Basics of database systems

NHL Database – Database design

Toni Koskinen

Eero Ryhänen

Lappeenranta-Lahti University of Technology LUT
Software Engineering

Basics of database systems
Spring 2023

TABLE OF CONTENTS

1 DEFINITION

NHL Database

In project ‘NHL Database’ a database is created to store information and advanced statistics about players and teams in NHL (National Hockey League). The database has data from seasons 2018-2022. The stats contain basic statistics, like goals and assists which are familiar to casual hockey fans, but also more advanced stats like corsi- and Fenwick-percentage and expected goals which may not be so familiar to everyone. These however are a powerful tool for a team’s head coaches and general staff. They can be used to evaluate teams and players strengths and weaknesses more accurately. For coaches they can be used to change training focus to areas which need it most, and for general manager and other staff, they can be used to find improvements to their team and find undervalued players from trade market. The value of data has in recent years been noticed by organisations, and nowadays each team in the league has an analyst team.

Other group which uses the data is professional and hobbyist betters. They use the data to make statistical models to predict the outcome of a game, and by that information place bets on the game to achieve profits. This would not be possible to do without data analysis, because the betting sites take their cut on the profits, and therefore by random betting one always achieves returns under 100%.

The following database queries have been implemented: (1) List all of the skaters in the league based on their points that season. The user can choose from which season they want the data from. (2) List of all the goalies in the league based on their save percentage that season. The user can choose from which season they want the data from (3) List all of the teams in the league based on their expected goals percentage that season. This data shows how dominant a team has been. The user can choose from which season they want the data from (4) List of all players in a team. The user can choose from which year and from which year they want the roster from. (5) Information about players stats and the team they have played from each season between 2018-2022. Also shows the total score from that timespan. (6) All information from one player. Their number, player id, birthdate, nationality, handedness and number. (7) An example that shows how the many-to-many example has been implemented.

2 MODELING

2.1 Concept model

In Figure 1 is the ER model of the designed database. There are six entities in the model and six relationships. There are three N:M relationships: in between Player and Season, in between Team and Season and finally in between Player and Team. The primary keys in entities Player and Team are ID. These are unique for every player and team in the database, and they allow the user to distinguish two players with same names from each other. A player can be considered as a goalie or a skater, therefore a player can have either Goaliestats or Skaterstats. Teams can have their own team level stats, Teamstats, too. The stats can be from different situations, for example stats of 5 on 4 (powerplay) situations, and from different seasons. Each season has unique ID, which identifies them from each other. The N:M relationship in between Player and Season is there because each Season has multiple Players, and each Player has played during multiple Seasons. Same thing occurs in between Team and Season; Each Season has multiple Teams and each Team has played in different Seasons, therefore there is N:M relationship between them. Finally, there is N:M relationship in between Team and Player: each Team has had multiple Players and there are Players that have played for multiple Teams (in different seasons). The relationship in between Seasons is also linked to Goalie- and Skaterstats because a Player has different stats each season.

