ÇANKAYA UNIVERSITY

SOFTWARE ENGINEERING DEPARTMENT

SPECIAL TOPICS IN SOFTWARE ENGINEERING

daire, grafik, logo, simge, sembol içeren bir resim

Açıklama otomatik olarak oluşturuldu

|  |  |
| --- | --- |
| Group Members | Cem Ünsal, Furkan Arslan |
| Identity Number | 202128009, 202028013 |
| Course | SENG449 |
| Project | NBA Data Mart |
| E-mail | c2128009@student.cankaya.edu.tr  c2028013@student.cankaya.edu.tr |

Table of Contents

[**Chapter 1: Introduction** 4](#_Toc186151524)

[General Information About the Chosen Organization 4](#_Toc186151525)

[Description of the Problem 4](#_Toc186151526)

[Current Situation 5](#_Toc186151527)

[**Chapter 2: Requirements Analysis** 6](#_Toc186151528)

[User Expectations 6](#_Toc186151529)

[Questions To Be Answered 6](#_Toc186151530)

[Conceptual Design of the System 7](#_Toc186151531)

[1. CSV Files 8](#_Toc186151532)

[Team.csv 8](#_Toc186151533)

[Game.csv 8](#_Toc186151534)

[Team\_history.csv 9](#_Toc186151535)

[Common\_player\_info.csv 9](#_Toc186151536)

[2. Loading CSV Data into MSSQL 10](#_Toc186151537)

[Setting Up the Database 10](#_Toc186151538)

[Importing CSV Files 11](#_Toc186151539)

[Data Validation and Verification 11](#_Toc186151540)

[3. ER Diagram 12](#_Toc186151541)

[**Chapter 3: Dimensional Design of the System** 17](#_Toc186151542)

[Dimensional Model Used 17](#_Toc186151543)

[Proposed Model 17](#_Toc186151544)

[Fact Table: Performance 18](#_Toc186151545)

[Star Model 22](#_Toc186151546)

[Logical Schema for Star Model 23](#_Toc186151547)

[Physical Schema for Star Model 24](#_Toc186151548)

[Snowflake Model 26](#_Toc186151549)

[Logical Schema for Snowflake Model 26](#_Toc186151550)

[Physical Schema for Snowflake Model 28](#_Toc186151551)

[**Chapter 4: ETL** 31](#_Toc186151552)

[ETL Operations 31](#_Toc186151553)

[**Chapter 5: Database Choice and Implementation** 36](#_Toc186151554)

[**Chapter 6: Data Analysis and Reports** 37](#_Toc186151555)

# **Chapter 1: Introduction**

### General Information About the Chosen Organization

The chosen organization is the National Basketball Association (NBA), a premier professional basketball league in North America. Comprising 30 teams (29 based in the United States and 1 in Canada), the NBA represents the pinnacle of basketball competition worldwide.

* **Headquarters:**  
  645 Fifth Avenue, New York, NY 10022, USA
* **Business Overview:**  
  The NBA is one of the major professional sports leagues in North America, renowned for organizing regular-season games, playoffs, and championship tournaments. Additionally, it provides a wealth of statistical data to enrich the experiences of fans, teams, and analysts, fostering a data-driven sports culture.

### Description of the Problem

The NBA generates and maintains extensive datasets encompassing team performance metrics such as rebounds, assists, blocks, and scores from every game. However, this valuable data is dispersed across multiple systems, creating significant challenges in conducting comprehensive performance analyses across seasons and venues.  
Teams, analysts, and management often struggle to extract actionable insights to refine strategies, evaluate trends, and support data-driven decision-making processes.

### Current Situation

Currently, the NBA relies on transactional databases to store raw game data. While these databases excel at handling data input and retrieval tasks, they fall short in supporting complex analytical queries.  
Challenges include:

* Difficulty identifying performance trends over time.
* Limited capability for comparing home versus away game results.
* Inefficiencies in conducting season-wide performance evaluations.

These limitations hinder decision-making efficiency and reduce the NBA's ability to leverage its data for strategic advancements, underscoring the need for a robust solution tailored for in-depth analysis and reporting.

# **Chapter 2: Requirements Analysis**

### User Expectations

The users of the proposed data warehouse system have high expectations for its functionality and usability. They expect the system to:

1. Centralize Data: Create a unified repository to store all historical and ongoing game performance data in a consistent format.
2. Enable Efficient Querying: Provide a high-performance infrastructure for querying, filtering, and retrieving insights from large datasets without delays.
3. Support Advanced Analytics: Facilitate comparative and trend analysis across teams, seasons, game locations, and player-level statistics.
4. Promote User-Friendly Access: Ensure intuitive access to both pre-defined queries.
5. Scalability: Accommodate growing data volumes and future expansions, such as incorporating real-time game statistics.

### Questions To Be Answered

The system should provide insights into key performance indicators, team trends, and strategic outcomes by addressing a wide array of questions, such as:

1. Team Performance Trends:
   * What is the performance comparison between home and away games?
   * How has a team's performance evolved across different seasons, in terms of points made?
   * Which teams are the most successful ones in terms of winning?
   * Which teams are the most unsuccessful ones in terms of winning?
2. Strategic Metrics:
   * Which teams have the highest amount of points, rebounds, assists, blocks, wins etc. and least amount of turnovers, missed shots, loses?
   * Which teams have the least amount of points, rebounds, assists, blocks, wins etc. and highest amount of turnovers, missed shots, loses?
3. Historical and Predictive Analysis:
   * What historical patterns predict a team's likelihood of winning next match as home team?
   * What historical patterns predict a team's likelihood of winning next match as home team?

## Conceptual Design of the System

The conceptual design establishes the foundation for constructing the Data Warehouse (DW), ensuring the data is organized and optimized for analytical queries and reporting. This process includes defining data sources, designing ER diagrams, and preparing/loading data into a MS SQL database.

## 1. CSV Files

The main data sources for this project are CSV files, which contain raw data gathered from various systems. Each CSV file corresponds to a specific entity or table represented in the ERD. The details of these files are as follows:

### Team.csv

Attributes: id,full\_name,abbreviation,nickname,city,state,year\_founded

Record Count: 30 team

Details: Contains essential information about NBA teams, including unique identifiers, official names, abbreviations, locations, and founding years.

