Design and Implementation of NBA Game Database

by **Ankit Akash**

1. Description of the Business Operation Task for the Database Design

The NBA game database is designed to capture comprehensive data related to NBA games, teams, players, and their performance statistics. The primary business operation tasks for the database design are as follows:

- **Data Collection and Storage:** The database must store detailed information about teams, players, games, and game statistics. This includes data on team identities, player profiles, game outcomes, and individual player performance metrics.
- **Data Retrieval and Analysis:** The database must support efficient retrieval and analysis of data to answer complex business questions. Analysts, coaches, and journalists need to extract insights about player performance, team effectiveness, and game outcomes.
- **Reporting and Decision Making:** The data stored in the database should facilitate reporting and support decision-making processes. This includes generating reports on top performers, average statistics, and trends over time.
- **Maintaining Data Integrity:** The database must ensure data integrity through well-defined relationships and constraints. This includes ensuring that each player is associated with a team, each game involves two teams, and each performance statistic is linked to a specific game and player.
- **Scalability and Efficiency:** The database must be scalable to handle increasing volumes of data over time and efficient to support quick query execution and data manipulation.

These requirements form the basis for the database design, ensuring that it meets the needs of stakeholders and supports key business operations effectively.

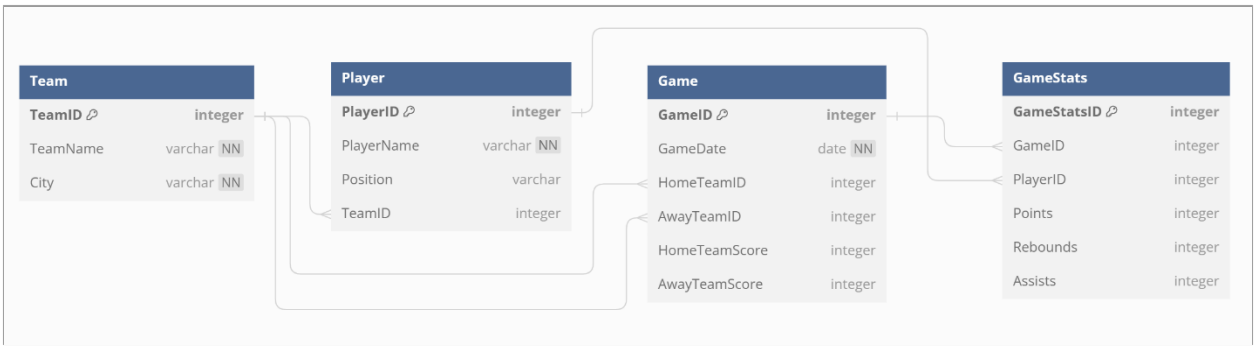
The objective of this project is to design and implement a database for managing and analyzing data related to NBA games. The database will capture detailed information about teams, players, games, and game statistics. This system will enable stakeholders to query and retrieve meaningful insights such as top performers in specific games, average points scored by teams, and individual player statistics across games.

2. Entity-Relationship Model

The entity-relationship model (ER model) is essential for visualizing and designing the structure of the database. This model outlines the entities involved, their attributes, and the relationships between them. The primary entities identified for this NBA game database are:

1. **Team**
    - Attributes: `TeamID`, `TeamName`, `City`
    - Each team is uniquely identified by `TeamID`.
2. **Player**
    - Attributes: `PlayerID`, `PlayerName`, `Position`, `TeamID`
    - Each player is uniquely identified by `PlayerID` and is associated with a team through `TeamID`.
3. **Game**
    - Attributes: `GameID`, `GameDate`, `HomeTeamID`, `AwayTeamID`, `HomeTeamScore`, `AwayTeamScore`
    - Each game is uniquely identified by `GameID` and involves two teams identified by `HomeTeamID` and `AwayTeamID`.
4. **GameStats**
    - Attributes: `GameStatsID`, `GameID`, `PlayerID`, `Points`, `Rebounds`, `Assists`
    - Each record captures the performance of a player in a specific game, identified by `GameStatsID`.

