive dependencies.
  - All non-prime attributes (**MatchID, SeatNumber, BuyerName**) depend directly on **TicketID**.
- **Conclusion**: The table is in **3NF**.

3.5 Normal Form (3.5NF)

- The table is in **3NF**.
- **Analysis**: No derived or unnecessary dependencies exist.

- **Conclusion**: The table is in **3.5NF**.

3. Tournament Table

TournamentID	Name	StartDate	EndDate	VenueID
1	ICC World Cup	2024-03-01	2024-04-01	1
2	IPL 2024	2024-05-01	2024-06-30	1
3	IPL 2024	2024-04-15	2024-06-31	3
4	T20 World Cup	2024-04-15	2024-09-01	4
5	Champions Trophy	2024-10-01	2024-06-31	5

Step 1: Functional Dependencies (FDs)

- **TournamentID → Name, StartDate, EndDate, VenueID**
  (TournamentID determines the rest of the attributes).

Step 2: Candidate Key

- The **Candidate Key** is **TournamentID** because it uniquely identifies each record.

Step 3: Prime and Non-prime Attributes

- **Prime Attribute**: **TournamentID** (since it's part of the candidate key).
- **Non-prime Attributes**: **Name, StartDate, EndDate, VenueID** (these are not part of the candidate key).

Step 4: Normalization Process

1st Normal Form (1NF)

- The table is in **1NF** because each column contains atomic values (no repeating groups).

2nd Normal Form (2NF)

- The table is in **1NF**.
- **Analysis**:
  - **TournamentID** is the candidate key, and all non-prime attributes (**Name, StartDate, EndDate, VenueID**) are fully dependent on **TournamentID**.
  - There is **no partial dependency** because **TournamentID** is the only candidate key.
- **Conclusion**: The table is in **2NF**.

3rd Normal Form (3NF)

- The table is in **2NF**.
- **Analysis**:
  - There are no transitive dependencies.
  - All non-prime attributes (**Name, StartDate, EndDate, VenueID**) depend directly on **TournamentID**.
- **Conclusion**: The table is in **3NF**.

3.5 Normal Form (3.5NF)

- The table is in **3NF**.
- **Analysis**: No derived or unnecessary dependencies exist.
- **Conclusion**: The table is in **3.5NF**.

Sponsorship Table:

4. Sponsorship Table

SponsorID	TeamID	Name	Amount	MatchID
1	1	Pepsi	5000000	1
2	2	Pepsi	5000000	1
3	2	Adidas	4000000	3
4	3	Nike	3500000	4
5	4	Puma	3000000	5

Columns: SponsorID, TeamID, Name, Amount, MatchID

Step 1: Functional Dependencies (FDs)

- **SponsorID → TeamID, Name, Amount, MatchID** (SponsorID determines all other attributes).
- **TeamID, MatchID → SponsorID, Name, Amount** (The combination of TeamID and MatchID determines SponsorID, Name, and Amount).

Step 2: Candidate Keys

- **SponsorID** is the **candidate key** because it uniquely identifies each record.
- **TeamID, MatchID** is also a **candidate key** (composite key) because this combination uniquely identifies the sponsorship.

Step 3: Prime and Non-prime Attributes

- **Prime Attributes**: **SponsorID**, **TeamID**, **MatchID** (because they are part of candidate keys).
- **Non-prime Attributes**: **Name**, **Amount** (because these are not part of the candidate key).

Step 4: Normalization Process

1st Normal Form (1NF)

- The table is already in **1NF** as it contains atomic values (no repeating groups).

2nd Normal Form (2NF)

- The table is in **2NF** because it has no partial dependencies. All non-prime attributes depend entirely on the **candidate keys**.

3rd Normal Form (3NF)

- **Analysis**:
  - There are no transitive dependencies between the non-prime attributes. However, **Name** (the sponsor's name) depends solely on **SponsorID**.
  - **SponsorID → Name** (transitive dependency through **SponsorID**)
  - This creates a situation where the **Name** attribute depends on the **primary key** (SponsorID) indirectly through the **candidate key** (TeamID, MatchID).
  - **Conclusion**: The table is **not in 3NF** because of this transitive dependency.

Step 5: 3NF Normalization

To convert the table to **3NF**, we need to remove the transitive dependency (i.e., **Name** depends on **SponsorID**).

Decomposition:

We can decompose the table into two tables:

1. **Sponsorship Table**:
   - **SponsorID, TeamID, MatchID, Amount**
   - This table will contain information about the sponsorships, with **SponsorID** as the primary key.
2. **Sponsor Information Table**:
   - **SponsorID, Name**

- This table will contain information about the sponsor (the sponsor's name), with **SponsorID** as the primary key.

Sponsorship Table (After 3NF):

SponsorID	TeamID	MatchID
1	1	1
2	2	1
3	2	3
4	3	4
5	4	5

Sponsor Information Table (After 3NF):

SponsorID	Name	Amount
1	Pepsi	5000000
2	Pepsi	5000000
3	Adidas	4000000
4	Nike	3500000
5	Puma	3000000

Step 6: 3.5NF Normalization

The tables are already in **3NF**. No derived dependencies exist.

