NBA Players and Teams Relational Database

Mingkuan Pang, Mingzan Liu, Yilong Wang, Tien Tran, and Johnny Dean

CSCI 126

Dr. David Ruby

April 26, 2022

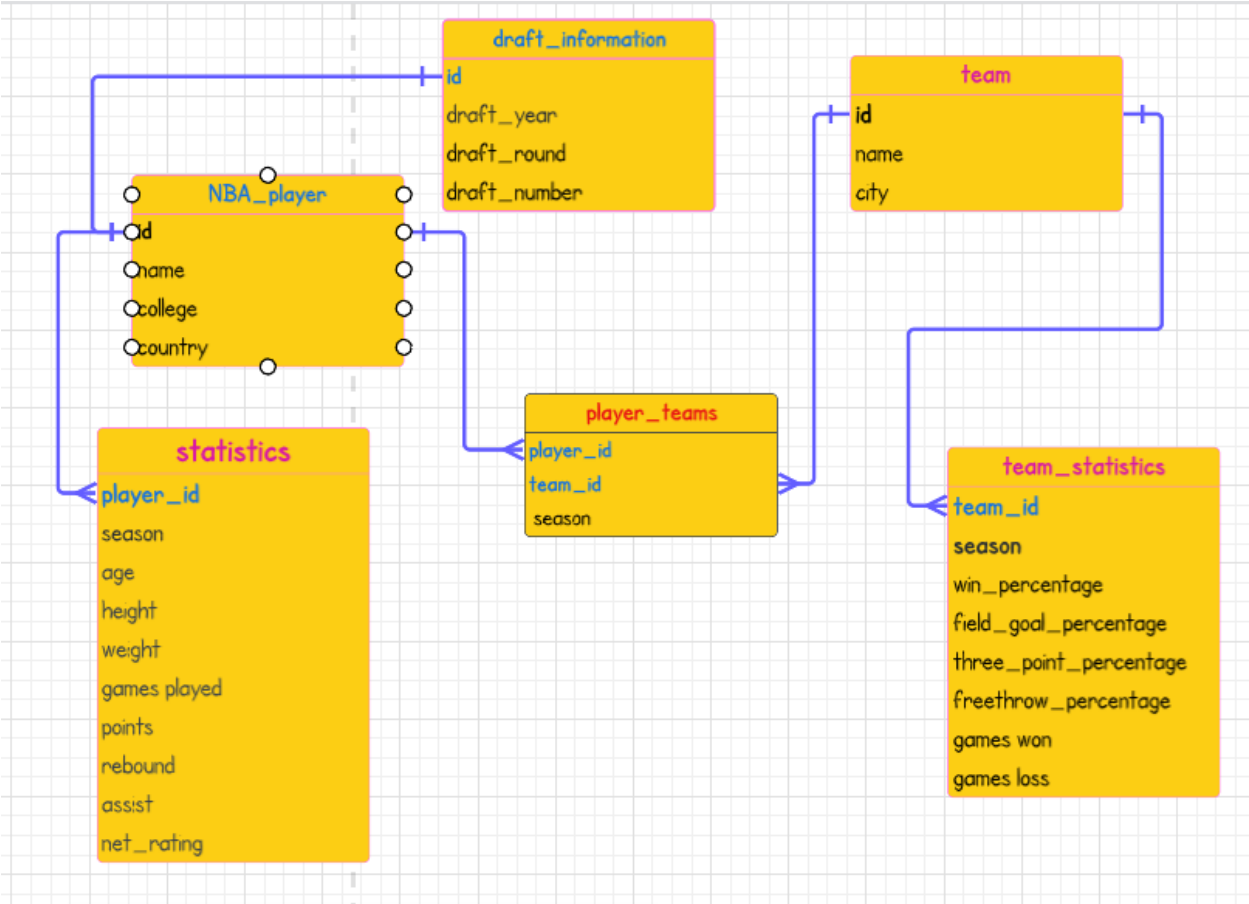
Introduction:

