Basics of database systems

**Project – Database design**

Lappeenranta-Lahti University of Technology LUT

Software Engineering

Basics of database systems

Spring 20XX

Table of contents

[Table of contents 1](#_Toc92697309)

[1 Definition 2](#_Toc92697310)

[2 modeling 4](#_Toc92697311)

[2.1 Concept model 4](#_Toc92697312)

[2.2 Relational model 5](#_Toc92697313)

[3 database implementation 6](#_Toc92697314)

[4 discussion 7](#_Toc92697315)

# Definition

**Example text: Staff 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.

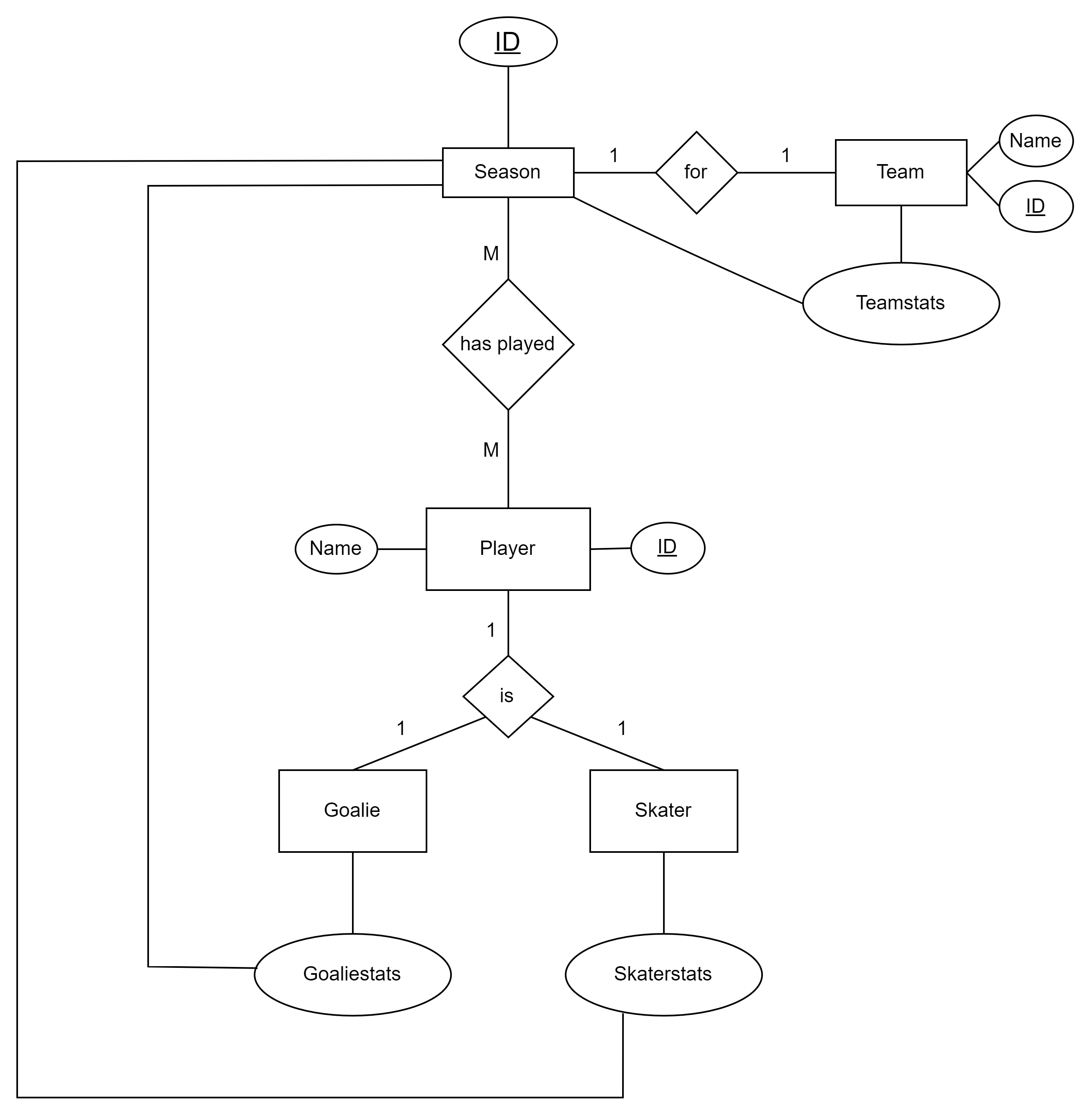
Other group which uses the data is professional and hobbyist betters. They use the data to make statistical models to predict the outcome of a game, and by that information place bets on the game to achieve profits. This would not be possible to do without data analysis, because the betting sites take 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 all of 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.