### Game.csv

Attributes: season\_id,team\_id\_home,team\_abbreviation\_home,team\_name\_home,game\_id,game\_date,matchup\_home,wl\_home,min,fgm\_home,fga\_home,fg\_pct\_home,fg3m\_home,fg3a\_home,fg3\_pct\_home,ftm\_home,fta\_home,ft\_pct\_home,oreb\_home,dreb\_home,reb\_home,ast\_home,stl\_home,blk\_home,tov\_home,pf\_home,pts\_home,plus\_minus\_home,video\_available\_home,team\_id\_away,team\_abbreviation\_away,team\_name\_away,matchup\_away,wl\_away,fgm\_away,fga\_away,fg\_pct\_away,fg3m\_away,fg3a\_away,fg3\_pct\_away,ftm\_away,fta\_away,ft\_pct\_away,oreb\_away,dreb\_away,reb\_away,ast\_away,stl\_away,blk\_away,tov\_away,pf\_away,pts\_away,plus\_minus\_away,video\_available\_away,season\_type

Record Count: 65,642 games

Details: Contains comprehensive data for each game, including detailed performance metrics for both home and away teams. Attributes cover scoring, rebounding, assists, steals, and other critical statistics. The dataset enables in-depth analysis of team and player performances across different seasons and game types (e.g., regular season vs. playoffs).

### Team\_history.csv

Attributes: team\_id,city,nickname,year\_founded,year\_active\_till

Record Count: 52 historical team records

Details: Provides a historical perspective on NBA teams, detailing their foundational and active years. This dataset includes teams that may have relocated, changed names, or ceased operations, making it valuable for understanding the evolution of the league and its teams over time.

### Common\_player\_info.csv

Attributes: person\_id,first\_name,last\_name,display\_first\_last,display\_last\_comma\_first,display\_fi\_last,player\_slug,birthdate,school,country,last\_affiliation,height,weight,season\_exp,jersey,position,rosterstatus,games\_played\_current\_season\_flag,team\_id,team\_name,team\_abbreviation,team\_code,team\_city,playercode,from\_year,to\_year,dleague\_flag,nba\_flag,games\_played\_flag,draft\_year,draft\_round,draft\_number,greatest\_75\_flag

Record Count: 3,468 player profiles

Details: This dataset contains comprehensive information about NBA players, including their biographical details, team affiliations, career timeline, draft information, and special accolades. It is essential for analyzing player contributions, career progression, and historical comparisons within the league.

## 2. Loading CSV Data into MSSQL

The process of transferring CSV files into an MSSQL database is a crucial step for structuring the data for querying and analysis. Here are the detailed steps we followed for importing CSV files into MSSQL:

### Setting Up the Database

* Creating the Database: To begin, we created a new database in MSSQL where all the data would be stored. We simply created the database using SQL Server Management Studio.

### Importing CSV Files

* Preparation for Import: We downloaded all CSV files and placed them in a directory that is easily accessible by the MSSQL server.
* We cleaned up some of the data in CSV files. (e.g. Duplicate records, records with not enough information, integers in float format etc.)
* Instead of creating tables, we directly created them using SSMS’s built-in SQL Server Import/Export Data Wizard from CSV files.
* Importing Using SQL Server Management Studio (SSMS) Import/Export Data Wizard:
  + Select the Data Source: We specified the CSV files as the data source.
  + Select the Destination: We selected our MSSQL database (e.g., NBA\_DataWarehouse) as the destination.
  + Assign CSV types: This was a crucial part since importing a numeric value as string may cause huge problems. After using “Suggest Types” option, then finding issues from errors and correcting them, the tables have been created.

### Data Validation and Verification

After importing the CSV files into the MSSQL database, we performed data validation to ensure that the import was successful:

Checking Record Column Types: We compared the record counts between the CSV files and the corresponding tables in MSSQL to verify that all records were imported correctly. Some parts of the data had wrong type, we used analytical queries to fix them.

Data Consistency: We ran queries to check for any null or invalid data entries in key fields such as team\_id’s in game table not existing on team table etc.

## 3. ER Diagram

Entities:

GAME

* + Attributes:
    - Game\_id: Primary Key
    - Season\_id: Foreign Key to indicate the season
    - Team\_id\_home: Foreign Key referencing TEAM
    - Team\_id\_away: Foreign Key referencing TEAM
    - Game\_date: Date of the game
    - Rebound\_home: Number of rebounds for the home team
    - Rebound\_away: Number of rebounds for the away team
    - Assist\_home: Number of assists for the home team
    - Assist\_away: Number of assists for the away team
    - Steal\_home: Number of steals for the home team
    - Steal\_away: Number of steals for the away team
    - Block\_home: Number of blocks for the home team
    - Block\_away: Number of blocks for the away team
    - Turnover\_home: Number of turnovers for the home team
    - Turnover\_away: Number of turnovers for the away team
    - PFoul\_home: Personal fouls for the home team
    - PFoul\_away: Personal fouls for the away team
    - Points\_home: Points scored by the home team
    - Points\_away: Points scored by the away team
    - Plus\_Minus\_Home: Plus-minus value for the home team
    - Plus\_Minus\_Away: Plus-minus value for the away team
    - Minutes: Game duration in minutes
    - WL\_home: Win/loss indicator for the home team
    - WL\_away: Win/loss indicator for the away team
    - Various shooting metrics for each team (e.g., FGM\_home, FGA\_home, etc.)
    - Season\_type: Type of season (e.g., regular, playoffs)
  + Relationships:
    - PLAY MATCHES: Relationships with the TEAM table through Team\_id\_home and Team\_id\_away.

TEAM

* + Attributes:
    - Id: Primary Key
    - Full\_name: Full name of the team
    - Abbreviation: Abbreviated team name
    - Nickname: Team nickname
    - City: City of the team
    - State: State of the team
    - Year\_founded: Year the team was founded
  + Relationships:
    - HAS: Relationship with TEAM\_HISTORY table through Team\_id.
    - PLAYS AT: Relationship with the PLAYER table through Team\_id.
    - PLAY MATCHES: Relationships with the GAME table through Team\_id\_home and Team\_id\_away.