This project involves the search and augmentation of data. Using the data, design a database that includes three to five relations using Entity-Relationship Diagrams (ERD). Which includes one-to-one relations, many-to-many relations, one-to-many relations, and weak entity sets or ISA hierarchy. After designing the database, develop the design into a database schema and include key definitions and referential integrity constraints (foreign key). Then, identify the functional dependencies or multi-valued dependencies for the database and determine whether it is free from violations for the 3rd Normal Form, Boyce-Codd Normal Form, and 4th Normal Form. Describe queries that can be used to interact with the database, such as subqueries, aggregation, insert queries, updates queries, etc. Lastly, include information about the type of applications that will run on the designed database, as well as include the use cases, not just support application functionality.

Data Selection:

For the NBA players and teams database, data are drawn from two datasets. One dataset contains the profile information and statistics of the NBA players from the 2019 - 2020 season and the 2020 - 2021 season. The other dataset contains the profile and the statistics of a team from the 2019 - 2020 season and the 2020 - 2021 season. Specifically from the NBA players dataset, the name, season, college, country, team, age, height, weight, games played, points, rebound, assist, net_rating, draft_year, draft_round, and draft_number are selected. From the NBA teams dataset, the name, city, season, win percentage, field goal percentage, three-point percentage, free-throw percentage, games won, and games loss are selected. From these data points of the two datasets, a database is designed and visualized through the Entity-Relationship Diagrams. The mentioned data is linked in the Links section.

Database Design:



In the Entity-Relationship Diagram above, the one-to-one relation, many-to-many relations, one-to-many relations, and weak entity set are marked. (1) There is a one-to-one relation between the NBA_player entity and the draft_information weak entity. Specifically, the **id** of the draft_information weak entity is a foreign key, and the **id** of the NBA_player entity is the key that the draft_information weak entity depends on. (2) The NBA_player entity has a one-to-many relation with the statistics weak entity. Statistics weak entity has a foreign key **player_id** that depends on the **id** of the NBA_player entity. Furthermore, one NBA_player can have multiple statistics depending on the candidate key **player_id** and **season** in the statistics weak entity; therefore, the NBA_player entity is a one-to-many relation with the statistic weak entity. The team entity has a one-to-many relation with the team_statistic weak entity. The

team_id in the team_statistic weak entity depends on the key in the team entity. Furthermore, the team can have multiple team_statistic records depending on the candidate key **team_id** and **season** in the team_statistics weak entity; thus, making this relation a one-to-many relation. This Entity-Relationship Diagram also has a many-to-many relation. The NBA_player entity has a one-to-many relation with the player_teams weak entity because one unique **player_id** can have multiple **team_id** depending on the season attribute. The team entity has a one-to-many relation with the player_teams weak entity because one unique **team_id** can have many different **player_id**. As a result, the player_teams entity contains multiple unique **player_id** for a **team_id**, and a **player_id** can have multiple **team_id** depending on the season; creating a many-to-many relation. As mentioned earlier, team_statistics, player_teams, and statistics are weak entities because they can not determine unique attributes without a foreign key, or a key from another table.

Database Schema:

From the Entity-Relationship Diagram in the Database Design section, the schema NBA is created in the database. The tables NBA_player, statistics, draft_information, player_teams, team, and team_statistics are created within the NBA schema. The table NBA_player contains the primary key **id** and attributes **name**, **college**, and **country**. The statistics table contains the referential integrity constraint **player_id**, which is the primary key **id** of the NBA_player table; and attributes **season**, **age**, **height**, **weight**, **games played**, **points**, **rebound**, **assist**, and **net_rating**. The draft_information table contains the referential integrity constraint **id** that is the primary key **id** of the NBA_player table. It also has the attributes **draft_year**, **draft_round**, and **draft_number**. The team table contains the primary key **id** and attributes **name** and **city**. The team_statistics table contains a referential integrity constraint **team_id** that is the primary key **id**

of the team table. It also has attributes **season**, **win_percentage**, **field_percentage**, **three_point_percentage**, **free-throw_percentage**, **games won**, and **games loss**. The player_teams table contains two referential integrity constraints. The referential integrity constraint **player_id** is the primary key **id** of the NBA_player table. The referential integrity constraint **team_id** is the primary key **id** of the team table. The relations used in the Entity-Relationship Diagram persist in the created schema through the referential integrity constraints and primary keys of the tables. The schema and table creation SQL queries are linked in the Links section.