Applying natural join:

Result (CROSS JOIN Output):

SponsorInfoID	Name	Amount	SponsorshipID	TeamID	MatchID
1	**Pepsi**	**5000000**	**1**	**1**	**1**
1	**Pepsi**	**5000000**	**2**	**2**	**1**
1	**Pepsi**	**5000000**	**3**	**2**	**3**
1	**Pepsi**	**5000000**	**4**	**3**	**4**
1	**Pepsi**	**5000000**	**5**	**4**	**5**
2	**Pepsi**	**5000000**	**1**	**1**	**1**
2	**Pepsi**	**5000000**	**2**	**2**	**1**
2	**Pepsi**	**5000000**	**3**	**2**	**3**
2	**Pepsi**	**5000000**	**4**	**3**	**4**
2	**Pepsi**	**5000000**	**5**	**4**	**5**
3	**Adidas**	**4000000**	**1**	**1**	**1**
3	**Adidas**	**4000000**	**2**	**2**	**1**

3	**Adidas**	**4000000**	**3**	**2**	**3**
3	**Adidas**	**4000000**	**4**	**3**	**4**
3	**Adidas**	**4000000**	**5**	**4**	**5**
4	**Nike**	**3500000**	**1**	**1**	**1**
4	**Nike**	**3500000**	**2**	**2**	**1**
4	**Nike**	**3500000**	**3**	**2**	**3**
4	**Nike**	**3500000**	**4**	**3**	**4**
4	**Nike**	**3500000**	**5**	**4**	**5**
5	**Puma**	**3000000**	**1**	**1**	**1**
5	**Puma**	**3000000**	**2**	**2**	**1**
5	**Puma**	**3000000**	**3**	**2**	**3**
5	**Puma**	**3000000**	**4**	**3**	**4**
5	**Puma**	**3000000**	**5**	**4**	**5**

Final Result:

SponsorID	Name	Amount	TeamID	MatchID
1	**Pepsi**	**5000000**	**1**	**1**
2	**Pepsi**	**5000000**	**2**	**1**
3	**Adidas**	**4000000**	**2**	**3**
4	**Nike**	**3500000**	**3**	**4**
5	**Puma**	**3000000**	**4**	**5**

5. Player Table

Joe Root			
PlayerID	**TeamID**	**Name**	**Age**
1	1	Virat Kohli	34
2	3	Steve Smith	35
3	3	Joe Root	33
4	4	Virat Kohli	37
5	5	Kane Williamson	33

Player Table Analysis

Columns: **PlayerID, TeamID, Name, Age**

Step 1: Functional Dependencies (FDs)

- **PlayerID → TeamID, Name, Age** (PlayerID uniquely identifies TeamID, Name, and Age for each player).

Step 2: Candidate Keys

- **PlayerID** is the **candidate key** because it uniquely identifies each record.

Step 3: Prime and Non-prime Attributes

- **Prime Attributes**: **PlayerID** (it is part of the candidate key).
- **Non-prime Attributes**: **TeamID**, **Name**, **Age** (these attributes are not part of the candidate key).

Step 4: Normalization Process

1st Normal Form (1NF)

- The table is already in **1NF** because it contains atomic values (no repeating groups).

2nd Normal Form (2NF)

- The table is in **2NF** because there are no partial dependencies. All non-prime attributes depend entirely on the **candidate key** (**PlayerID**).

3rd Normal Form (3NF)

- **Analysis**:
  - There are no transitive dependencies in this table. **Name** and **Age** are directly dependent on **PlayerID**.
  - **TeamID** is not dependent on **PlayerID**, so there's no violation of 3NF.
  - **Conclusion**: The table is already in **3NF**.

Step 5: 3.5NF Normalization

- Since the table is already in **3NF**, it is also in **3.5NF**, with no derived dependencies.

6. PlayerStats Table

PlayerStatsID	PlayerID	Runs	Wickets	Catches
1	1	500	20	15
2	2	600	30	18
3	3	500	25	10
4	4	700	30	20

5	5	550	28	18

Step 1: Functional Dependencies

- **PlayerStatsID → PlayerID, Runs, Wickets, Catches**
  (PlayerStatsID uniquely determines PlayerID and the statistics for that player).
- **PlayerID → Runs, Wickets, Catches**
  (PlayerID determines the statistics, assuming each player has only one set of statistics in the PlayerStats table).

Step 2: Candidate Keys

- **PlayerStatsID** is the **candidate key** since it uniquely identifies each row.
- **PlayerID** is **not** a candidate key because multiple players can have their statistics, but PlayerStatsID uniquely identifies the record.

Step 3: Prime and Non-prime Attributes

- **Prime Attribute**: **PlayerStatsID** (it is part of the candidate key).
- **Non-prime Attributes**: **PlayerID**, **Runs**, **Wickets**, **Catches** (these attributes are not part of the candidate key).

Step 4: Normalization Process

1st Normal Form (1NF)

- **Analysis**: The table is already in **1NF** because each column contains atomic (indivisible) values. There are no repeating groups or multi-valued attributes.
- **Conclusion**: The table is in **1NF**.

2nd Normal Form (2NF)

- **Analysis**: In 2NF, the table must meet 1NF and have no partial dependencies. Partial dependencies occur when a non-prime attribute is dependent on only part of a candidate key.
- In this case, **PlayerStatsID** is the candidate key, and all non-prime attributes (PlayerID, Runs, Wickets, Catches) depend entirely on **PlayerStatsID**.
- There are no partial dependencies.
- **Conclusion**: The table is in **2NF**.

3rd Normal Form (3NF)

- **Analysis**: In 3NF, there should be no transitive dependencies. A transitive dependency occurs when a non-prime attribute depends on another non-prime attribute.
    - **PlayerID → Runs, Wickets, Catches** creates a transitive dependency because **Runs**, **Wickets**, and **Catches** depend on **PlayerID**, which is not the candidate key.
    - This violates **3NF**.

To fix this, we will decompose the table into two separate tables to remove the transitive dependency.

Decomposition of the Table:

1. **PlayerStats Table (Decomposed)**
    - Columns: **PlayerStatsID**, **PlayerID**
    - **PlayerStatsID → PlayerID**
2. **PlayerStatsDetails Table (New Table)**
    - Columns: **PlayerID**, **Runs**, **Wickets**, **Catches**
    - **PlayerID → Runs**, **Wickets**, **Catches**

By splitting the data, **PlayerStats** and **PlayerStatsDetails** are now in **3NF**.