TEAM\_HISTORY

* + Attributes:
    - Team\_id: Primary Key and Foreign Key referencing TEAM
    - City: City of the team during a historical period
    - Nickname: Nickname of the team during a historical period
    - Year\_founded: Year the team was founded
    - Year\_active\_till: Year the team was active till
  + Relationships:
    - HAS: Relationship with TEAM.

PLAYER

* + Attributes:
    - Person\_id: Primary Key
    - First\_name: Player’s first name
    - Last\_name: Player’s last name
    - Birthdate: Player’s birthdate
    - Country: Player’s country
    - Height: Player’s height
    - Weight: Player’s weight
    - Jersey: Jersey number
    - Position: Playing position
    - Team\_id: Foreign Key referencing TEAM
    - Team\_name: Team name
    - Various additional details (e.g., Last\_First, Player\_Slug, Season\_exp, etc.)
    - Draft details (e.g., Draft\_Year, Draft\_Round, etc.)
    - Flags (e.g., DLeague\_Flag, NBA\_Flag)
  + Relationships:
    - PLAYS AT: Relationship with TEAM through Team\_id.

metin, diyagram, ekran görüntüsü, paralel içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 1 Conceptual Model of the Operational System

# **Chapter 3: Dimensional Design of the System**

## Dimensional Model Used

We are using Star Schema for the dimensional design. Snowflake Schema is also checked in case it would have a better performance. However, Star Schema was better in terms of simplicity.

* Star Schema is used for simplicity and fast query performance, where the fact table is connected to dimension tables.
* Snowflake Schema could be used for normalizing complex dimension tables to reduce redundancy.

## Proposed Model

1. Fact Table:
   * Performance: Tracks game performance metrics for both home and away teams, such as points, rebounds, assists, etc. Also tracks winstreaks and losestreaks of home team and away team.
2. Dimension Tables:
   * Team Dimension: Contains team details like team name, abbreviation, city, and state.
   * Game Dimension: Captures common game data such as game ID, season ID, matchup, season type.
   * Date Dimension: Tracks information related to the date such as day, month, year.

This model helps efficiently store and query data while maintaining normalization for optimal performance.

### Fact Table: Performance

* Columns:
  + Rebound\_home
  + Rebound\_away
  + Assist\_home
  + Assist\_away
  + Steal\_home
  + Steal\_away
  + Block\_home
  + Block\_away
  + Turnover\_home
  + Turnover\_away
  + PFoul\_home
  + PFoul\_away
  + Points\_home
  + Points\_away
  + Plus\_Minus\_Home
  + Plus\_Minus\_Away
  + WL\_home
  + WL\_away
  + Minutes
  + FGM\_home, FGM\_away (Field Goals Made)
  + FGA\_home, FGA\_away (Field Goals Attempted)
  + FG\_pct\_home, FG\_pct\_away (Field Goal Percentage)
  + FG3M\_home, FG3M\_away (3-Point Goals Made)
  + FG3A\_home, FG3A\_away (3-Point Goals Attempted)
  + FG3\_pct\_home, FG3\_pct\_away (3-Point Goal Percentage)
  + FTM\_home, FTM\_away (Free Throws Made)
  + FTA\_home, FTA\_away (Free Throws Attempted)
  + FT\_pct\_home, FT\_pct\_away (Free Throw Percentage)
  + OReb\_home, OReb\_away (Offensive Rebounds)
  + DReb\_home, DReb\_away (Defensive Rebounds)
  + Home\_perf, Away\_perf
  + Home\_winstreak, Away\_winstreak
  + Home\_losestreak, Away\_losestreak

Dimension Tables

1. Team Dimension
   * Columns:
     + Id (Primary Key)
     + Full\_name: Full name of the team
     + Abbreviation: Team abbreviation
     + Nickname: Team nickname
     + City: Team’s city
     + State: Team’s state
     + Year\_founded: Year the team was founded
2. Date Dimension
   * Columns:
     + Date\_id (Primary Key)
     + Year: Year of the game
     + Month: Month of the game
     + Day: Day of the game
3. Game Dimension
   * Columns:
     + Game\_id (Primary Key)
     + Season\_id: Season identifier
     + Matchup: Teams playing the game
     + Season\_type: Type of season (e.g., regular, playoffs)

Relationships

1. R1: Performance.Game\_id → Game.Game\_id
2. R2: Performance.Date\_id → Date.Date\_id
3. Home Relationship: Performance.Home → Team.Id
4. Away Relationship: Performance.Away → Team.Id

