CMPT 354 – Mini Project 2

Presented by:

Yawen Hou 301354054

Zihan Wang 301329429

Simon Fraser University

Aug 1, 2019

**Brief Overview of Our Mini Project 1:**

For the mini project 1, we designed a database called “FootballHome” for football fans to help them look up relative football information, including matches, football teams, team players, awards, etc. Basically, we designed seven main entities sets and four relationships in tables:

Entity sets:

* FootballTeam (TeamName, Country, City, Founded)
* HeadCoach (Name, DateOfBirth, CoachingYears, Gender)
* Award (TeamName, Competitions, Titles)
* Matches (GameDate, HomeTeam, AwayTeam, Location)
* Referees (Name, DateofBirth, Gender, Country)
* TeamPlayers (Name, DateOfBirth, Gender, Height, Position, Nationality)
* RetiredPlayers (Name, RetiredDate)

Relationships:

* BelongTo (PlayerName, TeamName) - connects TeamPlayers and FootballTeam
* Plays (TeamName, GameDate) – connects FootballTeam and Matches
* Manage (headCoachName, TeamName) – connects FootballTeam and HeadCoach
* Preside (GameDate, RefereeName) – connects Matches and Referees

Please note that we do not set the relation table for the relationship of weak entity sets and subclasses, just to avoid redundancy. The detailed design (graph) is attached in Appendix.

**Q1: Does your design allow anomalies?**

We did a comprehensive review of our design and supposed that there did not occur many serious anomalies. As Dr. Karimi discussed, there are three main anomalies that may happen:

* Update Anomaly: one or more instances of duplicated data is updated, but not all.
* Insertion Anomaly: certain attributes cannot be inserted into the database without the presence of other attributes.
* Deletion Anomaly: certain attributes are lost because of the deletion of other attributes.

Here is a detailed analysis of each table:

* FootballTeam (TeamName, Country, City, Founded)

A sample tuple of FootballTeam is like: ('Inter Milan', 'Italy', 'Milan', '1908-03-09'). The primary key of the table FootballTeam is TeamName. To simplify we will use “T, Co, Ci, F” to represent these four attributes.

If we know the value of TeamName, we can obtain its origin Country, City and when Founded. By this, we can say that the Country, City and Founded are functionally depended (FD) on TeamName. Using the splitting rule, it could be shown like: “T -> Co, T -> Ci and T -> F”. Notice that this is also the minimal basis for FootballTeam table.

If we are concerned on the attributes closure, some combinations of Co, Ci and F with T

also make sense as FD. For example, “T -> Co, Ci”, “T -> Ci, T” or “T, Co -> Co, Ci, F” are acceptable FDs as well. As long as we know the value of a TeamName (it is unique: no 2 footabllTeams can have the same name and do match), we could know its origin country, city and founded date. When we update a football team, that team will be updated and will not influence the others. When we insert a new team, it does not make much sense to just input Country, City and Founded Date if we do not provide the TeamName. When doing deletion, some information may get lost with the deletion of TeamName. But they are not that important as we are concerned the TeamName best. Hence, there is no anomalies in FootballTeam table.

* HeadCoach (Name, DateOfBirth, CoachingYears, Gender)

Similay with the analysis before, for HeadCoach table, we use “N, D, C, G” to represent these four attributes.

FDs: as the primary key is N and D, the direct FDs are: “N,D -> C, G”, which means CoachingYears and Gender are functionally depended on the Name and DateOfbirth. Some other acceptable FDs are like “N, D -> C”, “N, D -> G” and “N, D, C -> G”.

There are no FD Anomalies because: for update anomaly, it is almost impossible to have two coaches with the same name and date of birth, which means that every coach is independent and will not influence the others. The same works for deletion and insertion. We do not care about the information of CoachingYears and Gender as far as we know which coach it is. Hence, there is no serious anomalies in HeadCoach.

* Award (TeamName, Competitions, Titles)

The award table is a subclass of FootballTeam, which shows how many awards that the football team has. We will use “Te, C Ti” to represent these three attributes.