Normal Forms:

The functional dependencies and multi-valued dependencies of each table are tested for the 3rd Normal Form, Boyce-Codd Normal Form, and 4th Normal Form violations.

Table 1: NBA_player NBA_player = {id, name, college, country}

Functional Dependencies: id -> {name, college, country}

3rd form: No violation.

Explanation: (1) The candidate key, **id**, can uniquely determine the attributes: **name**, **college**, and **country**. (2) This relation does not contain any partial dependency and it conforms to 1st normal form.

BCNF: No violation.

Explanation: (1) The candidate key, **id,** can uniquely determine the attributes: **name**, **college**, and **country**. Since the determinant of functional dependency **id** is a key, it is

conforming to BCNF. (2) Since it is already conforming to the 3rd form, and the determinant **id** of the functional dependency is a key, it conforms to BCNF.

4th form: Violation.

Explanation: (1) The functional dependency id -> {name, college, country} is a multivalue dependency. **id ->> name**, **id ->> college**, and **id ->> country**. (2) This table conforms with case 1: conforming to BCNF. But it contains multivalue dependencies, therefore, this table is not in 4th form.

Table 2: Statistics Statistics = {player_id, season, age, height, weight, games played, points, rebound, assist, net_rating}

Functional dependencies: {player_id, season} -> {age, height, weight, games played, points, rebound, assist, net_rating}

3rd form: No violation.

Explanation: (1) The candidate key **{player_id, season}** can uniquely determine the attributes: **age**, **height**, **weight**, **games played**, **points**, **rebound**, **assist**, and **net_rating**. (2) This relation does not contain any partial dependency and it conforms to 1st normal form.

BCNF: No violation.

Explanation: (1) The candidate key **{player_id, season}** can uniquely determine the attributes: **age**, **height**, **weight**, **games played**, **points**, **rebound**, **assist**, and **net_rating**. Since the determinant of functional dependency **{player_id, season}** is a key, it is

conforming to BCNF. (2) Since it is already conforming to the 3rd form, and the determinant **{player_id, season}** of the functional dependency is a key, it conforms to BCNF.

4th form: Violation.

Explanation: (1) The functional dependency {player_id, season} -> {age, height, weight, games played, points, rebound, assist, net_rating} has multivalue dependencies. **{player_id, season} ->> age, {player_id, season} ->> height**, **{player_id, season} ->> weight**, **{player_id, season} ->> games played**, **{player_id, season} ->> points**, **{player_id, season} ->> rebound**, **{player_id, season} ->> assist**, and **{player_id, season} ->> net_rating**. (2) This table conforms with case 1: conforming to BCNF. But it contains multivalue dependencies, therefore, this table is not in 4th form.

Table 3: Team team = {id, name, city}

Functional dependencies: id -> {name, city}

3rd form: No violation

Explanation: (1) The candidate key, **id**, can uniquely determine the attributes: **name**, and **city**. (2) This relation does not contain any partial dependencies, and it conforms to 1st normal form.

BCNF: No violation

Explanation: (1) The candidate key, **id**, can uniquely determine the attributes: **name**, **city**. (2) Since the determinant **id** of the functional dependency is a key, and it conforms to the 3rd form; it conforms to BCNF.

4th form: Violation

Explanation: (1) For the functional dependency id->{name, city}, it has multivalued dependencies. **id->>name**, and **id->>city**. Multivalued dependencies are not allowed in the 4th normal form, therefore it violates the 4th form.

Table 4: Player_teams Player_teams = {player_id, team_id, season}

Function dependencies: {season,player_id} -> team_id

3rd form: No violation

Explanation: (1) The candidate key **{season, player_id}** can uniquely determine the attributes: **team_id**. (2) This relation does not contain any partial dependencies, and it conforms to 2nd normal form.