## Star Model

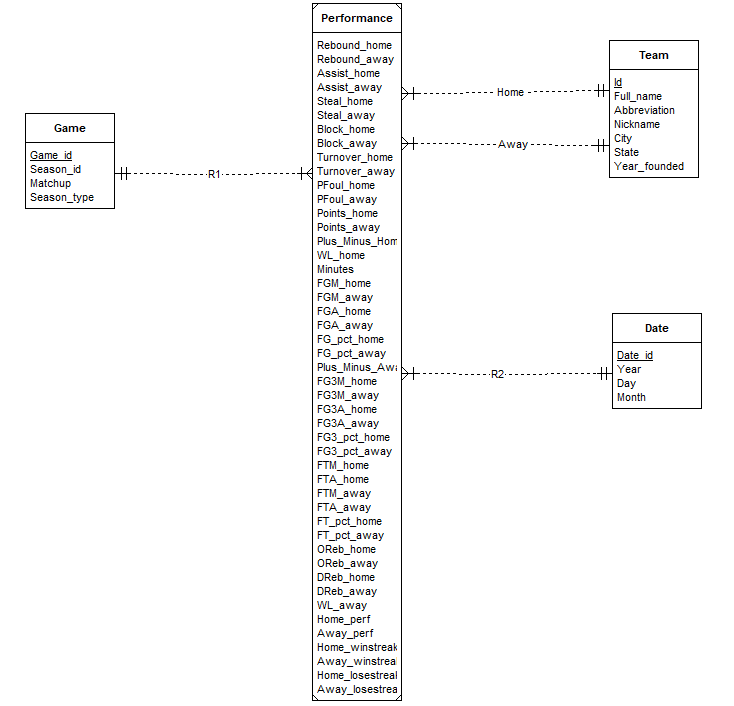


Figure 2 Star Shema for Proposal Data Mart

### Logical Schema for Star Model

1. GAME  
   (Game\_id, Season\_id, Matchup, Season\_type)
2. TEAM  
   (id, Full\_name, Abbreviation, Nickname, City, State, Year\_founded)
3. DATE  
   (Date\_id, Day, Month, Year)
4. PERFORMANCE  
   (Game\_id, Date\_id, Home\_team\_id, Away\_team\_id, Rebound\_home, Rebound\_away, Assist\_home, Assist\_away, Steal\_home, Steal\_away, Block\_home, Block\_away, Turnover\_home, Turnover\_away, PFoul\_home, PFoul\_away, Points\_home, Points\_away, Plus\_Minus\_Home, Plus\_Minus\_Away, WL\_home, WL\_away, FGM\_home, FGM\_away, FGA\_home, FGA\_away, FG\_pct\_home, FG\_pct\_away, FG3M\_home, FG3M\_away, FG3A\_home, FG3A\_away, FG3\_pct\_home, FG3\_pct\_away, FTM\_home, FTM\_away, FTA\_home, FTA\_away, FT\_pct\_home, FT\_pct\_away, OReb\_home, OReb\_away, DReb\_home, DReb\_away, Home\_perf, Away\_perf, Home\_winstreak, Away\_winstreak, Home\_losestreak, Away\_losestreak)
   * Foreign Keys:
     + Game\_id FK refers to GAME
     + Date\_id FK refers to DATE
     + Home\_team\_id and Away\_team\_id FKs refer to TEAM

### Physical Schema for Star Model

CREATE TABLE Team

(id int PRIMARY KEY,full\_name VARCHAR(22),abbreviation VARCHAR(3),nickname VARCHAR(13),city VARCHAR(13),[state] VARCHAR(20),year\_founded SMALLINT);

CREATE TABLE [Date]

(date\_id INT PRIMARY KEY,[day] SMALLINT,[month] SMALLINT,[year] SMALLINT);

CREATE TABLE Game

(game\_id INT PRIMARY KEY,season\_id INT,matchup VARCHAR(9),season\_type VARCHAR(14));

CREATE TABLE Performance

(home\_team\_id INT FOREIGN KEY REFERENCES Team(id),away\_team\_id INT FOREIGN KEY REFERENCES Team(id),game\_id INT FOREIGN KEY REFERENCES Game(game\_id),date\_id INT FOREIGN KEY REFERENCES [Date](date\_id),wl\_home VARCHAR(1),[min] SMALLINT,fgm\_home SMALLINT,fga\_home SMALLINT,fg\_pct\_home FLOAT,fg3m\_home SMALLINT,fg3a\_home SMALLINT,fg3\_pct\_home FLOAT,ftm\_home SMALLINT,fta\_home SMALLINT,ft\_pct\_home FLOAT, oreb\_home SMALLINT,dreb\_home SMALLINT,reb\_home SMALLINT,ast\_home SMALLINT,stl\_home SMALLINT,blk\_home SMALLINT,tov\_home SMALLINT,pf\_home SMALLINT,pts\_home SMALLINT,plus\_minus\_home SMALLINT,wl\_away VARCHAR(1),fgm\_away SMALLINT,fga\_away SMALLINT,fg\_pct\_away FLOAT,fg3m\_away SMALLINT,fg3a\_away SMALLINT,fg3\_pct\_away FLOAT,ftm\_away SMALLINT,fta\_away SMALLINT,ft\_pct\_away FLOAT, oreb\_away FLOAT,dreb\_away SMALLINT,reb\_away SMALLINT,ast\_away SMALLINT,stl\_away SMALLINT,blk\_away SMALLINT,tov\_away SMALLINT,pf\_away SMALLINT,pts\_away SMALLINT,plus\_minus\_away SMALLINT,home\_performance INT,away\_performance INT,home\_winstreak INT,away\_winstreak INT,home\_losestreak INT,away\_losestreak INT);

