FOOTBALL LEAGUE MANAGEMENT

2023

Group 9

- Lê Đại Lâm
- Nguyễn Minh Khôi
- Hoàng Khải Mạnh

INTRODUCTION

The purpose of this project is to provide a friendly environment to:

- **Maintain and update the details of players, Team, League and every record needed.**
- **Check input data**

PROBLEM

- **Football tournaments are often large-scale.**
- **Managing manually face many difficulties and costs:**
  - Difficulty in storing and retrieving
  - Easy to make mistakes
  - Difficulty in analyzing data

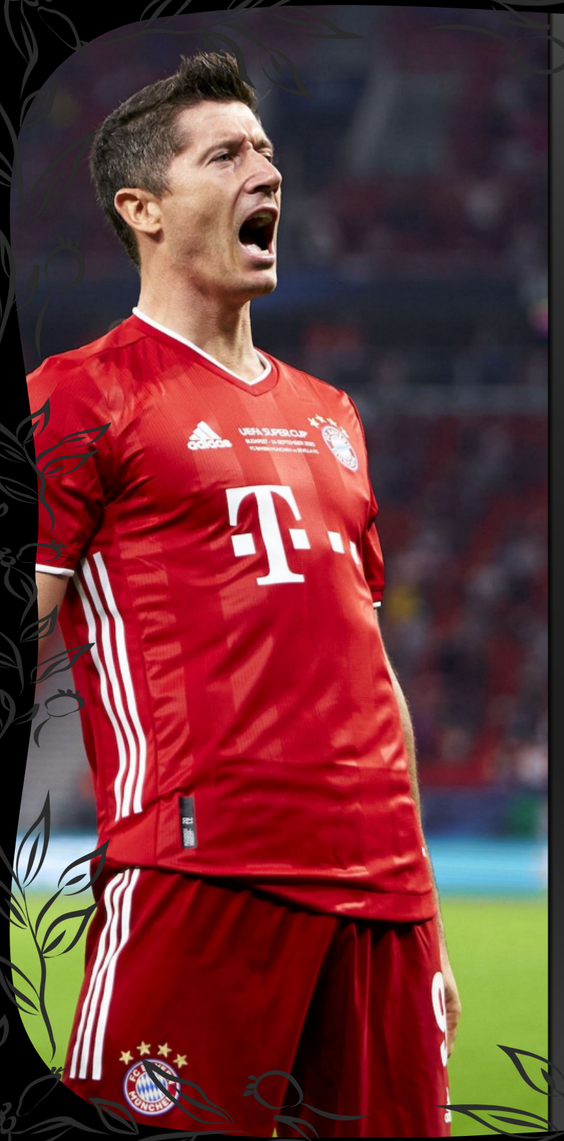
REQUIREMENT

- Manage team information
- Manage player information
- Manage match information
- Manage coach information
- Manage referee information
- Manage ticket information
- Generate reports and analyze data
- Manage Injury information

BACKGOUND

- **Fuelled by the frustration of missed matches, we've built software that delivers football fans instant game summaries, player stats, and league standings, even when they're on the go.**

- **Our software brings the beautiful game to you. Get instant game recaps, player stats, and league tables, all in one convenient platform.**