FDs: As the primary key is Te, the direct FDs are: “Te -> C, Ti”, which means that competitions and titles are functionally depended by the teamname. Other optional Fds are like: “Te ->C” and “Te -> Ti”.

This is a simple table and there are no obvious anomalies. For update action, basically we just update the titles of that competitions if a team wins an award again. If it is a new award, we just create its competition title and set the title number to be 1. There does not have two same competitions occur. For deletion / insertion, it may cause some anomaly if we are really concerned about one competition, let’s say World Cup. However, the awards are belong to the team and together with which team wins it, the information could make sense. Hence there are no anomalies.

* Matches (GameDate, HomeTeam, AwayTeam, Location)

We use “G, H, A, L” to represent the table. The primary key is “G, H, A” together.

FDs: The most important functional dependency is “G, H, A -> L”, which means that the location of a match is functionally depended by the gamedate, hometeam and awayteam.

There are some kind of anomalies occurred in this table. For updating, everything works

well as the GameDate and two against teams are uniquely defined together. However,

for insertion and deletion, if we only care about the HomeTeam, AwayTeam and location, we cannot add a new match unless we have at least a clear GameDate for that. Also, there is some kind of redundancy because if two teams fight with each other, there will be two records stored in the database with the same GameDate and Location as well as two inverse HomeTeam and AwayTeam information. Actually, we just need one of them. No need of decomposition, but we need to simplify it a little bit. Hence, if the database sees two matches have the same GameDate and Location, just inverse HomeTeam and AwayTeam, it should just keep one and stop the insertion.

* Referees (Name, DateofBirth, Gender, Country)

The primay key is “Name and DateofBrith”. We use “N, D, G, C” to represent the attributes. Referees preside the specific match. Unlike the table “HeadCoach”, this is not one-to-one relationship with table Matches because for each match there could be many referees and for each referee, he/she could preside many matches. This is a many-to-many relationship.

The most obvious FD is “N, D -> G, C”. Some other combinations like “N, D -> G”, “N, D, G -> C” also work.

Like the table “HeadCoach”, there is no anomaly for this table. Because we only care about the information of Gender and Country as far as we know the Referees’ Name and DateOfBirth. Name and DateofBirth togerther as a key could almost identify a referee uniquely. It does not make much sense to leave the Gender and Country information

separately. Hence, the table satisfies the requirements of avoiding insertion, deletion and update anomalies.

* TeamPlayers (Name, DateOfBirth, Gender, Height, Position, Nationality)

The key is “Name and DateOfBirth”. We use “N, D, G, H, P, Nation” to reprensent the whole attribute. Note that this table has a many-to-one relationship with FootballTeam because for each team there are many teamplayers.

FDs: “N, D -> G, H, P, Nation”. Using the splitting rule, it could be splitted into 4 different functional dependencies. Combine any “G, H, P, Nation” to the left side, the FDs make sense as well.

The corresponding information “gender, height, position, nationality” all make sense with the existence of a TeamPlayer. Hence, it gets rid of insertion, deletion and update anomalies.

* RetiredPlayers (Name, RetiredDate)

The primary key is the Name of the player. Notice that RetiredPlayers is a subclass of TeamPlayers. They just have a special attribute “RetiredDate” to show that they are retired. We use “N, R” to represent the table.

The most obvious FD is “N -> R”, which means that the RetiredDate is functionally depended by the Name of players. As there are just two attributes, there are not many other important functional dependencies.

Similarly, as this is a simple relation schema, it could avoid the anomalies. However, we notice a design mistake in this table. The schema should also contain the DateOfBirth attribute from TeamPlayers table. Hence, the improved schema should be (Name, DateOfBrith, RetiredDate).

* For relationship tables:

As the relationship tables (BelongTo, Manage, Plays, Preside) just stores the keys of the tables that they connect with,

should we include all the keys in the relationship table? (I think so.)