# modeling

## Concept model

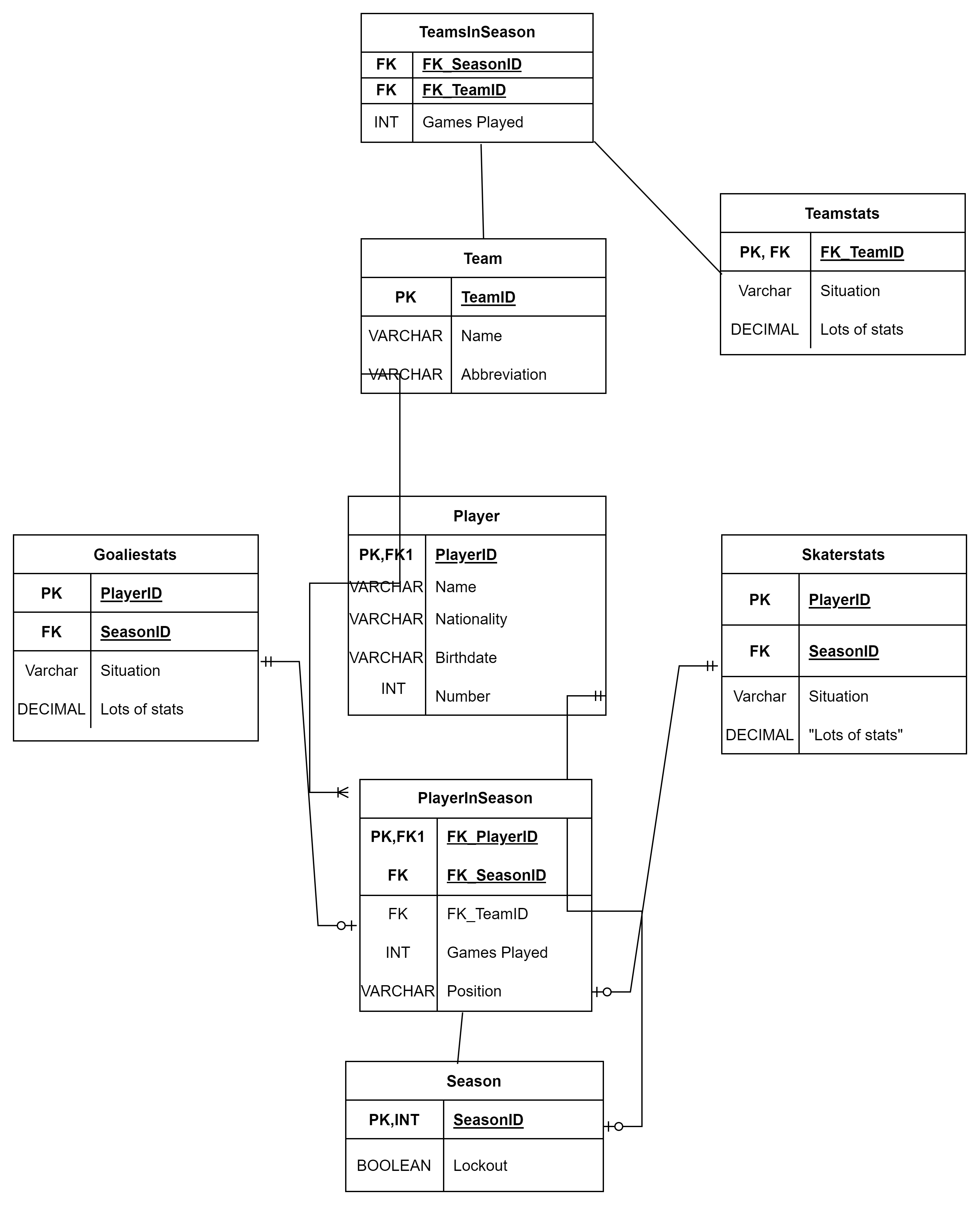
In Figure 1 is the ER model of the designed database. There are five entities in the model and three relationships. There is one N:M relationship in between Player and Season. The primary keys in entities Player and Team are ID. These are unique for every player and team in the database, and they allow the user to distinguish two players with same names from each other. Players are separated into two different entities. A player is either a Goalie, or a Skater. Each Skater and Goalie object is linked to the Player with a foreign key ID. Players “position”-attribute determines if a Player is Skater or Goalie. Each season has unique ID, which identifies them form each other. The N:M relationship between Player and Season is there, because each Season has multiple Players, and each Player has played during multiple Seasons. Seasons are also linked to Goalie- and Skaterstats, because a Player has different stats each season.



**Figure 1:** ER model

## 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 seasons. The position-variable then determines whether the player has Skaterstats or Goaliestats. There is also an interim relation between Team and Teamstats called TeamsInSeason. This is done for the same reasons, teams have different stats each season. Stats-entities have been written with “Lots of stats” because each has >100 different columns for advanced stats in them.

** Figure 2:** Relational modelfrom the ER model

# Database implementation

During the implementation, you have to develop the different integrity constraints as well as indices. Describe the constraints and indices you have created for your database. You can decide what kind of format you use for describing them. The example shows each relation in a list and the constraints in them, you can use the same format or use tables or whatever seems best for you.

If you have created a Python interface, describe that here as well.

**Example text**

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

* **Staff**:
  + Name, date of birth and address cannot be null (NOT NULL)
  + Date of birth has to be at least 18 years ago (CHECK)
* **Wife**:
  + Foreign key reference to staff.
  + Name cannot be null (NOT NULL)
  + ON DELETE CASCADE
* **Child**:
  + Foreign key reference to staff
  + unique composite key of staffID and child name so that the same staff member doesn’t have children with the same name.
* **Perform**:
  + Foreign key reference to staff and task
  + ON UPDATE CASCADE
* **Task**:
  + Name cannot be null (NOT NULL)
* **Work**:
  + Foreign key reference to staff and company.
  + Date cannot be null and defaults to current date (NOT NULL, DEFAULT)
  + ON UPDATE CASCADE
* **Company**:
  + Name cannot be null (NOT NULL)
* **PhoneNumber**:
  + Foreign key reference to staff
  + unique phone number so that there aren’t any others with the same phone number. (UNIQUE)
  + ON UPDATE CASCADE
  + ON DELETE RESTRICT

In addition to the integrity constraints listed above, the database will also implement two indices; One based on the Task name, another based on the Work since. These indices are to allow quickly search who perform the same tasks as well as to search for employees that have started working during a specific time period.

# discussion

If you want to mention something that has not been discussed in the previous chapters, you can discuss them here.

**Example**

Nothing to discuss.