## Snowflake Model

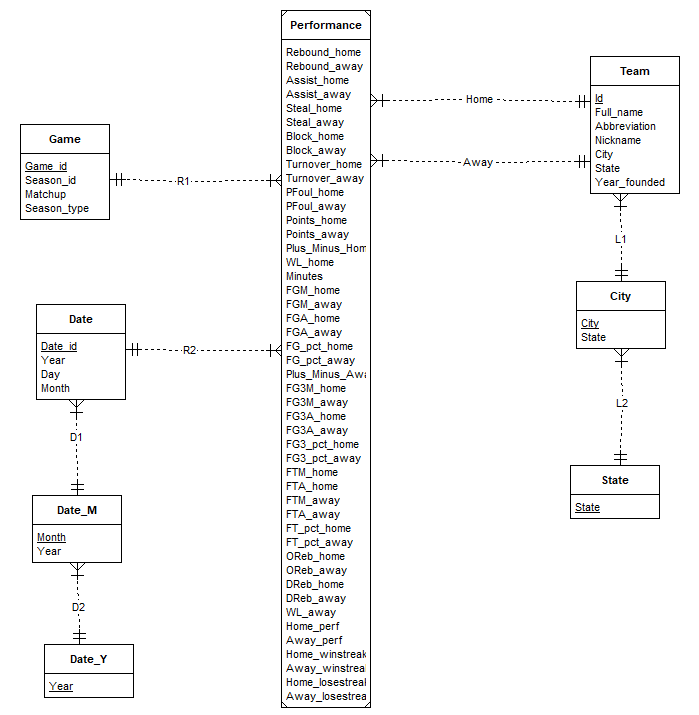


Figure 3 Snowflake Shema for Proposal Data Mart

### Logical Schema for Snowflake Model

1. GAME  
   (Game\_id, Season\_id, Matchup, Season\_type)
2. TEAM  
   (Team\_id, Full\_name, Abbreviation, Nickname, City, Year\_founded)
   * Foreign Keys:
     + City FK refers to STATE
3. CITY  
   (City, State)
   * Foreign Keys:
     + State FK refers to STATE
4. STATE  
   (State)
5. DATE  
   (Date\_id, Day, Month, Year)
   * Foreign Keys:
     + Month FK refers to DATE\_M
6. DATE\_M (Month Dimension)  
   (Month, Year)
   * Foreign Keys:
     + Year FK refers to DATE\_Y
7. DATE\_Y (Year Dimension)  
   (Year)
8. PERFORMANCE  
   (Game\_id, Date\_id, Home\_team\_id, Away\_team\_id, Rebound\_home, Rebound\_away, Assist\_home, Assist\_away, Steal\_home, Steal\_away, Block\_home, Block\_away, Turnover\_home, Turnover\_away, PFoul\_home, PFoul\_away, Points\_home, Points\_away, Plus\_Minus\_Home, Plus\_Minus\_Away, WL\_home, WL\_away, FGM\_home, FGM\_away, FGA\_home, FGA\_away, FG\_pct\_home, FG\_pct\_away, FG3M\_home, FG3M\_away, FG3A\_home, FG3A\_away, FG3\_pct\_home, FG3\_pct\_away, FTM\_home, FTM\_away, FTA\_home, FTA\_away, FT\_pct\_home, FT\_pct\_away, OReb\_home, OReb\_away, DReb\_home, DReb\_away, Home\_perf, Away\_perf, Home\_winstreak, Away\_winstreak, Home\_losestreak, Away\_losestreak)
   * Foreign Keys:
     + Game\_id FK refers to GAME
     + Date\_id FK refers to DATE
     + Home\_team\_id and Away\_team\_id FKs refer to TEAM

### Physical Schema for Snowflake Model

1. CREATE TABLE GAME

(game\_id INT PRIMARY KEY,season\_id INT,matchup VARCHAR(9),season\_type VARCHAR(14));

1. CREATE TABLE TEAM

(id int PRIMARY KEY,full\_name VARCHAR(22),abbreviation VARCHAR(3),nickname VARCHAR(13),city VARCHAR(13) FOREIGN KEY REFERENCES CITY(City\_id),[state] VARCHAR(20),year\_founded SMALLINT);

1. CREATE TABLE CITY

(City VARCHAR(13) PRIMARY KEY,[State] VARCHAR(20) FOREIGN KEY REFERENCES STATE([State]));

1. CREATE TABLE [STATE]

([State] VARCHAR(20) PRIMARY KEY);

1. CREATE TABLE [DATE]

(date\_id INT PRIMARY KEY,[day] SMALLINT,[month] SMALLINT FOREIGN KEY REFERENCES DATE\_M(Month),[year] SMALLINT);

1. CREATE TABLE DATE\_M (Month Dimension)

([Month] INT PRIMARY KEY,[Year] INT FOREIGN KEY REFERENCES DATE\_Y(Year));

1. CREATE TABLE DATE\_Y (Year Dimension)

([Year] INT PRIMARY KEY);

1. CREATE TABLE PERFORMANCE

(home\_team\_id INT FOREIGN KEY REFERENCES Team(id),away\_team\_id INT FOREIGN KEY REFERENCES Team(id),game\_id INT FOREIGN KEY REFERENCES Game(game\_id),date\_id INT FOREIGN KEY REFERENCES [Date](date\_id),wl\_home VARCHAR(1),[min] SMALLINT,fgm\_home SMALLINT,fga\_home SMALLINT,fg\_pct\_home FLOAT,fg3m\_home SMALLINT,fg3a\_home SMALLINT,fg3\_pct\_home FLOAT,ftm\_home SMALLINT,fta\_home SMALLINT,ft\_pct\_home FLOAT,oreb\_home SMALLINT,dreb\_home SMALLINT,reb\_home SMALLINT,ast\_home SMALLINT,stl\_home SMALLINT,blk\_home SMALLINT,tov\_home SMALLINT,pf\_home SMALLINT,pts\_home SMALLINT,plus\_minus\_home SMALLINT,wl\_away VARCHAR(1),fgm\_away SMALLINT,fga\_away SMALLINT,fg\_pct\_away FLOAT,fg3m\_away SMALLINT,fg3a\_away SMALLINT,fg3\_pct\_away FLOAT,ftm\_away SMALLINT,fta\_away SMALLINT,ft\_pct\_away FLOAT,oreb\_away FLOAT,dreb\_away SMALLINT,reb\_away SMALLINT,ast\_away SMALLINT,stl\_away SMALLINT,blk\_away SMALLINT,tov\_away SMALLINT,pf\_away SMALLINT,pts\_away SMALLINT,plus\_minus\_away SMALLINT,home\_performance INT,away\_performance INT,home\_winstreak INT,away\_winstreak INT,home\_losestreak INT,away\_losestreak INT);

# **Chapter 4: ETL**

This section will explain the ETL operations, their descriptions and how we created a data mart from a database. First of all, in our project, we selected star schema as it is a better option for our cause. The reasons are:

* Better Query Performance: Snowflake has much more joins than Star schema. This makes queries much complex and longer than wanted. So star schema will be a better choice.
* Scalability: Even though Snowflake’s non-redundant structures makes a project better in terms of storage, adding new records and making changes in tables will require much more effort. Star schema has less efficiency in terms of storage. However, since we do not have problems with storage but
* Much More Simple: Snowflake’s hierarchical structure makes the project a hassle. Star schema is very straightforward to implement, maintain, and query.

## ETL Operations

First ETL Operation (449\_NBA to 449\_STAGING)

This part of ETL is about the initial transfer from database to staging area. The cause of staging area is to transfer the data into data mart cleaner. First of all we created the tables in star schema. The queries for creating tables in star schema are shown above in chapter 3.

1. This query inserts Team table in 449\_NBA into Team table in 449\_STAGING. All of the previous data is inserted without any changes.

