Databases HW2

Taya Harshuk 207944257

Jonathan Josef 203304480

Database Design:

Our DB contains 6 Tables and 9 views. A graph of table's dependencies attached below.

The tables are:

Teams: Contain team IDs

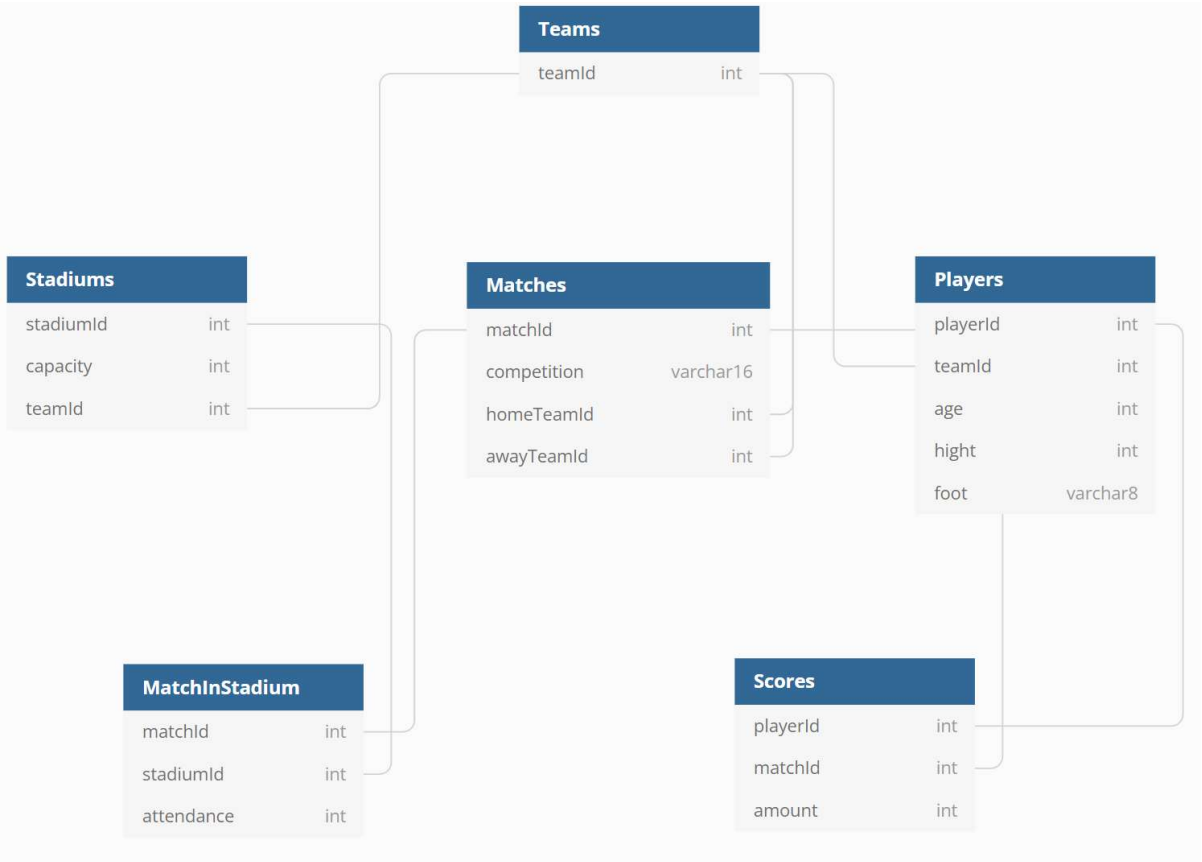
Players: Contain Players (player Id, team id [reference of team Id in Teams], age, height, age, foot)

Matches: Contain Matches (match Id, home & away team ids [references of team Id in Teams], competition)

Stadiums: Contain Stadiums (Stadium Id, team Id [reference of team Id in Teams], capacity)

MatchInStadium: Contain records of matches taking places in stadiums (match Id, stadium Id, attendance)

Scores: Contain records of players score in a match (player Id, match Id, amount)



The views are:

personalStats:

Schema:

PlayerId (for all players)	matchId (a record for every match played by PlayerId), None if didn't play any match	amount (the amount of goals scored by PlayerId in MatchId). 0 if didn't score.

Used for: playerIsWinner

goalsPerMatch:

Schema:

matchId (only for matches that had goals)	Goals scored in that match

Used for: playerIsWinner, goalsPerStadium (another view)

goalsPerPlayer:

Schema:

playerId (for all players)	Goals scored by that player (including 0)

Used for: mostGoalsForTeam

activeTeams: A list of active team IDs. Used for ActiveTallTeam

tallTeams: A list of tall team IDs. Used for ActiveTallTeam

activeTallTeams: A list of active tall team IDs. Intersect between the last two views.

Used for: getActiveTallTeams, getActiveTallRichTeams.

minAttendancePerTeam:

Schema:

teamId (only teams that played at least one home match)	minAttendanceInHomeMatch (self-explanatory). 0 If # attendance doesn't exist for at least 1 home match.

Used for: popularTeams

goalsPerStadium:

Schema:

StadiumId (only stadiums that hosted at least 1 match)	goals Number of goals scored in the stadium. Including 0.

Used for: getMostAttractiveStadiums

friends:

Schema:

Pid1	Pid2

Description:

A record for each pid1 scored in the same match pid2.

Contain duplicates (both for multiple matches, and ‘reversed’ duplicates for the same match. i.e., the relation is symmetric)

Used for: getClosePlayers

Code Design:

We created a few utils function and two global lists – Tables & Views.

It's important to understand that those lists aren't important for the code logic, only for good coding practices (for example, avoid using explicit table names on create/clean/drop).

Utils function are:

_errorHandling(e) -> ReturnValue:
Catch an exception and convert it to.
Another error has been added: Error # 22001.
This extra error is a special treatment to a case where a string (foot/competition) is not in the defined domain ({'Left', 'Right'} / {'Domestic', 'International'}) but is also out of the defined length (8/16).
With this special treatment we could limit the strings without fearing of DB error instead of BAD_PARAMS.

sendQuery(query) -> collections.namedtuple("QueryResult", ["Status", "RowsAffected","Set"])
Send a query, convert exceptions if gets any, and returns a named tuple with the values of Status (the ReturnValue), RowsAffected, and Set (which is the ResultSet)

_createTable(name, colNames, colTypes, extraProperties, foreignKey=None, checks=None, extraStatements=None)
Returns a table definition for the table generator

_createView(name, query, toMaterialize)
Returns a view definition for the view generator

defineTables()
Define all the tables, and append them to Tables in the right order

defineView()
Define all the views, and append them to Views in the right order