Non-Pen

Bundesliga,

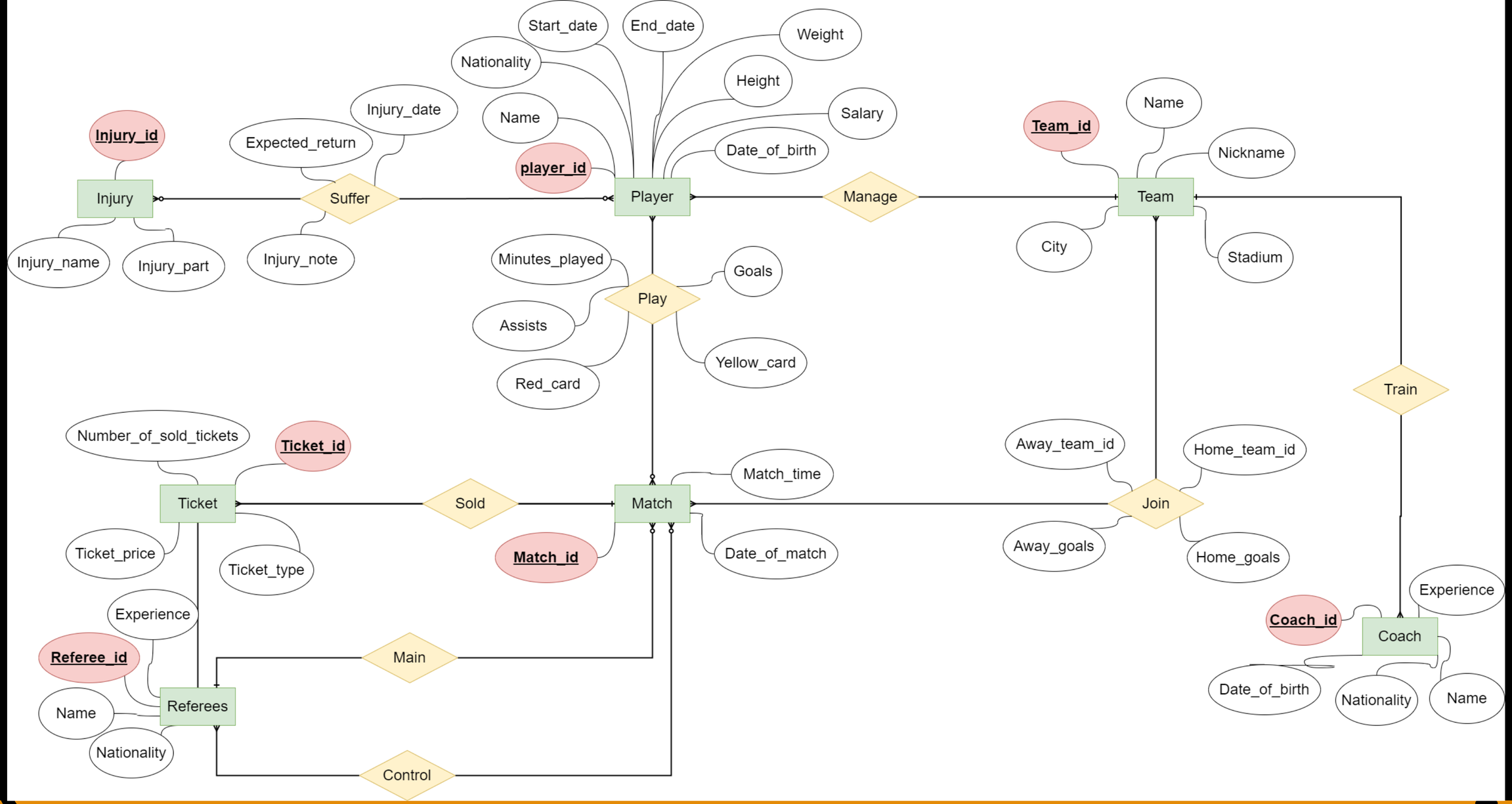
1	Robert Lev	Bayern Munich
2	Wout Weg	Wolfsburg
3	André Silv	Eintracht Frankfur
4	Erling Haa	Borussia Dortmun
5	Timo Wern	RB Leipzig (since

METHOD

Data modeling: This method uses data models to describe the structure and data of a database.