- **PlayerStats Table** is in **3NF** because **PlayerStatsID** directly determines **PlayerID**.
- **PlayerStatsDetails Table** is in **3NF** because **PlayerID** directly determines **Runs**, **Wickets**, and **Catches**.

3.5 Normal Form (3.5NF)

- The tables are in **3.5NF** because there are no derived dependencies. Each non-prime attribute is directly dependent on the primary key in its respective table.

Step 5: Final Tables

1. **PlayerStats Table (Decomposed)**

PlayerStatsID	PlayerID
1	1
2	2
3	3
4	4
5	5

2.PlayerStatsDetails Table

PlayerID	Runs	Wickets	Catches
1	500	20	15
2	600	30	18
3	500	25	10
4	700	30	20
5	550	28	18

Natural join

PlayerStatsID	PlayerID (from PlayerStats)	PlayerID (from PlayerStatsDetails)	Runs	Wickets	Catches
1	1	1	500	20	15
1	1	2	600	30	18
1	1	3	500	25	10
1	1	4	700	30	20
1	1	5	550	28	18
2	2	1	500	20	15
2	2	2	600	30	18
2	2	3	500	25	10
2	2	4	700	30	20
2	2	5	550	28	18
3	3	1	500	20	15
3	3	2	600	30	18
3	3	3	500	25	10
3	3	4	700	30	20
3	3	5	550	28	18
4	4	1	500	20	15
4	4	2	600	30	18
4	4	3	500	25	10
4	4	4	700	30	20
4	4	5	550	28	18
5	5	1	500	20	15
5	5	2	600	30	18
5	5	3	500	25	10
5	5	4	700	30	20
5	5	5	550	28	18

Result:

PlayerStatsID	PlayerID	Runs	Wickets	Catches
1	1	500	20	15
2	2	600	30	18

3	3	500	25	10
4	4	700	30	20
5	5	550	28	18

7. Venue Table

VenueID	Name	City	Capacity
1	Eden Gardens	Kolkata	60000
2	MCG	Melbourne	55000
3	Lord's	London	30000
4	MCG	Mumbai	25000
5	The Oval	London	25000

Step 1: Functional Dependencies

- **VenueID → Name, City, Capacity** (The venue's ID determines the name, city, and capacity of the venue.)

Step 2: Candidate Keys

- **VenueID** is the **candidate key** as it uniquely identifies each venue.

Step 3: Prime and Non-prime Attributes

- **Prime Attribute**: **VenueID** (part of the candidate key).
- **Non-prime Attributes**: **Name**, **City**, **Capacity** (these are dependent on the candidate key **VenueID**).

Step 4: Normalization Process

1st Normal Form (1NF)

- **Analysis**: The table is already in **1NF** because each column contains atomic values (no multi-valued or repeating attributes).
- **Conclusion**: The table is in **1NF**.

2nd Normal Form (2NF)

- **Analysis**: In 2NF, the table must meet **1NF** and have no partial dependencies (i.e., all non-prime attributes must depend on the entire candidate key).

  - **VenueID** is the only candidate key, and **Name**, **City**, and **Capacity** all depend entirely on **VenueID**.
  - There are no partial dependencies.
- **Conclusion**: The table is in **2NF**.

3rd Normal Form (3NF)

- **Analysis**: In **3NF**, the table should have no transitive dependencies, where a non-prime attribute depends on another non-prime attribute.
  - There are no transitive dependencies because **Name**, **City**, and **Capacity** depend directly on the **VenueID** and not on each other.
- **Conclusion**: The table is in **3NF**.

3.5 Normal Form (3.5NF)

- **Analysis**: In **3.5NF**, there should be no derived dependencies, and each non-prime attribute should be directly dependent on the primary key.
  - There are no derived dependencies, as each non-prime attribute is directly dependent on **VenueID**.

8. Team Table

TeamID	Name	CoachID
1	India	1
2	Australia	2
3	England	3
4	Pakistan	4
5	New Zealand	5

1. Functional Dependencies:

- **TeamID → Name, CoachID**

(TeamID determines Name and CoachID)

2. Step 1: First Normal Form (1NF)

1NF requires that:

- All attributes contain atomic (single) values.
- There are no repeating groups of attributes.

The original table already satisfies **1NF** because:

- All cells contain a single value.
- There are no repeating groups.

Conclusion: The table is in **1NF**.

. Step 2: Second Normal Form (2NF)

2NF requires that:

- The table must be in **1NF**.
- There must be no partial dependency, meaning non-prime attributes (attributes that are not part of the primary key) must depend on the **entire** primary key.

In this table, **TeamID** is the **primary key**, and there are no partial dependencies because **TeamID** determines both **Name** and **CoachID** directly.

Conclusion: The table is in **2NF**.

Step 3: Third Normal Form (3NF)

3NF requires that:

- The table must be in **2NF**.
- There must be no transitive dependencies, meaning non-prime attributes must not depend on other non-prime attributes.

In this case:

- **TeamID** determines **Name** and **CoachID**.
- There are no transitive dependencies because **CoachID** does not depend on **Name**, and **Name** does not depend on **CoachID**.

Conclusion: The table is in **3NF**.

Step 4: Third and a Half Normal Form (3.5NF)

3.5NF requires that:

- The table must be in **3NF**.
- There are no **derived dependencies** (i.e., no non-prime attributes are derived from other non-prime attributes).

This table doesn't have derived dependencies.

Conclusion: The table is in **3.5NF**.