Below is the ERD (Entity-Relationship Diagram):



3. Relational Schema

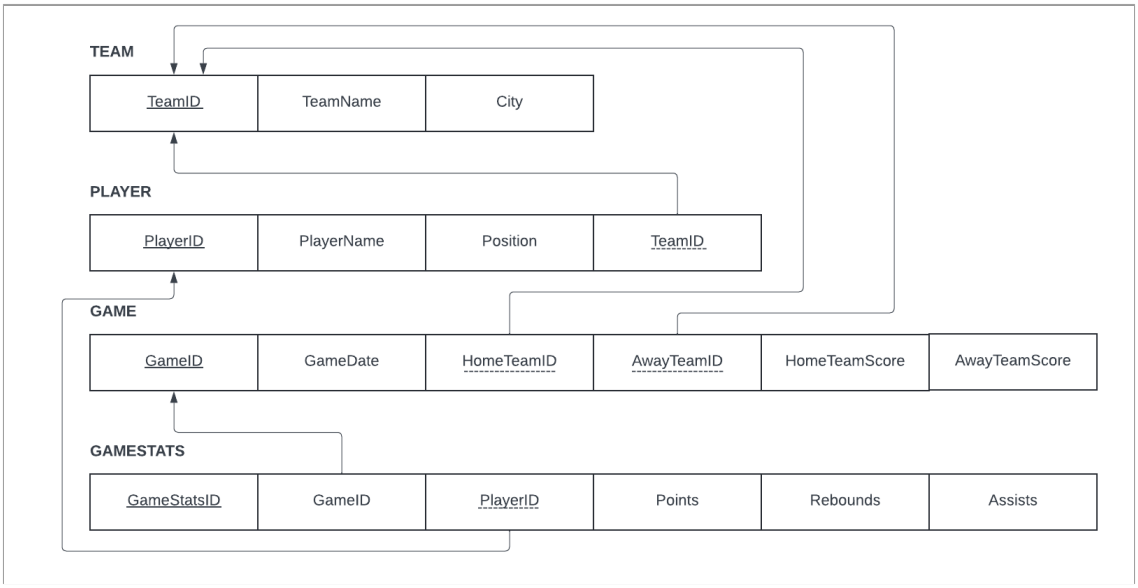
The relational schema translates the ER model into a set of relations (tables). Here is the schema for our database:

Team (`TeamID`, `TeamName`, `City`)

Player (`PlayerID`, `PlayerName`, `Position`, `TeamID`)

Game (`GameID`, `GameDate`, `HomeTeamID`, `AwayTeamID`, `HomeTeamScore`, `AwayTeamScore`)

GameStats (`GameStatsID`, `GameID`, `PlayerID`, `Points`, `Rebounds`, `Assists`)



4. Process for Creating the Database

Creating the NBA game database involves defining tables, fields in those tables, primary keys, foreign keys, and indexes. This process ensures data integrity, facilitates efficient data retrieval, and supports complex queries. Below is a detailed description of the process:

Steps to Create the Database

1. **Define the Tables:** The database includes four main tables: `Team`, `Player`, `Game`, and `GameStats`.
2. **Specify Fields in Tables:** Each table contains specific columns to store relevant data.
3. **Define Primary Keys:** Primary keys ensure each record can be uniquely identified.
4. **Establish Foreign Keys:** Foreign keys define relationships between tables and ensure referential integrity.
5. **Create Indexes:** Indexes improve the speed of data retrieval operations.

Table Definitions

Team Table:

- **Purpose:** Stores information about NBA teams.
- **Columns:**
    - `TeamID`: Integer, primary key.
    - `TeamName`: Varchar(50).
    - `City`: Varchar(50).