INSERT INTO [449\_STAGING].dbo.Team(id,full\_name,abbreviation,nickname,city,

[state],year\_founded)

SELECT id,full\_name,abbreviation,nickname,city,[state],year\_founded

FROM [449\_NBA].dbo.team;

1. This query inserts some of the attributes of Game table in 449\_NBA into Game table in 449\_STAGING. All of the previous data except all the measures in fact table Performance (e.g. Points\_home, Steals\_home etc.) and date information.

INSERT INTO [449\_STAGING].dbo.Game(game\_id,season\_id,matchup,season\_type)

SELECT game\_id,season\_id,matchup\_home,season\_type

FROM [449\_NBA].dbo.game;

1. This query inserts data of dates in Game table in 449\_NBA into Date table in 449\_STAGING. Date data here are converted into integers as seen below to be more readable and flexible in queries.

INSERT INTO [449\_STAGING].dbo.[Date](date\_id,[day],[month],[year])

SELECT game\_id,DATEPART(day,game\_date),DATEPART(month,game\_date),DATEPART(year,game\_date)

FROM [449\_NBA].dbo.game;

1. The fact table Performance has 2 different parts. The measure data that comes from Game table in 449\_NBA database (points, steals, blocks, percentages etc.), and the measures that are created from these (performance, winstreaks, losestreaks.) Below is the first part:

INSERT INTO Performance(home\_team\_id,away\_team\_id,game\_id,date\_id,wl\_home,[min],

fgm\_home,fga\_home,fg\_pct\_home,fg3m\_home,fg3a\_home,

fg3\_pct\_home,ftm\_home,fta\_home,ft\_pct\_home,oreb\_home,dreb\_home,reb\_home,

ast\_home,stl\_home,blk\_home,tov\_home,pf\_home,pts\_home,plus\_minus\_home,wl\_away,

fgm\_away,fga\_away,fg\_pct\_away,fg3m\_away,fg3a\_away,fg3\_pct\_away,ftm\_away, fta\_away,ft\_pct\_away,oreb\_away,dreb\_away,reb\_away,ast\_away,stl\_away,blk\_away,tov\_away,pf\_away,pts\_away,plus\_minus\_away,home\_performance,away\_performance,home\_winstreak,home\_losestreak,away\_winstreak, away\_losestreak)

SELECT team\_id\_home,team\_id\_away,g.game\_id,g.game\_id,wl\_home,[min],

fgm\_home,fga\_home,fg\_pct\_home,fg3m\_home,fg3a\_home,

fg3\_pct\_home,ftm\_home,fta\_home,ft\_pct\_home,oreb\_home,dreb\_home,reb\_home,

ast\_home,stl\_home,blk\_home,tov\_home,pf\_home,pts\_home,plus\_minus\_home,wl\_away,

fgm\_away,fga\_away,fg\_pct\_away,fg3m\_away,fg3a\_away,fg3\_pct\_away,ftm\_away,

fta\_away,ft\_pct\_away,oreb\_away,dreb\_away,reb\_away,ast\_away,stl\_away,blk\_away,tov\_away,pf\_away,pts\_away,plus\_minus\_away,

* All the attributes from select text to here, are taken directly from Game table in 449\_NBA. For the second part, performance measure is calculated by (Points scored + Rebounds + Assists + Steals + Blocks) - (Missed Field Goals + Missed Free Throws + Turnovers) \* (1.1 for winning matches, 0.9 for losing matches). Given below:

-- Home Performance

((pts\_home + (oreb\_home + dreb\_home) + ast\_home + stl\_home + blk\_home)

- ((fga\_home - fgm\_home) + (fta\_home - ftm\_home) + tov\_home))

\* CASE WHEN wl\_home = 'W' THEN 1.1 ELSE 0.9

END,

-- Away Performance

((pts\_away + (oreb\_away + dreb\_away) + ast\_away + stl\_away + blk\_away)

- ((fga\_away - fgm\_away) + (fta\_away - ftm\_away) + tov\_away))

\* CASE WHEN wl\_away = 'W' THEN 1.1 ELSE 0.9

END,

* For winstreaks and losestreaks, very hard queries are done to extract it. To summarize, program creates a table including all the matches twice, one for home team, one for away team, and their winstreak and losestreak. Winning team’s winstreak is increased by 1 and losestreak is set to 0; losing team’s winstreak is set to 0, losestreak is increased by 1. Looks at previous matches to increase these values. This query took about 18 minutes to complete.

IF OBJECT\_ID('tempdb..#Streaks') IS NOT NULL

DROP TABLE #Streaks;

CREATE TABLE #Streaks (

team\_id INT,

game\_id INT,

date\_id INT,

result CHAR(1),

winstreak INT DEFAULT 0,

losestreak INT DEFAULT 0

);

-- 2. Maçları sıralı şekilde #Streaks tablosuna ekle

INSERT INTO #Streaks (team\_id, game\_id, date\_id, result)

SELECT

team\_id,

game\_id,

date\_id,

result

FROM (

SELECT

home\_team\_id AS team\_id,

game\_id,

date\_id,

wl\_home AS result

FROM Performance

UNION ALL

SELECT

away\_team\_id AS team\_id,

game\_id,

date\_id,

wl\_away AS result

FROM Performance

) AS CombinedResults

ORDER BY team\_id, date\_id;

-- 3. Winstreak ve Losestreak değerlerini hesapla

DECLARE @team\_id INT, @game\_id INT, @date\_id INT, @result CHAR(1);

DECLARE @prev\_winstreak INT = 0, @prev\_losestreak INT = 0, @prev\_team\_id INT = NULL;

DECLARE streak\_cursor CURSOR FOR

SELECT team\_id, game\_id, date\_id, result

FROM #Streaks

ORDER BY team\_id, date\_id;

OPEN streak\_cursor;

FETCH NEXT FROM streak\_cursor INTO @team\_id, @game\_id, @date\_id, @result;

WHILE @@FETCH\_STATUS = 0

BEGIN

IF @team\_id <> @prev\_team\_id

BEGIN

-- Takım değiştiyse serileri sıfırla

SET @prev\_winstreak = 0;

SET @prev\_losestreak = 0;

END;

-- Winstreak ve Losestreak hesaplama

IF @result = 'W'

BEGIN

SET @prev\_winstreak = @prev\_winstreak + 1;