9. TeamStats Table

TeamStatsID	Wins	MatchesPlayed	Losses	Points	Team id
1	10	15	5	20	1
2	12	16	4	24	2
3	8	14	5	16	3
4	8	14	6	16	4
5	14	18	4	28	5

Step 1: Functional Dependencies (FDs)

The functional dependencies in the **TeamStats** table are:

1. **TeamStatsID → Wins, MatchesPlayed, Losses, Points, TeamID**
    - **TeamStatsID** determines all other attributes in the table.
2. **TeamID → TeamName**
    - **TeamID** determines the **TeamName** (transitive dependency via TeamInfo table).

Step 2: Candidate Keys (CK)

- A **Candidate Key (CK)** is a minimal set of attributes that can uniquely identify a record.
- **TeamStatsID** is the primary key, and no other attribute or combination of attributes can uniquely identify a record in this table.

Thus, **TeamStatsID** is the **Candidate Key**.

- **TeamID** is not a candidate key by itself, as it doesn't uniquely identify a record in **TeamStats**.

Step 3: Prime and Non-Prime Attributes

- **Prime Attributes** are attributes that are part of a **candidate key**.
    - **Prime Attribute**: **TeamStatsID**
- **Non-Prime Attributes** are attributes that are not part of any candidate key.
    - **Non-Prime Attributes**: **Wins, MatchesPlayed, Losses, Points, TeamID**

Step 4: Normalization Process

1st Normal Form (1NF):

- **1NF** requires that the table has atomic values, meaning each field contains only one value, and no repeating groups.
- The **TeamStats** table is already in **1NF**, as all attributes contain atomic values and there are no repeating groups.

2nd Normal Form (2NF):

- **2NF** requires that the table is in **1NF** and has **no partial dependencies**. That means every non-prime attribute must depend on the whole candidate key.
- In this case, **TeamStatsID** is the **candidate key**, and all other attributes depend on it. There are **no partial dependencies**.

Conclusion: The table is in **2NF**.

3rd Normal Form (3NF):

- **3NF** requires that the table is in **2NF** and has **no transitive dependencies**. That means there should be no dependencies between non-prime attributes.
- **TeamID → TeamName** is a **transitive dependency** because **TeamID** determines **TeamName**, and **TeamID** is not the primary key (it's a non-prime attribute).

Conclusion: The table is **not in 3NF** due to the transitive dependency **TeamID → TeamName**.

3.5 Normal Form (3.5NF):

- **3.5NF** requires that the table is in **3NF** and has **no derived dependencies** (non-prime attributes depending on other non-prime attributes).
- To remove the transitive dependency, we separate **TeamName** into a new table.

After splitting the data into two tables, we have:

TeamStats Table

TeamStatsID	TeamID
1	1
2	2
3	3
4	4
5	5

TeamStatisticsDetails Table

TeamID	Wins	MatchesPlayed	Losses	Points
1	10	15	5	20
2	12	16	4	24
3	8	14	5	16
4	8	14	6	16
5	14	18	4	28

Natural join

TeamStatsID	TeamID (from TeamStats)	TeamID (from TeamStatisticsDetails)	Wins	MatchesPlayed	Losses	Points
1	1	1	10	15	5	20
1	1	2	12	16	4	24
1	1	3	8	14	5	16
1	1	4	8	14	6	16
1	1	5	14	18	4	28
2	2	1	10	15	5	20
2	2	2	12	16	4	24
2	2	3	8	14	5	16
2	2	4	8	14	6	16
2	2	5	14	18	4	28
3	3	1	10	15	5	20
3	3	2	12	16	4	24
3	3	3	8	14	5	16
3	3	4	8	14	6	16
3	3	5	14	18	4	28
4	4	1	10	15	5	20
4	4	2	12	16	4	24
4	4	3	8	14	5	16
4	4	4	8	14	6	16
4	4	5	14	18	4	28
5	5	1	10	15	5	20
5	5	2	12	16	4	24
5	5	3	8	14	5	16
5	5	4	8	14	6	16
5	5	5	14	18	4	28

Result

TeamStatsID	Wins	MatchesPlayed	Losses	Points	Team id
1	10	15	5	20	1
2	12	16	4	24	2
3	8	14	5	16	3
4	8	14	6	16	4
5	14	18	4	28	5

10.Umpire Table

UmpireID	Name	Experience
1	Nitin Menon	12
2	Kumar Dharmasena	15
3	Aleem Dar	18
4	Marais Erasmus	15
5	Hassan ali	13

Step 1: Functional Dependencies (FDs)

1. **UmpireID → Name, Experience**
    - **UmpireID** uniquely determines the **Name** and **Experience** of the umpire.

Step 2: Candidate Keys (CK)

- **Candidate Key**: **UmpireID**
    - **UmpireID** is the unique identifier for each record, and no other combination of attributes can uniquely identify a record.

Thus, **UmpireID** is the **Candidate Key**.

Step 3: Prime and Non-Prime Attributes

- **Prime Attributes**: Attributes that are part of the **Candidate Key**.
    - **Prime Attribute**: **UmpireID**
- **Non-Prime Attributes**: Attributes that are not part of any **Candidate Key**.
    - **Non-Prime Attributes**: **Name**, **Experience**

Step 4: Normalization Process

1st Normal Form (1NF):

- **1NF** requires that the table has atomic values, meaning each field contains only one value, and no repeating groups.
- The **Umpire Table** is already in **1NF** since all attributes have atomic values and there are no repeating groups.

2nd Normal Form (2NF):

- **2NF** requires that the table is in **1NF** and has **no partial dependencies**. A partial dependency occurs when a non-prime attribute depends on only a part of the candidate key.
- Since **UmpireID** is the only candidate key, and both **Name** and **Experience** depend entirely on **UmpireID**, there are **no partial dependencies**.

Conclusion: The table is in **2NF**.