SQL Statement:

```
CREATE TABLE Team (
    TeamID NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    TeamName VARCHAR2(50) NOT NULL,
    City VARCHAR2(50) NOT NULL
);
```

Player Table:

- **Purpose:** Stores information about players.

- **Columns:**
    - `PlayerID`: Integer, primary key.
    - `PlayerName`: Varchar(50).
    - `Position`: Varchar(10).
    - `TeamID`: Integer, foreign key.

SQL Statement:

```
CREATE TABLE Player (
    PlayerID NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    PlayerName VARCHAR2(50) NOT NULL,
    Position VARCHAR2(10) NOT NULL,
    TeamID NUMBER,
    CONSTRAINT fk_team FOREIGN KEY (TeamID) REFERENCES Team(TeamID)
);
```

Game Table:

- **Purpose:** Stores information about NBA games.
- **Columns:**
    - `GameID`: Integer, primary key.
    - `GameDate`: Date.
    - `HomeTeamID`: Integer, foreign key.
    - `AwayTeamID`: Integer, foreign key.
    - `HomeTeamScore`: Integer.
    - `AwayTeamScore`: Integer.

SQL Statement:

```
CREATE TABLE Game (
    GameID NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    GameDate DATE NOT NULL,
    HomeTeamID NUMBER NOT NULL,
    AwayTeamID NUMBER NOT NULL,
    HomeTeamScore NUMBER,
    AwayTeamScore NUMBER,
    CONSTRAINT fk_home_team FOREIGN KEY (HomeTeamID) REFERENCES Team(TeamID),
    CONSTRAINT fk_away_team FOREIGN KEY (AwayTeamID) REFERENCES Team(TeamID)
);
```

GameStats Table:

- **Purpose:** Stores statistics for each player in each game.

- **Columns:**
    - `GameStatsID`: Integer, primary key.
    - `GameID`: Integer, foreign key.
    - `PlayerID`: Integer, foreign key.
    - `Points`: Integer.
    - `Rebounds`: Integer.
    - `Assists`: Integer.

SQL Statement:

```
CREATE TABLE GameStats (
    GameStatsID NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    GameID NUMBER NOT NULL,
    PlayerID NUMBER NOT NULL,
    Points NUMBER,
    Rebounds NUMBER,
    Assists NUMBER,
    CONSTRAINT fk_game FOREIGN KEY (GameID) REFERENCES Game(GameID),
    CONSTRAINT fk_player FOREIGN KEY (PlayerID) REFERENCES Player(PlayerID)
);
```

5. Business Questions and SQL Statements

In this section, we identify and address three complex business questions that can be answered using the NBA game database. These questions are designed to demonstrate the analytical capabilities of the database and involve queries that make use of various SQL features including joins, aggregate functions, and the GROUP BY clause. Here's an explanation of each business question, why it is complex, and the corresponding SQL statements to answer them:

Business Question 1: Who is the top scorer in a specific game?

Complexity and Business Relevance:

Identifying the top scorer in a game is crucial for performance analysis and highlighting key players. This question requires joining multiple tables to fetch player names and their respective scores, sorting the results to find the highest scorer. It is complex because it involves:

- Joining the `GameStats` table with the `Player` table to link player names with their performance stats.
- Filtering records to focus on a specific game.
- Ordering results to identify the player with the highest points.

SQL Statement:

```
SELECT p.PlayerName, gs.Points
FROM GameStats gs
JOIN Player p ON gs.PlayerID = p.PlayerID
WHERE gs.GameID = 1
ORDER BY gs.Points DESC
FETCH FIRST 1 ROWS ONLY;
```

This query:

- Joins `GameStats` with `Player` to get player names.
- Filters the data for a specific game (GameID = 1).
- Orders the results by points in descending order.
- Retrieves the player with the highest points.

Output:

Business Question 2: What is the average points scored by players in each team?

Complexity and Business Relevance:

Understanding the average points scored by players in each team provides insights into team performance and player contributions. This question requires:

- Aggregating data to calculate average points.
- Grouping results by team, which involves joining multiple tables.
- Ensuring that the calculations are accurate and reflect the correct team associations.