SET @prev\_losestreak = 0;

END

ELSE IF @result = 'L'

BEGIN

SET @prev\_losestreak = @prev\_losestreak + 1;

SET @prev\_winstreak = 0;

END;

-- Güncellenmiş değerleri tabloya yaz

UPDATE #Streaks

SET winstreak = @prev\_winstreak,

losestreak = @prev\_losestreak

WHERE team\_id = @team\_id AND game\_id = @game\_id;

-- Önceki takım ID'sini güncelle

SET @prev\_team\_id = @team\_id;

FETCH NEXT FROM streak\_cursor INTO @team\_id, @game\_id, @date\_id, @result;

END;

CLOSE streak\_cursor;

DEALLOCATE streak\_cursor;

* After these functions, temporary streaks table is left joined (only for selection part) to Game table to fill winstreak and losestreak information. Last part of insertion after away performance’s end is given below:

END,

-- Home Winstreak and Losestreak

hs.winstreak,

hs.losestreak,

-- Away Winstreak and Losestreak

as\_.winstreak,

as\_.losestreak

FROM

[449\_NBA].dbo.game g

LEFT JOIN #Streaks hs

ON g.team\_id\_home = hs.team\_id AND g.game\_id = hs.game\_id

LEFT JOIN #Streaks as\_

ON g.team\_id\_away = as\_.team\_id AND g.game\_id = as\_.game\_id;

-- Geçici tabloyu sil

DROP TABLE #Streaks;

* With this, last part of first ETL is completed. All the wanted data for data warehouse is extracted, transformed and extracted successfully.

Second ETL Operation (449\_STAGING to 449\_DW)

This part of ETL is about the transfer from cleaned staging area to data warehouse (mart). With the cleaned data, all of the data from staging area is extracted and loaded without any changes. The queries are given below:

INSERT INTO [449\_DW].dbo.Team(id,full\_name,abbreviation,nickname,city,[state],year\_founded)

SELECT id,full\_name,abbreviation,nickname,city,[state],year\_founded

FROM [449\_STAGING].dbo.Team;

INSERT INTO [449\_DW].dbo.Game(game\_id,season\_id,matchup,season\_type)

SELECT game\_id,season\_id,matchup,season\_type

FROM [449\_STAGING].dbo.Game;

INSERT INTO [449\_DW].dbo.[Date](date\_id,[day],[month],[year])

SELECT date\_id,[day],[month],[year]

FROM [449\_STAGING].dbo.[Date];

INSERT INTO Performance(home\_team\_id,away\_team\_id,game\_id,date\_id,wl\_home,[min],

fgm\_home,fga\_home,fg\_pct\_home,fg3m\_home,fg3a\_home,fg3\_pct\_home,ftm\_home,fta\_home,ft\_pct\_home,oreb\_home,dreb\_home,reb\_home,ast\_home,stl\_home,blk\_home,tov\_home,pf\_home,pts\_home,plus\_minus\_home,wl\_away,fgm\_away,fga\_away,fg\_pct\_away,fg3m\_away,fg3a\_away,fg3\_pct\_away,ftm\_away,fta\_away,ft\_pct\_away,oreb\_away,dreb\_away,reb\_away,ast\_away,stl\_away,blk\_away,tov\_away,pf\_away,pts\_away,plus\_minus\_away,home\_performance,away\_performance,home\_winstreak,home\_losestreak,away\_winstreak, away\_losestreak)

SELECT home\_team\_id,away\_team\_id,game\_id,date\_id,wl\_home,[min],

fgm\_home,fga\_home,fg\_pct\_home,fg3m\_home,fg3a\_home,fg3\_pct\_home,ftm\_home,fta\_home,ft\_pct\_home,oreb\_home,dreb\_home,reb\_home,ast\_home,stl\_home,blk\_home,tov\_home,pf\_home,pts\_home,plus\_minus\_home,wl\_away,fgm\_away,fga\_away,fg\_pct\_away,fg3m\_away,fg3a\_away,fg3\_pct\_away,ftm\_away,fta\_away,ft\_pct\_away,oreb\_away,dreb\_away,reb\_away,ast\_away,stl\_away,blk\_away,tov\_away,pf\_away,pts\_away,plus\_minus\_away,home\_performance,away\_performance,home\_winstreak,home\_losestreak,away\_winstreak,away\_losestreak

FROM [449\_STAGING].dbo.Performance;

# **Chapter 5: Database Choice and Implementation**

In this project, we chose MSSQL Server as the RDBMS for our NBA database due to its advanced features and compatibility with our system's requirements. MSSQL Server offers superior performance, reliability, and extensive support for complex queries and transactions.

Implementation of the Developed System in MSSQL Server

Our NBA data management system is implemented in MSSQL Server, leveraging its high performance, security features, and scalability. MSSQL Server’s advanced transaction management and robust support for relational data models made it an ideal choice for efficiently handling large volumes of data.

Designed Tables with Constraints

We meticulously designed tables to represent various entities within the NBA database, incorporating relationships and constraints to maintain data integrity. This ensures that our data is accurate, consistent, and easily accessible for complex queries and analytics.

* Game: Stores information about NBA matches including box scores, dates etc.
* Team: Stores data about NBA teams.
* Common Player Info: Contains common data for individual players.
* Team\_History: Stores information about teams that are changed their names but with same id (e.g. Brooklyn Nets was New Jersey Nets).

# **Chapter 6: Data Analysis and Reports**

In this project, our most important goal was to get the data to a point that it can be analyzed easily with queries. Using this, decision makers such as team owners, players, analysts, coaches can make decision depending on these statistical information. These reports can be done from web application.

Team Performance Analysis: All of the metrics in game are calculated to find performance metric. Using this the decision maker can determine the team’s current performance compared to previous performances. Also winstreaks and losestreaks can be examined to determine if the team is likely to win the next game or not.

A screenshot of a computer

Description automatically generated

Figure 4 Query List for Proposal Data Mart

A screenshot of a computer

Description automatically generated

Figure 5 Selected Query Example for Proposal Data Mart

A screenshot of a computer

Description automatically generated

Figure 6 Output of Query for Proposal Data Mart

As examples given above there are 15 different queries in current state. Below are the details of the queries.