3rd Normal Form (3NF):

- **3NF** requires that the table is in **2NF** and has **no transitive dependencies**. A transitive dependency occurs when a non-prime attribute depends on another non-prime attribute.
- In this case, **Name** and **Experience** are both directly dependent on **UmpireID**, and there is no dependency between the non-prime attributes (**Name** and **Experience**).

Conclusion: The table is in **3NF**.

3.5 Normal Form (3.5NF):

- **3.5NF** requires that the table is in **3NF** and has **no derived dependencies** (non-prime attributes depending on other non-prime attributes).
- Since the table is already in **3NF**, and there are no derived dependencies, it is also in **3.5NF**.

11. Coach Table

CoachID	TeamID	Name	Experience
1	1	Ricky Ponting	10
2	3	Shane Warne	12
3	3	Shane Warne	12
4	4	Gary Kirsten	8
5	5	Justin Langer	7

Step 1: Functional Dependencies (FDs)

Analyze the given data to determine the functional dependencies:

1. **CoachID → TeamID, Name, Experience**
   (Each CoachID uniquely determines the associated team, name, and experience.)

Step 2: Candidate Key

The **Candidate Key** is the minimal attribute(s) that uniquely identify each record.

- In this case, **CoachID** is the Candidate Key because it uniquely identifies each row.

Step 3: Prime and Non-prime Attributes

- **Prime Attribute:**
  Attributes that are part of the candidate key.
  - Prime Attribute: **CoachID**
- **Non-prime Attributes:**
  Attributes that are not part of the candidate key.
  - Non-prime Attributes: **TeamID, Name, Experience**

Step 4: Normalization Process

1st Normal Form (1NF):

- A table is in 1NF if:
  - All values are atomic (no repeating groups or multi-valued attributes).
- **Coach Table is already in 1NF** because all columns contain atomic values, and there are no repeating groups.

2nd Normal Form (2NF):

- The candidate key is **CoachID**.
- Non-prime attributes (**TeamID, Name, Experience**) are fully dependent on **CoachID** (no partial dependencies).

Conclusion:

- The table is in 2NF.

3rd Normal Form (3NF):

- Non-prime attributes (**TeamID, Name, Experience**) directly depend on the candidate key (**CoachID**).
- There are no transitive dependencies.

Conclusion:

- The table is in 3NF.

3.5 Normal Form (3.5NF or BCNF):

The only determinant is **CoachID**, which is already a candidate key.

Conclusion:

- The table is in BCNF.

12. MatchTimeDetails Table

MatchID	StartTime	EndTime
1	10:00 AM	01:00 PM
2	11:00 AM	02:00 PM
3	10:00 AM	04:00 PM
4	02:00 PM	05:00 PM
5	03:00 PM	01:00 PM

Step 1: Functional Dependencies (FDs)

1. **MatchID → StartTime, EndTime**
    - **MatchID** uniquely determines the **StartTime** and **EndTime** of a match.

Step 2: Candidate Keys (CK)

- **Candidate Key**: **MatchID**
  - **MatchID** is the unique identifier for each match, and no other combination of attributes can uniquely identify a record.

Thus, **MatchID** is the **Candidate Key**.

Step 3: Prime and Non-Prime Attributes

- **Prime Attributes**: Attributes that are part of the **Candidate Key**.
  - **Prime Attribute**: **MatchID**
- **Non-Prime Attributes**: Attributes that are not part of any **Candidate Key**.
  - **Non-Prime Attributes**: **StartTime**, **EndTime**

Step 4: Normalization Process

1st Normal Form (1NF):

- **1NF** requires that the table has atomic values, meaning each field contains only one value, and no repeating groups.
- The **MatchTimeDetails Table** is already in **1NF** since all attributes have atomic values and there are no repeating groups.

2nd Normal Form (2NF):

- **2NF** requires that the table is in **1NF** and has **no partial dependencies**. A partial dependency occurs when a non-prime attribute depends on only a part of the candidate key.
- Since **MatchID** is the only candidate key, and both **StartTime** and **EndTime** depend entirely on **MatchID**, there are **no partial dependencies**.

Conclusion: The table is in **2NF**.

3rd Normal Form (3NF):

- **3NF** requires that the table is in **2NF** and has **no transitive dependencies**. A transitive dependency occurs when a non-prime attribute depends on another non-prime attribute.
- In this case, **StartTime** and **EndTime** are both directly dependent on **MatchID**, and there is no dependency between the non-prime attributes (**StartTime** and **EndTime**).

Conclusion: The table is in **3NF**.

3.5 Normal Form (3.5NF):

- **3.5NF** requires that the table is in **3NF** and has **no derived dependencies** (non-prime attributes depending on other non-prime attributes).
- Since the table is already in **3NF**, and there are no derived dependencies, it is also in **3.5NF**.

TABLES AFTER NORMALIZATION

1. Match Table

MatchID	TeamID1	TeamID2	VenueID	UmpireID
1	1	2	1	1
2	1	3	2	2
3	3	3	3	4
4	4	5	5	4
5	5	6	5	5

2. TicketDetails Table

TicketID	MatchID	SeatNumber	BuyerName
1	1	A1	John Doe
2	1	A2	Alice Smith
3	2	B1	Emma Watson
4	4	A1	Emma Watson
5	3	C1	Robert Downey

3. Tournament Table

TournamentID	Name	StartDate	EndDate	VenueID
1	ICC World Cup	2024-03-01	2024-04-01	1
2	IPL 2024	2024-05-01	2024-06-30	1
3	IPL 2024	2024-04-15	2024-06-31	3
4	T20 World Cup	2024-04-15	2024-09-01	4
5	Champions Trophy	2024-10-01	2024-06-31	5

4.Sponsorship Table :