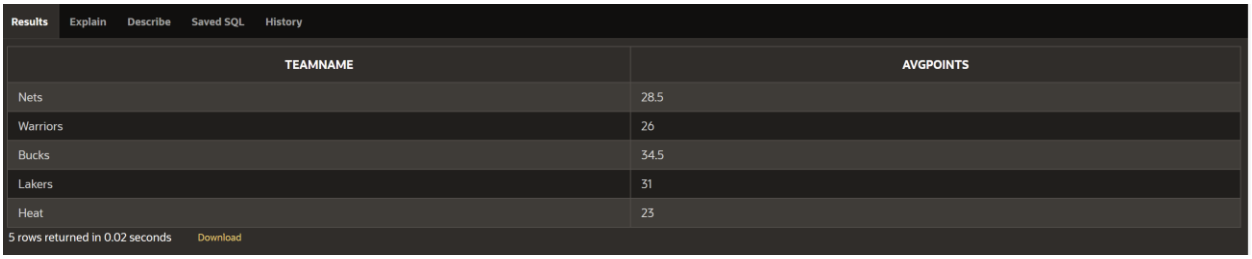
SQL Statement:

```
SELECT t.TeamName, AVG(gs.Points) AS AvgPoints
FROM GameStats gs
JOIN Player p ON gs.PlayerID = p.PlayerID
JOIN Team t ON p.TeamID = t.TeamID
GROUP BY t.TeamName;
```

This query:

- Joins `GameStats` with `Player` and `Team` to link player performance to teams.
- Groups the results by team name.
- Calculates the average points scored by players within each team using the `AVG` function.

Output:



Results | Explain | Describe | Saved SQL | History

TEAMNAME	AVGPOINTS
Nets	28.5
Warriors	26
Bucks	34.5
Lakers	31
Heat	23

5 rows returned in 0.02 seconds Download

Business Question 3: What are the details of games played by a specific player?

Complexity and Business Relevance:

Tracking the performance and involvement of a specific player in various games is essential for player evaluation and strategy formulation. This question requires:

- Joining tables to gather comprehensive game details.
- Filtering data to focus on a specific player.
- Combining multiple attributes to provide a detailed view of the player's performance across games.

SQL Statement:

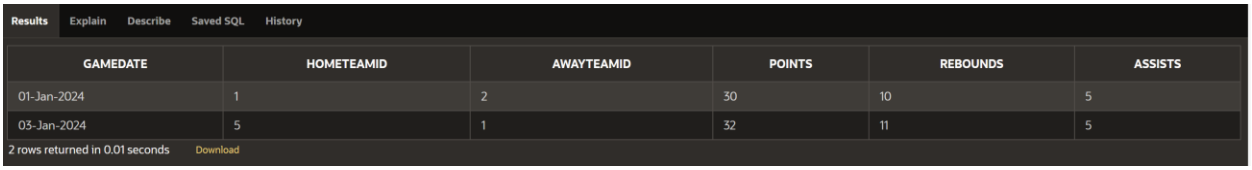
```
SELECT g.GameDate, g.HomeTeamID, g.AwayTeamID, gs.Points, gs.Rebounds,
gs.Assists
FROM GameStats gs
JOIN Game g ON gs.GameID = g.GameID
```

```
WHERE gs.PlayerID = 1;
```

This query:

- Joins `GameStats` with `Game` to retrieve game details.
- Filters the results to include only the records for a specific player (PlayerID = 1).
- Selects relevant columns to provide detailed information on game dates, team involvement, and individual performance metrics (points, rebounds, assists).