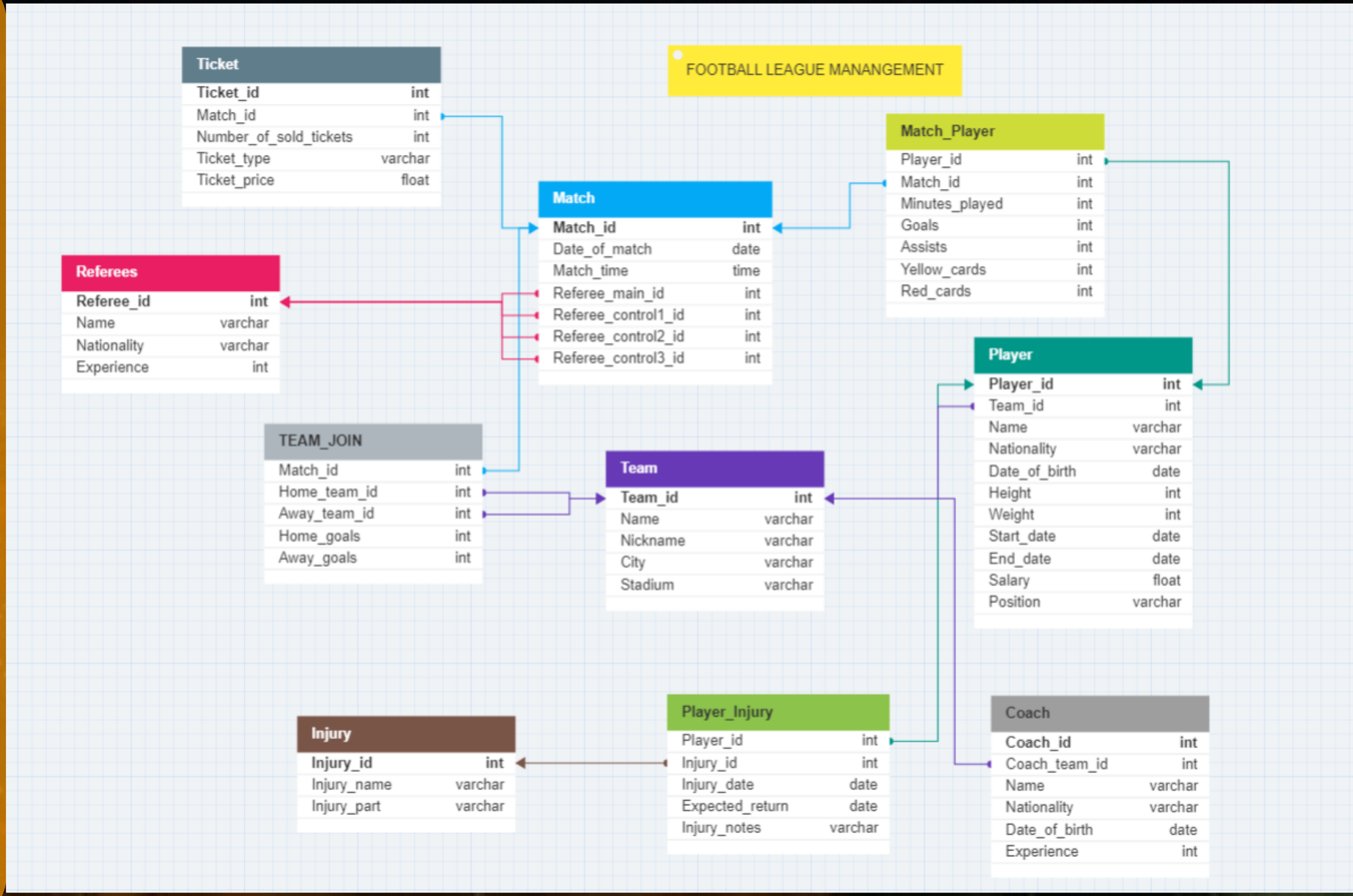
- **Entity Relationship Diagram (ERD):** A conceptual data model used to describe entities and the relationships between entities.

- **Relational schemal:** A logical data model used to describe tables and the relationships between tables.



ENTITY RELATIONSHIP DIAGRAM (ERD)

Relational schema



INDEX

- **Match:**
  - Date_of_match: B-tree index. Useful for date-based queries, such as finding matches within a specific date range.
  - Referee_main: B-tree index. Enhance query performance related to referee information.
- **Player:**
  - Nationality, Date_of_birth: Composite B-tree index. If there are frequent queries based on nationality or date of birth.
  - Player_id, Match_id: Composite B-tree index. Useful for querying player information within each match.
- **Team_join:**
  - Home_team_id, Away_team_id: Composite B-tree index. Supporting queries for match results of specific teams.
- **Ticket:**
  - Match_id: B-tree index. Supporting ticket queries by match.