SponsorID	TeamID	MatchID
1	1	1

2	2	1
3	2	3
4	3	4
5	4	5

5.Sponsor Information Table:

SponsorID	Name	Amount
1	Pepsi	5000000
2	Adidas	4000000
3	Nike	3500000
4	Puma	3000000

6. Player Table

Joe Root			
PlayerID	**TeamID**	**Name**	**Age**
1	1	Virat Kohli	34
2	3	Steve Smith	35
3	3	Joe Root	33
4	4	Virat Kohli	37
5	5	Kane Williamson	33

7.PlayerStats Table :

PlayerStatsID	PlayerID
1	1
2	2
3	3
4	4
5	5

8.PlayerStatsDetails Table

PlayerID	Runs	Wickets	Catches
1	500	20	15
2	600	30	18
3	500	25	10
4	700	30	20

5	550	28	18

9. Team Table

TeamID	Name	CoachID
1	India	1
2	Australia	2
3	England	3
4	Pakistan	4
5	New Zealand	5

10. Coach Table

CoachID	TeamID	Name	Experience
1	1	Ricky Ponting	10
2	3	Shane Warne	12
3	3	Shane Warne	12
4	4	Gary Kirsten	8
5	5	Justin Langer	7

11.1TeamStats Table

TeamStatsID	TeamID
1	1
2	2
3	3
4	4
5	5

12.TeamStatisticsDetails Table

TeamID	Wins	MatchesPlayed	Losses	Points
1	**10**	**15**	**5**	**20**
2	**12**	**16**	**4**	**24**
3	**8**	**14**	**5**	**16**
4	**8**	**14**	**6**	**16**
5	**14**	**18**	**4**	**28**

13. Umpire Table

UmpireID	Name	Experience
1	Nitin Menon	12

2	Kumar Dharmasena	15
3	Aleem Dar	18
4	Marais Erasmus	15
5	Hassan ali	13

14. MatchTimeDetails Table

MatchID	StartTime	EndTime
1	10:00 AM	01:00 PM
2	11:00 AM	02:00 PM
3	10:00 AM	04:00 PM
4	02:00 PM	05:00 PM
5	03:00 PM	01:00 PM

15.Venue Table

VenueID	Name	City	Capacity
1	Eden Gardens	Kolkata	60000
2	MCG	Melbourne	55000
3	Lord's	London	30000
4	MCG	Mumbai	25000
5	The Oval	London	25000

ATTRIBUTES AND RELATIONSHIP

1. Match Table

- **Attributes:** MatchID, TeamID1, TeamID2, VenueID, UmpireID
- **Relationships:**
  - One-to-Many with TicketDetails (Match has many tickets)
  - Many-to-One with Venue (Match happens at one venue)
  - Many-to-One with Umpire (Match has one umpire)
  - Many-to-One with Team (TeamID1 and TeamID2 are teams)
  - Many-to-Many with Tournament (Match can be part of many tournaments)

2. TicketDetails Table

- **Attributes:** TicketID, MatchID, SeatNumber, BuyerName
- **Relationships:**
  - Many-to-One with Match (A ticket is associated with a match)

3. Tournament Table

- **Attributes:** TournamentID, Name, StartDate, EndDate, VenueID
- **Relationships:**
  - Many-to-Many with Match (Tournament has multiple matches)
  - Many-to-One with Venue (Tournament occurs at a venue)

4. SponsorInformation Table

- **Attributes:** SponsorID, Name, Amount
- **Relationships:**
  - One-to-Many with Sponsorship (Sponsor sponsors multiple matches)

5. Sponsorship Table

- **Attributes:** SponsorID, TeamID, MatchID
- **Relationships:**
  - Many-to-One with SponsorInformation (Sponsorship links to sponsor)
  - Many-to-One with Team (A team is sponsored by a sponsor)
  - Many-to-One with Match (A sponsorship is linked to a match)

6. Player Table

- **Attributes:** PlayerID, TeamID, Name, Age
- **Relationships:**
  - Many-to-One with Team (Player belongs to a team)
  - One-to-One with PlayerStats (Player has one set of stats)

7. PlayerStats Table

- **Attributes:** PlayerStatsID, PlayerID
- **Relationships:**
  - One-to-One with Player (Player has one set of stats)
  - One-to-One with PlayerStatsDetails (PlayerStats contains detailed stats)

8. PlayerStatsDetails Table

- **Attributes:** PlayerID, Runs, Wickets, Catches
- **Relationships:**
  - Many-to-One with PlayerStats (Stats are linked to PlayerStats)

9. Venue Table

- **Attributes:** VenueID, Name, City, Capacity
- **Relationships:**
  - One-to-Many with Match (A venue hosts many matches)
  - One-to-Many with Tournament (A venue can host multiple tournaments)

10. Team Table

- **Attributes:** TeamID, Name, CoachID
- **Relationships:**
  - One-to-Many with Player (A team has many players)
  - One-to-Many with Sponsorship (A team can have multiple sponsors)
  - One-to-Many with TeamStats (A team has one set of stats)
  - Many-to-One with Coach (A team has one coach)

11. Coach Table

- **Attributes:** CoachID, TeamID, Name, Experience
- **Relationships:**
  - One-to-One with Team (A coach works for one team)

12. TeamStats Table

- **Attributes:** TeamStatsID, Wins, MatchesPlayed, Losses, Points, TeamID
- **Relationships:**
  - Many-to-One with Team (Stats belong to one team)

13. TeamStatisticsDetails Table (New Entity)

- **Attributes:** TeamID, Wins, MatchesPlayed, Losses, Points
- **Relationships:**
  - Many-to-One with TeamStats (Stats are linked to team statistics)