Output:



Results | Explain | Describe | Saved SQL | History

GAMEDATE	HOMETEAMID	AWAYTEAMID	POINTS	REBOUNDS	ASSISTS
01-Jan-2024	1	2	30	10	5
03-Jan-2024	5	1	32	11	5

2 rows returned in 0.01 seconds Download

6. Explanation of Complexity

These business questions are complex for several reasons:

1. **Multiple Table Joins:** Each query involves joining multiple tables to gather and correlate the necessary data. This requires an understanding of the relationships between different entities (teams, players, games, and statistics).
2. **Aggregate Functions:** Calculations such as averages necessitate the use of aggregate functions, which add a layer of complexity in ensuring that the data is grouped and calculated correctly.
3. **Filtering and Sorting:** The need to filter data based on specific criteria (e.g., a particular game or player) and to sort results to identify key metrics (e.g., top scorer) adds complexity to the queries.
4. **Data Integrity:** Ensuring that the joins and calculations maintain data integrity and produce accurate results requires careful design and execution of the SQL statements.

By addressing these complex business questions, the database demonstrates its capability to support detailed and nuanced analyses, providing valuable insights into player and team performances in the NBA.

7. Tables, Columns, Primary Keys, Foreign Keys, Indexes

Below are the detailed specifications of the database tables and their constraints:

- **Team**:
  - Columns: `TeamID` (Primary Key), `TeamName`, `City`
  - Primary Key: `TeamID` is a unique identifier for each team, ensuring that each team can be uniquely identified in the database.
  - Index: Primary Key on `TeamID`
- **Player**:
  - Columns: `PlayerID` (Primary Key), `PlayerName`, `Position`, `TeamID` (Foreign Key)
  - Primary Key: `PlayerID` is a unique identifier for each player, ensuring that each player can be uniquely identified.
  - Foreign Key: `TeamID` links each player to a team, enforcing referential integrity by ensuring that each player belongs to a valid team. This is enforced by the foreign key constraint fk_team, which references Team(TeamID).
  - Index: Primary Key on `PlayerID`, Foreign Key on `TeamID` referencing `Team(TeamID)`
- **Game**:
  - Columns: `GameID` (Primary Key), `GameDate`, `HomeTeamID` (Foreign Key), `AwayTeamID` (Foreign Key), `HomeTeamScore`, `AwayTeamScore`
  - Primary Key: `GameID` is a unique identifier for each game, ensuring that each game can be uniquely identified.
  - Foreign Keys:
    - `HomeTeamID` ensures that the home team in each game is a valid team by referencing Team(`TeamID`). This is enforced by the foreign key constraint fk_home_team.
    - `AwayTeamID`: Ensures that the away team in each game is a valid team by referencing Team(`TeamID`). This is enforced by the foreign key constraint fk_away_team.

- Index: Primary Key on `GameID`, Foreign Keys on `HomeTeamID` and `AwayTeamID` referencing `Team(TeamID)`

- **GameStats**:
  - Columns: `GameStatsID` (Primary Key), `GameID` (Foreign Key), `PlayerID` (Foreign Key), `Points`, `Rebounds`, `Assists`
  - Primary Key: `GameStatsID` is a unique identifier for each set of game statistics, ensuring that each record can be uniquely identified.
  - Foreign Keys:
    - `GameID`: Links each set of game statistics to a specific game, enforcing referential integrity by ensuring that each record is associated with a valid game. This is enforced by the foreign key constraint fk_game, which references Game(GameID).
    - `PlayerID`: Links each set of game statistics to a specific player, enforcing referential integrity by ensuring that each record is associated with a valid player. This is enforced by the foreign key constraint fk_player, which references Player(PlayerID).
  - Index: Primary Key on `GameStatsID`, Foreign Keys on `GameID` referencing `Game(GameID)` and `PlayerID` referencing `Player(PlayerID)`