Reasons for Choosing B-tree Index:

- B-tree index is preferred due to its ability to support range-based queries, sorting, and efficient searching.
- B-tree indexes perform well with both queries on a specific value and range-based queries.
- Most cases in your database require the flexibility and performance that a B-tree index can provide, especially in queries involving date ranges, foreign keys, and data sorting.

INDEX

TRIGGER

- Ensure one referee can control only one match per round
- No match has been duplicate
- Check players are eligible or not to play the match (have red card in previous match or have injury before the match and not return jet
- Check players are play the match in two teams joined or not

FUNCTION

- **For players:**
  - Insert into match_player, player_injury with optional amount of data
  - Top scorers
  - Query to find out how many assists and goals a player has in a given match
  - Query how many U23 players currently play in a given nation
- **For teams:**
  - Insert into team_join with optional amount of data
  - League Table sorted follow: Points, DIFF, Goal_for
- **For matches:**
  - Insert into match with optional amount of data
  - The match has the highest revenue
  - The most lopsided match (number of goals diff is highest)
  - Search matches between 2 given days
  - Find the results of matches between 2 given teams
  - Query which match has the highest revenue
- **For referees:**
  - Search referee who has captured the most matches

AFTER CLASS - LESSON

- Teamwork Lesson
- Database problem analysis steps
- Database design experience

TEAMWORK LESSON

- Listen and respect each other's opinions
- Define the team's common goals
- Assign clear responsibilities
- Interact regularly
- Resolve conflicts effectively

DATABASE PROBLEM ANALYSIS STEPS

Gather requirements: This can be done through interviews, surveys, existing documentation, or other methods.

Analyze requirements: Identify the data to be stored, the necessary system functions, and the relationships between data.

Design a data model: The choice of the appropriate data model depends on the system requirements.

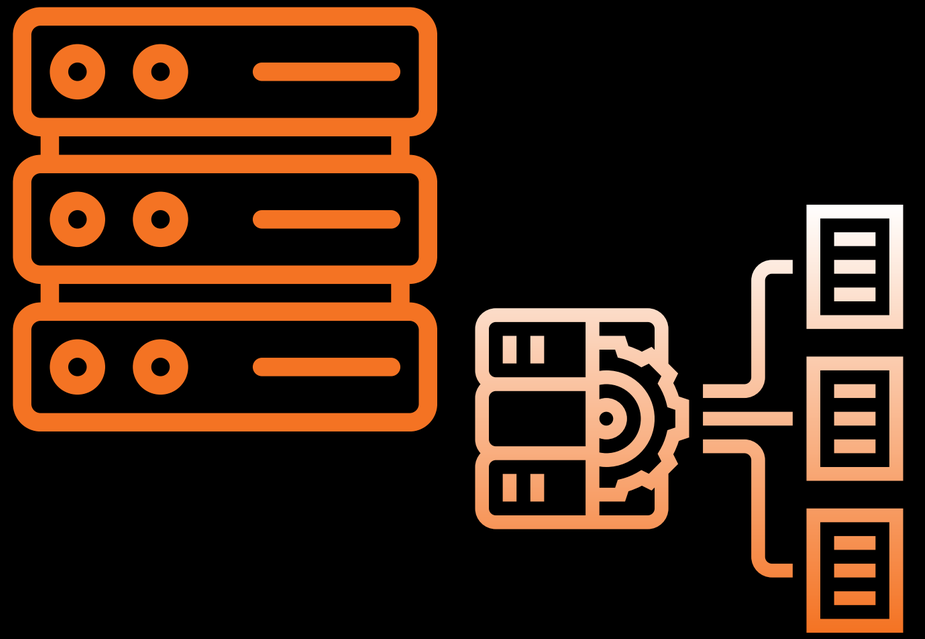
Design the database: Identifying the tables, columns, primary keys, foreign keys, and other constraints.

Test and evaluate: Ensure that the database meets the requirements of the system.

DATABASE DESIGN EXPERIENCE

- Understand the system requirements.
- Use data models.
- Adhere to database design principles.

- Choose a design method.
- Testing and evaluation.
- Use supportive tools.



THANK YOU!