BCNF: No violation

Explanation: (1) The candidate key **{season, player_id}** can uniquely determine the **team_id**. (1) This functional dependency conforms to BCNF because the determinant **{season, player_id}** is a key; and it conforms to the 3rd normal form.

4th form: No violation

Explanation: (1) For the functional dependency{season, player_id}->team_id, it conforms to the 4th form. (2) The candidate key **{season,player_id}** only determines **team_id** and nothing else. As well as the functional dependency already conforms to BCNF.

Table 5: team_statistics team_statistics = {team_id, season, win_percentage, field_goal_percentage, three_point_percentage, freethrow_percentage, games won, games loss}

Functional dependencies: {team_id, season} -> { win_percentage, field_goal_percentage, three_point_percentage, freethrow_percentage, games won, games loss}

3rd form: No violation

Explanation: (1) The candidate key, **{team_id, season}**, can uniquely determine the attributes: **win_percentage**, **field_goal_percentage**, **three_point_percentage**, **freethrow_percentage**, **games won**, and **games loss**. (2) This relation does not contain any partial dependencies, and it conforms to 1st normal form.

BCNF: No violation

Explanation: (1) The candidate key, **{team_id, season}**, can uniquely determine the attributes: **win_percentage**, **field_goal_percentage**, **three_point_percentage**, **freethrow_percentage**, **games won**, and **games loss**. (2) Since the determinant **{team_id, season}** of the functional dependency is a key, and it conforms to the 3rd form; it conforms to BCNF.

4th form: Violation

Explanation: (1) For the functional dependency {team_id, season} -> { win_percentage, field_goal_percentage, three_point_percentage, freethrow_percentage, games won, games loss}, it has multivalued dependencies. **{team_id, season} ->> win_percentage**, **{team_id, season} ->> field_goal_percentage**, **{team_id, season} ->> three_point_percentage**, **{team_id, season} ->> freethrow_percentage, {team_id, season} ->> games won**, and **{team_id, season} ->> games loss**. (2) Despite it conforms to BCNF, Multivalued dependencies are not allowed in the 4th normal form, therefore it violates the 4th form.

Table 6: draft_information draft_information = {id, college, draft_round, draft_number, country}

Functional dependencies: id -> {draft_year, draft_round, draft_number}

3rd form: No violation

Explanation: (1) The candidate key, **id**, can uniquely determine the attributes: **draft_year**, **draft_round**, **draft_number**. (2) This relation does not contain any partial dependencies, and it conforms to 1st normal form.

BCNF: No violation

Explanation: (1) The candidate key, **id**, can uniquely determine the attributes: **draft_year**, **draft_round**, **draft_number**. (2) Since the determinant **id** of the functional dependency is a key, and it conforms to the 3rd form; it conforms to BCNF.

4th form: Violation

Explanation: (1) For the functional dependency id -> {draft_year, draft_round, draft_number}, it has multivalued dependencies. **id ->> draft_year**, **id ->> draft_round**, **id ->> draft_number**. (2) Despite it conforms to BCNF, Multivalued dependencies are not allowed in the 4th normal form, therefore it violates the 4th form.

Queries:

Many queries of use are created for this **nba** database. It includes aggregations, insert queries, subqueries, and update queries. For insert queries, inserting a player, inserting a player's statistics, inserting a player who is at least 18 years old (constraint), and inserting a non-unique player (constraint) are available for use in this database. Inserting a player requires a tuple **(player_id**, **name**, **college**, and **country)**. **player_id** is required and has to be unique from every entry that is currently in the database. The other attributes are optional and will be set to NULL if left empty. Similar to inserting a player, inserting a player's statistics requires the unique key of **player_id,** and **season** to uniquely identify a player in a particular season; other attributes (**age**, **height**, **weight**, **games played**, **points**, **rebound**, **assist**, **net_rating**) are optional and set to NULL if left empty. When inserting a player that is less than the age of 18, which is a constraint created at table creation, it will be invalidated. Similarly, for inserting a non-unique player, it will be invalidated automatically without using the *CHECK constraint* command due to **player_id** being a unique key. The constraint is checked using *CHECK constraint* command before the insertion. Test queries are provided to check if the player and statistics are inserted successfully and correctly. For update queries, updating a player's team and updating a particular statistic of a player are provided for use in this database. In order to update a player's team, the **player_id** in the NBA_player table and the **player_id** in the table player_teams are required. For updating a player's statistic, specifically, increment the rebound by one. The **player_id** is required to