8. Populating the Database with Data

To ensure the database is functional and can support the required queries, we populate it with sample data:

```
-- Insert data into Team table
INSERT INTO Team (TeamName, City) VALUES ('Lakers', 'Los Angeles');
INSERT INTO Team (TeamName, City) VALUES ('Warriors', 'San Francisco');
INSERT INTO Team (TeamName, City) VALUES ('Nets', 'Brooklyn');
INSERT INTO Team (TeamName, City) VALUES ('Bucks', 'Milwaukee');
INSERT INTO Team (TeamName, City) VALUES ('Heat', 'Miami');

-- Insert data into Player table
INSERT INTO Player (PlayerName, Position, TeamID) VALUES ('LeBron James',
'SF', 1);
INSERT INTO Player (PlayerName, Position, TeamID) VALUES ('Stephen Curry',
'PG', 2);
```

```
INSERT INTO Player (PlayerName, Position, TeamID) VALUES ('Kevin Durant',
'SF', 3);
INSERT INTO Player (PlayerName, Position, TeamID) VALUES ('Giannis
Antetokounmpo', 'PF', 4);
INSERT INTO Player (PlayerName, Position, TeamID) VALUES ('Jimmy Butler',
'SF', 5);

-- Insert data into Game table
INSERT INTO Game (GameDate, HomeTeamID, AwayTeamID, HomeTeamScore,
AwayTeamScore) VALUES (TO_DATE('2024-01-01', 'YYYY-MM-DD'), 1, 2, 102, 99);
INSERT INTO Game (GameDate, HomeTeamID, AwayTeamID, HomeTeamScore,
AwayTeamScore) VALUES (TO_DATE('2024-01-02', 'YYYY-MM-DD'), 3, 4, 110, 108);
INSERT INTO Game (GameDate, HomeTeamID, AwayTeamID, HomeTeamScore,
AwayTeamScore) VALUES (TO_DATE('2024-01-03', 'YYYY-MM-DD'), 5, 1, 95, 105);
INSERT INTO Game (GameDate, HomeTeamID, AwayTeamID, HomeTeamScore,
AwayTeamScore) VALUES (TO_DATE('2024-01-04', 'YYYY-MM-DD'), 2, 3, 98, 100);
INSERT INTO Game (GameDate, HomeTeamID, AwayTeamID, HomeTeamScore,
AwayTeamScore) VALUES (TO_DATE('2024-01-05', 'YYYY-MM-DD'), 4, 5, 112, 103);

-- Insert data into GameStats table
INSERT INTO GameStats (GameID, PlayerID, Points, Rebounds, Assists) VALUES
(1, 1, 30, 10, 8);
INSERT INTO GameStats (GameID, PlayerID, Points, Rebounds, Assists) VALUES
(1, 2, 28, 5, 10);
INSERT INTO GameStats (GameID, PlayerID, Points, Rebounds, Assists) VALUES
(2, 3, 35, 8, 5);
INSERT INTO GameStats (GameID, PlayerID, Points, Rebounds, Assists) VALUES
(2, 4, 32, 12, 6);
INSERT INTO GameStats (GameID, PlayerID, Points, Rebounds, Assists) VALUES
(3, 5, 25, 7, 9);
```

9. Data Supporting SQL Execution

The data inserted into the database supports the execution of the SQL statements provided in Section 5. The sample data includes a variety of player performances and game outcomes, enabling meaningful analysis and query results.

10. Summary

The NBA game database design project encompasses the full cycle of database development, from requirements analysis to implementation. The database is designed to capture comprehensive data on NBA games, teams, players, and their performance statistics, ensuring data integrity and enabling detailed analysis.

Key components include:

- **Entity-Relationship Model:** Illustrates the relationships between teams, players, games, and game statistics.
- **Relational Schema:** Translates the ER model into a structured set of relations (tables).
- **Database Creation Process:** Includes DDL commands to create tables and define constraints.
- **Business Questions:** Demonstrates the analytical capabilities of the database through complex SQL queries addressing specific business needs.
- **Data Integrity:** Ensured through primary and foreign key constraints, maintaining referential integrity and supporting efficient data retrieval.

This project showcases the practical application of database design principles, highlighting the importance of structure, integrity, and analytical capability in managing and analyzing sports data.