14. Umpire Table

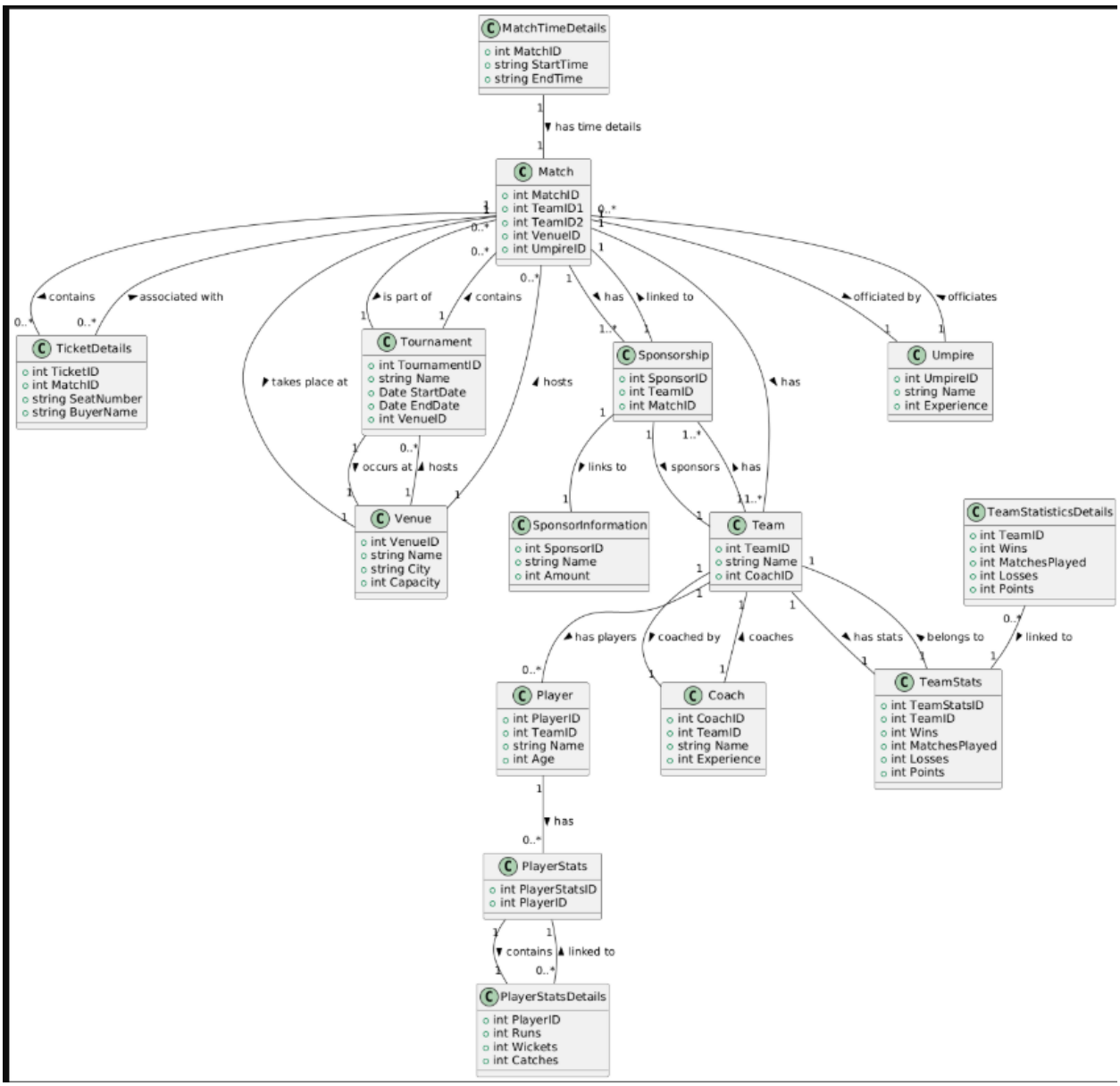
- **Attributes:** UmpireID, Name, Experience
- **Relationships:**
  - One-to-Many with Match (An umpire officiates multiple matches)

15. MatchTimeDetails Table

- **Attributes:** MatchID, StartTime, EndTime

- **Relationships:**
  - Many-to-One with Match (Time details are linked to a match)

ERD DIAGRAM



DATABASE QUERIES FOR CRICKET MANAGEMENT SYSTEM

Button 1: Insert Player Data

Purpose: Insert data into the Player table, including PlayerID, TeamID, Name, and Age.

```
INSERT INTO Player (PlayerID, TeamID, Name, Age)
VALUES (@PlayerID, @TeamID, @Name, @Age);
```

Button 2: Fetch Match Details with Team Names, Venue, and Umpire Information

Purpose: Retrieve match details with team names, venue name, venue city, and umpire information.

```
SELECT
   m.MatchID,
   t1.Name AS Team1Name,
   t2.Name AS Team2Name,
   v.Name AS VenueName,
   v.City AS VenueCity,
   u.Name AS UmpireName,
   u.Experience AS UmpireExperience
FROM Match m
JOIN Team t1 ON m.TeamID1 = t1.TeamID
JOIN Team t2 ON m.TeamID2 = t2.TeamID
JOIN Venue v ON m.VenueID = v.VenueID
JOIN Umpire u ON m.UmpireID = u.UmpireID;
```

Button 3: Fetch Venue and Umpire Details

Purpose: Get venue city and umpire name for each match.

```
SELECT
   m.MatchID,
   (SELECT City FROM Venue v WHERE v.VenueID = m.VenueID) AS VenueCity,
   (SELECT Name FROM Umpire u WHERE u.UmpireID = m.UmpireID) AS UmpireName
FROM Match m;
```