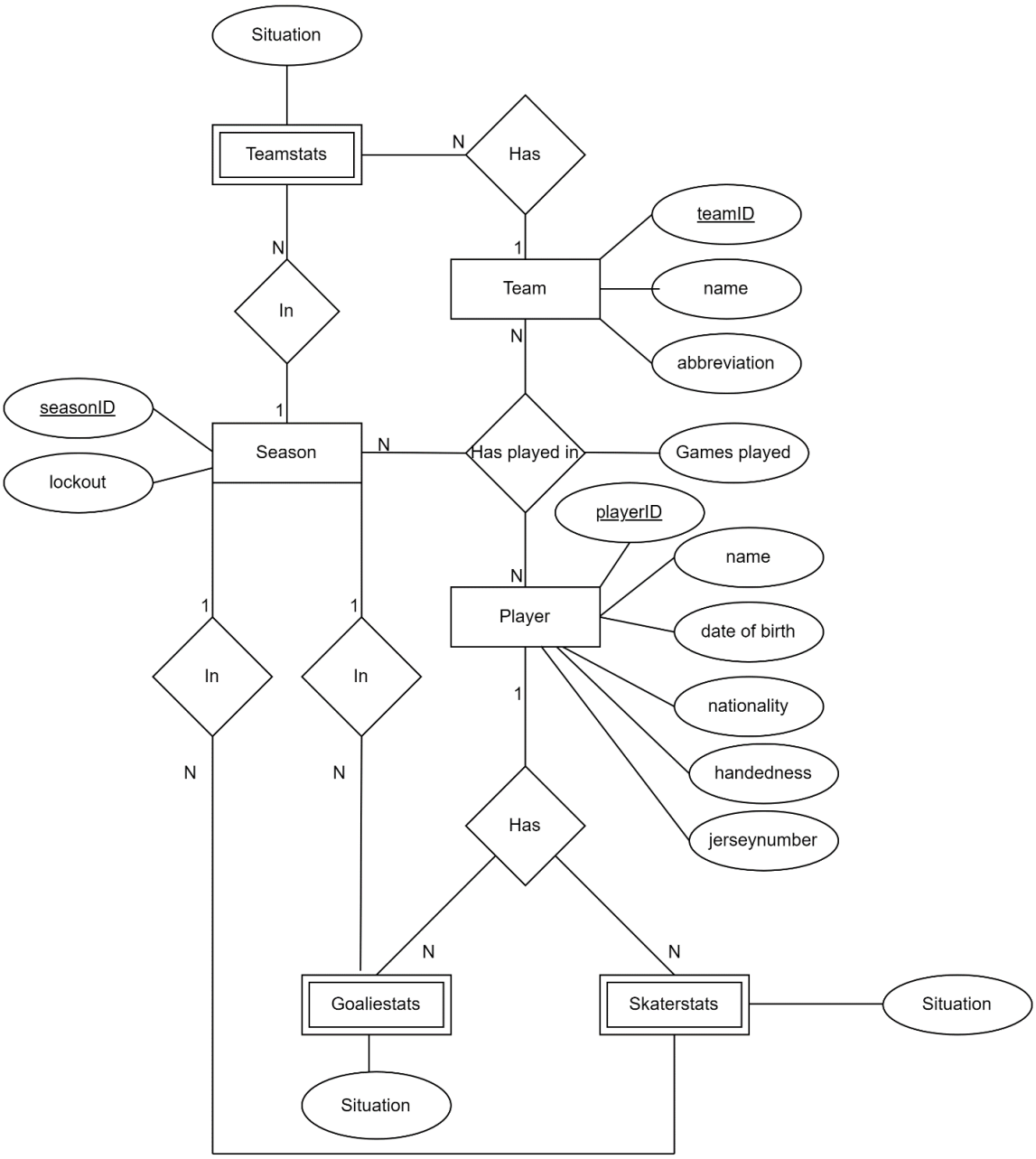

Figure 1: ER model

2.2 Relational model

Figure 2 shows the relational model that has been created based on the ER model. Due to the N:M relationship, an interim relation was created between Player and Season entities. PlayersInSeason-entity describes which team, which position and how many games the player has played during the season. This is implemented like this, because some players

change their teams and positions between the seasons. The position-variable then determines whether the player has Skaterstats or Goaliestats. There is also an interim relation between Team and Season called TeamsInSeason which describes how many games a team has played in a given season. This was implemented due to N:M relationship and because the teams have started their “journey” in the NHL in different seasons: for example, Seattle Kraken, the newest team in the league, has been there only since 2021. Stats-entities have been written with “Lots of stats” because Goaliestats has more than 30 columns and both Skaterstats and Teamstats have more than 100 different columns for advanced stats in them.