identify the correct player for a particular season. The rebound incrementation is performed in the *SET* command. Again, test queries are provided to check whether the update query was successful and performed correctly.

For subqueries and aggregations, get players with greater than average height, get teams with greater than average win rate, get teams with an average height greater than the average height across all teams, and get the teams with greater than average **heights** who have a greater than average win rate. For getting players with greater than average **heights**, an aggregation *AVG()* is used inside a subquery in order to get the average **height** in the player_statistics table. Similarly, for getting teams that have a greater than average win rate, an aggregation *AVG()* is used inside a subquery to get the average of the **win_percentage** in the team_statistics table. For finding the teams with a team **height** average that is above average **height** across all the **NBA_players**. We used two *AVG()* aggregations, one inside a subquery to find the average player **height**. Another is to see if the team satisfies the condition that the average **height** of the team is greater than the average **height** across all the players. The result of this query is grouped by the **team_name** and ordered by **win_percentage** in descending order. For finding teams with greater than average **heights** that also have greater than average win rates, two aggregations, and two subqueries are used. One *AVG()* is used inside a subquery that finds the average **height** of the players in the player_statstistics table. Another *AVG()* is used inside a subquery that finds the average **win_percentage** from the team_statistics table. The result of this query is grouped by the **team_name** and ordered by **win_percentage** in descending order.

Further documentation of the queries is made inside their respective files. The mentioned queries are linked in the Links section.

Application and Use Cases:

This NBA database allows applications such as basketball-related games to determine if a player exists in the NBA during a particular season and is performing well before they decide to create this player inside the game. If it is a less-known player, then it is not worthwhile for the application to create this player inside the game and generate revenue. Also, this database allows basketball-related games to determine the in-game worth of a player based on the information stored inside the database. For example, the database stores a player's **shooting_percentage** and **net_rating** of a particular season. The application can use this information to decide the in-game value of this player based on the records stored inside the database. This database can provide functionalities beyond applications. For example, the query for getting teams with a **win_percentage** above the average **win_percentage** of a particular season (mentioned in the Queries section). This can provide analysis about which team will likely perform well in the post-season and win the championship as these data are from the regular season, before the championship games. Also, the query for getting the player with above-average **height** (mentioned in the Queries section) can provide useful analysis. In particular, this query can determine if a particular player's **height** affects his team's **win_percentage**. Similar analysis can be made using this database through small alterations to the query that finds players with above-average **height**. The query can be changed to players with above-average shooting_percentage instead by changing the **height** attribute to the **shoot_percentage** attribute. By doing so, analysis can be made about if a player with an above-average **shooting_percentage** affects his team's **win_percentage**.

Loading Collected Data Instructions:

These instructions assume that MySQL is used as the database. Some of these instructions might apply or are similar to other databases.

1. Download the csv and sql files from the Github repository.
2. Create a new connection in MySQL and run the create.sql file inside the new connection to create the schema and tables.
3. Inside the schema, right-click on the tables directory to access the table data import wizard and import the NBA_player.csv, player_statistics.csv, draft_information.csv, team.csv, team_statistics.csv, and player_teams.csv in this specific order. During the import, make sure the source columns match the destination columns.
4. Queries in the insert_and_update.sql and sub_queries.sql files can be run after importing the data.

Links:

The datasets, Entity-Relationship Diagrams, database and table creation queries, SQL files with testable queries, and documentation are inside the Github repository below.

Link: https://github.com/Venceyv/DBMS-NBAplayer