Button 4: Get Player Details by Name

Purpose: Retrieve player details (such as runs, wickets, and catches) for a specific player (e.g., Virat Kohli).

```
SELECT
   p.PlayerID,
   p.Name AS PlayerName,
   t.Name AS TeamName,
   ps.Runs,
   ps.Wickets,
   ps.Catches
FROM Player p
JOIN Team t ON p.TeamID = t.TeamID
JOIN PlayerStatsDetails ps ON p.PlayerID = ps.PlayerID
WHERE p.Name = 'Virat Kohli';
```

Button 5: Get Average Wickets Per Team

Purpose: Calculate the average number of wickets taken by players in each team.

```
SELECT
   t.Name AS TeamName,
   AVG(ps.Wickets) AS AverageWickets
FROM Player p
JOIN Team t ON p.TeamID = t.TeamID
JOIN PlayerStatsDetails ps ON p.PlayerID = ps.PlayerID
GROUP BY t.Name;
```

Button 6: Buyer Name, Tournament Name, and Venue Name

Purpose: Retrieve ticket details, including buyer name, seat number, tournament name, and venue name.

```
SELECT
   td.TicketID,
   td.BuyerName,
   td.SeatNumber,  -- Added SeatNumber
   t.Name AS TournamentName,
   v.Name AS VenueName
FROM TicketDetails td
JOIN Match m ON td.MatchID = m.MatchID
JOIN Tournament t ON m.VenueID = t.VenueID
```

```
JOIN Venue v ON m.VenueID = v.VenueID;
```

Button 7: Match Stats (Runs, Catches)

Purpose: Retrieve the total runs and catches for both teams in each match.

```
SELECT
   m.MatchID,
   t1.Name AS Team1Name,
   SUM(ps1.Runs) AS Team1TotalRuns,
   SUM(ps1.Catches) AS Team1TotalCatches,
   t2.Name AS Team2Name,
   SUM(ps2.Runs) AS Team2TotalRuns,
   SUM(ps2.Catches) AS Team2TotalCatches
FROM Match m
JOIN Player p1 ON p1.TeamID = m.TeamID1
JOIN PlayerStatsDetails ps1 ON p1.PlayerID = ps1.PlayerID
JOIN Team t1 ON m.TeamID1 = t1.TeamID
JOIN Player p2 ON p2.TeamID = m.TeamID2
JOIN PlayerStatsDetails ps2 ON p2.PlayerID = ps2.PlayerID
JOIN Team t2 ON m.TeamID2 = t2.TeamID
GROUP BY m.MatchID, t1.Name, t2.Name;
```

Button 8: Find the Venues, Points, and Team Where Matches Were Played by the Team with the Most Points

Purpose: Find the venues, points, and teams where matches were played by the team with the most points.

```
SELECT DISTINCT
   v.Name AS VenueName,
   tsd.Points AS Points,
   t.Name AS TeamName
FROM Venue v
JOIN Match m ON v.VenueID = m.VenueID
JOIN TeamStatisticsDetails tsd ON tsd.TeamID = m.TeamID1 OR tsd.TeamID = m.TeamID2
JOIN Team t ON t.TeamID = tsd.TeamID
WHERE
   (m.TeamID1 = (
      SELECT TeamID
      FROM TeamStatisticsDetails
      WHERE Points = (SELECT MAX(Points) FROM TeamStatisticsDetails)
   )
```

```
   OR m.TeamID2 = (
      SELECT TeamID
      FROM TeamStatisticsDetails
      WHERE Points = (SELECT MAX(Points) FROM TeamStatisticsDetails)
   ))
   AND tsd.Points = (SELECT MAX(Points) FROM TeamStatisticsDetails);
```

Button 9: Average Age of Players Per Team

Purpose: Calculate the average age of players for each team.

```
SELECT
   t.Name AS TeamName,
   AVG(p.Age) AS AveragePlayerAge
FROM Player p
JOIN Team t ON p.TeamID = t.TeamID
GROUP BY t.Name
ORDER BY AveragePlayerAge DESC;
```

Button 10: Team with Average Experienced Coaches

Purpose: Retrieve teams with coaches having more experience than the average experience.

```
SELECT
   t.Name AS TeamName,
   c.Name AS CoachName,
   c.Experience AS CoachExperience
FROM Coach c
JOIN Team t ON c.TeamID = t.TeamID
WHERE c.Experience > (SELECT AVG(Experience) FROM Coach);
```

Button 11: Team with Highest Sponsorship

Purpose: Retrieve the team with the highest total sponsorship amount from the Sponsorship table, calculating the sum of sponsorship amounts for each team.

```
SELECT
   T.TeamID,
   T.Name AS TeamName,
   S.TotalSponsorship
FROM
   Team T
INNER JOIN (
```

```
    SELECT
        TeamID,
        SUM(SI.Amount) AS TotalSponsorship
    FROM
        Sponsorship S
    INNER JOIN SponsorInformation SI ON S.SponsorID = SI.SponsorID
    GROUP BY
        TeamID
) AS S ON T.TeamID = S.TeamID
WHERE
    S.TotalSponsorship = (
        SELECT MAX(SumSponsorship)
        FROM (
            SELECT
                TeamID,
                SUM(SI.Amount) AS SumSponsorship
            FROM
                Sponsorship S
            INNER JOIN SponsorInformation SI ON S.SponsorID = SI.SponsorID
            GROUP BY
                TeamID
        ) AS InnerQuery
    );
```

Button 12: Match Duration in Minutes

Purpose: Calculate the duration of each match in minutes based on the StartTime and EndTime from the MatchTimeDetails table.

```
SELECT
    MatchID,
    StartTime,
    EndTime,
    DATEDIFF(MINUTE, StartTime, EndTime) AS DurationInMinutes
FROM
    MatchTimeDetails;
```