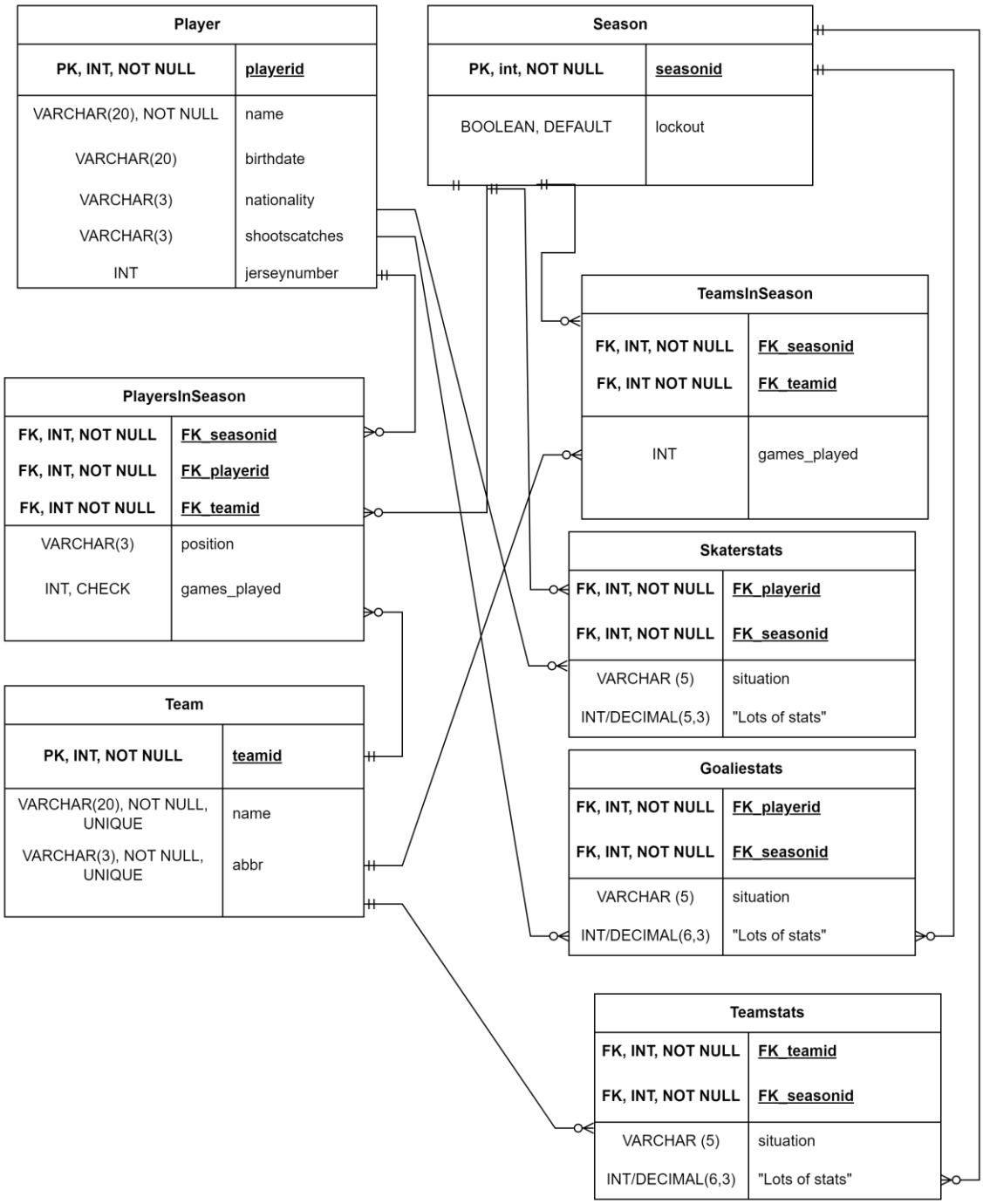

Figure 2: Relational model from the ER model

3 DATABASE IMPLEMENTATION AND PYTHON INTERFACE

During the implementation, the following constraints are created for relations:

- **Player**
  - ID of the player and name cannot be null. (NOT NULL)
- **Team**

    - ID of the team, name of the team and abbreviation of the name cannot be null. (NOT NULL)
    - Team name and its abbreviation are unique, so there are no other teams with the same name or abbreviation. (UNIQUE)
- **Season**
    - ID of the season cannot be null (NOT NULL).
    - Lockout tells if a season has been affected by a lockout and defaults to false. (DEFAULT)
- **PlayersInSeason**
    - Foreign key reference to player, (represented) team and season.
    - ON DELETE CASCADE
    - Player must have at least one (1) game played to be part of a season. (CHECK)
- **TeamsInSeason**
    - Foreign key reference to team and season.
    - ON DELETE CASCADE
- **Skaterstats**
    - Foreign key reference to player and season.
    - ON DELETE CASCADE
- **Goaliestats**
    - Foreign key reference to player and season.
    - ON DELETE CASCADE
- **Teamstats**
    - Foreign key reference to team and season.
    - ON DELETE CASCADE

In addition to the integrity constraints, the database will also implement five indices which are based on the foreign keys that are in PlayersInSeason, TeamsInSeason, Skaterstats, Goaliestats and TeamStats. These indices enable to perform faster search for specific player and team statistics.

Python Interface

The Python interface contains the following functions:

- **main**
    - Runs the menu structure of the program and calls other functions based on user's selections.
- **initializeDB**
    - Initializes the database (if needed) in the beginning of the program by executing the commands from the file: sqlcommands.sql.
- **showSkaters**
    - Prints the name, team (abbreviation), goals, assists and points of each skater in the user-selected season. The results are ordered from the highest number of points to the lowest.

- **showGoalies**
  - Prints the name, team (abbreviation), save percentage, and goals against average of each goalie in the season selected by the user. The results are ordered from the highest save percentage to the lowest.
- **showTeams**
  - Prints the name, name's abbreviation, goals, goals against and expected goals percentage of each team in a user-selected season. The results are ordered from highest expected goals percentage to the lowest.
- **showPlayersInTeam**
  - Prints the name, team (abbreviation), nationality, handedness, position and jersey number of each player of selected team in the selected season. The results are ordered from the lowest number to the highest.
- **showCareers**
  - Prints the number of games played, goals, assists, points, plus-minus statistics and penalty minutes of the selected player from each season and these statistics combined for the entire career.
- **searchOnePlayer**
  - Prints a row from Player table based on user's input: name.
- **manyToManyExample**
  - Visualizes many-to-many relation between Player and Season.
- **insertPlayer**
  - Allows user to insert data to Player table. The user is allowed to insert a name, birthdate, nationality, handedness and jersey number.
- **updatePlayer**
  - Allows user to update data in Player table by changing jersey number of a selected player.
- **deletePlayer**
  - Allows user to delete data from Player table. The removal is based on the name of the player user enters. The referenced rows, for example the statistics, from the other tables will be removed too.

4 DISCUSSION

The data used in the project was from MoneyPuck.com.