--Sonraki maçları kazanma ihtimali yüksek olan ev sahibi takımlar hangileri?

SELECT DISTINCT t.id, t.full\_name AS team\_name, p.home\_winstreak

FROM Performance p

JOIN Team t ON t.id = p.home\_team\_id

WHERE p.home\_winstreak >= 3

ORDER BY p.home\_winstreak desc;

--Sonraki maçları kazanma ihtimali yüksek olan deplasman takımları hangileri?

SELECT DISTINCT t.id, t.full\_name AS team\_name, p.away\_winstreak

FROM Performance p

JOIN Team t ON t.id = p.away\_team\_id

WHERE p.away\_winstreak >= 3

ORDER BY p.away\_winstreak desc;

--En yüksek toplam skoru yapan takımlar hangileri?

SELECT TOP 50

t1.full\_name AS home\_team,

t2.full\_name AS away\_team,

(p.pts\_home + p.pts\_away) AS total\_score

FROM Performance p

JOIN Team t1 ON t1.id = p.home\_team\_id

JOIN Team t2 ON t2.id = p.away\_team\_id

ORDER BY total\_score DESC;

--2010'dan sonra En iyi 3 sayı yüzdesine sahip takımlar hangileri?

SELECT TOP 5

t.full\_name AS team\_name,

AVG(CAST(p.fg3\_pct\_home AS FLOAT)) AS avg\_fg3\_pct

FROM Performance p

JOIN Team t ON t.id = p.home\_team\_id

JOIN [Date] d ON d.date\_id = p.date\_id

WHERE (d.[year]>2010)

GROUP BY t.full\_name

ORDER BY avg\_fg3\_pct DESC;

--En fazla galibiyet serisi olan takımlar hangileri?

SELECT TOP 5

team\_name,

MAX(winstreak) AS max\_winstreak

FROM (

SELECT t.full\_name AS team\_name, p.home\_winstreak AS winstreak

FROM Performance p

JOIN Team t ON t.id = p.home\_team\_id

UNION ALL

SELECT t.full\_name AS team\_name, p.away\_winstreak AS winstreak

FROM Performance p

JOIN Team t ON t.id = p.away\_team\_id

) AS streaks

GROUP BY team\_name

ORDER BY max\_winstreak DESC;

--Maçlarda en fazla asist yapan takım hangisi?

SELECT TOP 5

t.full\_name AS team\_name,

SUM(p.ast\_home) AS total\_assists

FROM Performance p

JOIN Team t ON t.id = p.home\_team\_id

GROUP BY t.full\_name

ORDER BY total\_assists DESC;

--Maç kazanmada ev sahibi avantajı ne kadar önemli?

SELECT

CAST(100.0 \* SUM(CASE WHEN p.wl\_home = 'W' THEN 1 ELSE 0 END) / COUNT(\*) AS DECIMAL(5, 2)) AS home\_win\_percentage

FROM Performance p;

--Son 5 maçta performansı düşen ev sahibi takımlar hangileri?

SELECT DISTINCT t.full\_name AS team\_name, p.home\_losestreak

FROM Performance p

JOIN Team t ON t.id = p.home\_team\_id

WHERE p.home\_losestreak >= 5

ORDER BY p.home\_losestreak DESC;

--Son 5 maçta performansı düşen deplasman takımları hangileri?

SELECT DISTINCT t.full\_name AS team\_name, p.away\_losestreak

FROM Performance p

JOIN Team t ON t.id = p.away\_team\_id

WHERE p.away\_losestreak >= 5

ORDER BY p.away\_losestreak DESC;

--(Bir sezondaki) en skorer takım hangisi?

SELECT TOP 5

t.full\_name AS team\_name,

SUM(p.pts\_home) AS total\_points

FROM Performance p

JOIN Team t ON t.id = p.home\_team\_id

JOIN Game g ON g.game\_id = p.game\_id

WHERE g.season\_id = 2010 -- Örnek sezon

GROUP BY t.full\_name

ORDER BY total\_points DESC;

--Tarihteki en yüksek farkla kazanılan maçlar hangileri?

SELECT TOP 5

t1.full\_name AS home\_team,

p.wl\_home as home\_winlose,

t2.full\_name AS away\_team,

p.wl\_away as away\_winlose,

ABS(p.plus\_minus\_home) AS score\_difference,

g.matchup,

d.[day],

d.[month],

d.[year]

FROM Performance p

JOIN Team t1 ON t1.id = p.home\_team\_id

JOIN Team t2 ON t2.id = p.away\_team\_id

JOIN Game g ON g.game\_id = p.game\_id

JOIN [Date] d ON d.date\_id = p.date\_id

ORDER BY score\_difference DESC;

--Home İken Ortalama Performansı En Yüksek Takımlar

SELECT TOP 10

t.full\_name AS team\_name,

AVG(p.home\_performance) AS avg\_home\_performance

FROM Performance p

JOIN Team t ON t.id = p.home\_team\_id

GROUP BY t.full\_name

ORDER BY avg\_home\_performance DESC;

--Away İken Ortalama Performansı En Yüksek 10 Takım

SELECT TOP 10

t.full\_name AS team\_name,

AVG(p.away\_performance) AS avg\_away\_performance

FROM Performance p

JOIN Team t ON t.id = p.away\_team\_id

GROUP BY t.full\_name

ORDER BY avg\_away\_performance DESC;

--Home İken Ortalama Performansı En Düşük 10 Takım

SELECT TOP 10

t.full\_name AS team\_name,

AVG(p.home\_performance) AS avg\_home\_performance

FROM Performance p

JOIN Team t ON t.id = p.home\_team\_id

GROUP BY t.full\_name

ORDER BY avg\_home\_performance ASC;

--Away İken Ortalama Performansı En Düşük 10 Takım

SELECT TOP 10

t.full\_name AS team\_name,

AVG(p.away\_performance) AS avg\_away\_performance

FROM Performance p

JOIN Team t ON t.id = p.away\_team\_id

GROUP BY t.full\_name

ORDER BY avg\_away\_performance ASC;