ARISTOTLE UNIVERSITY OF THESSALONIKI

SURVIVOR

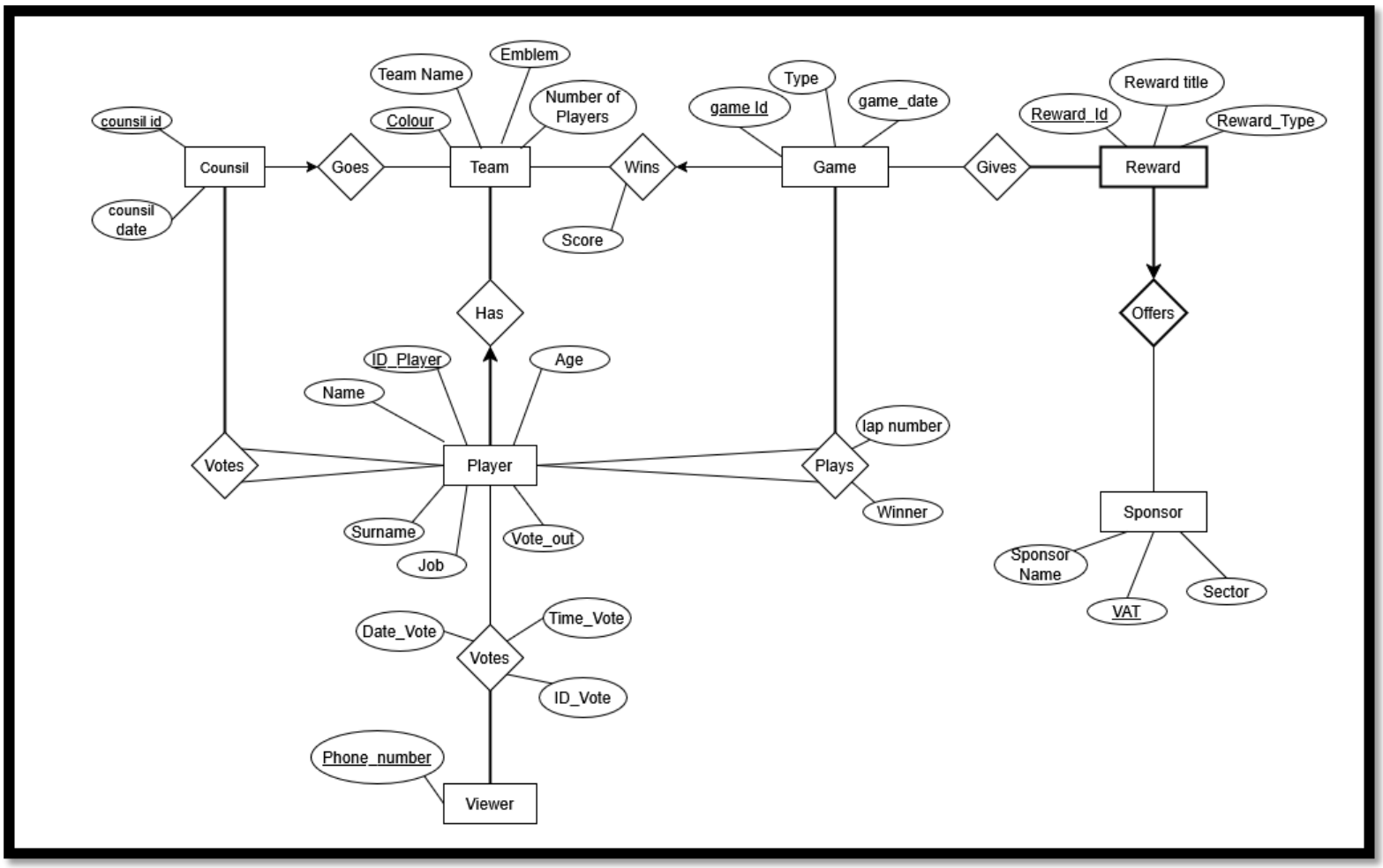
Databases Project

Anthopoulou Zoe Karakostas Konstantinos Petridou Georgia Tzitzis Anastasios

12/30/2017

Contents

At this point in the work, the updated Entity/Relationship diagram is presented, which has been edited with the aim of having a history of all activities, which offers great ease in statistical data. For this purpose, new entities and relationships were added.

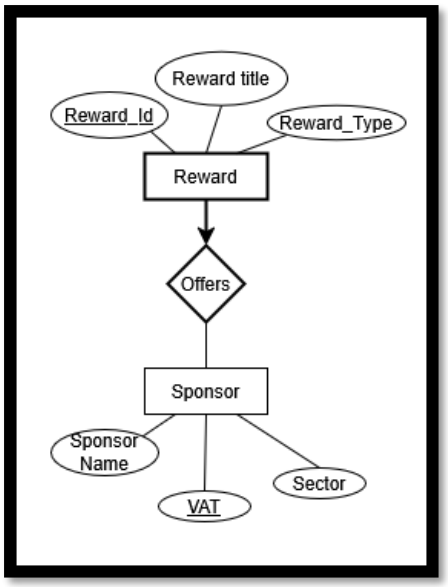


Below, the detailed presentation of the entities and relationships follows, along with the deconstruction of the code that implements the above database. The limitations of the variables used and their connection to the various entities through relationships are explained in detail.

Associations

"Offers" association

(Offers)



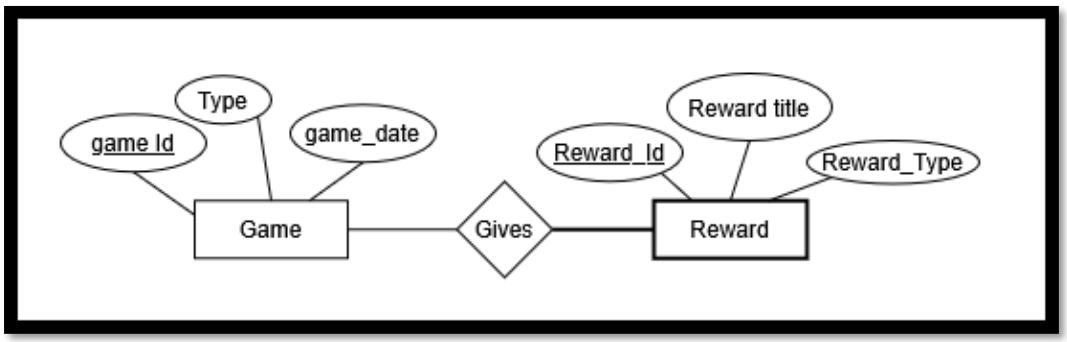
The prize is a weak entity. A sponsor may not offer a prize, but each prize must be offered by only one sponsor. According to the rules, only two tables will be created, and the Sponsor key will be entered into the prize table as a foreign key. The VAT number participates in the prize's primary key, so it cannot be NULL. Thus, each prize will have a sponsor, achieving the total participation of the prize in the association.

Reward{ ID_Reward, Name_Reward, Type_Reward, AFM }

Sponsors{ AFM, Brand, Field }

"Gives" association

(Gives)



We have a many-to-many relationship. A competition can have 0, 1, or more prizes, and a prize can be awarded to 1 or more competitions. The theory tells us that we need to make 3 tables.

We see the total participation of the prize, which however cannot be implemented (with what we know). Note that total participation means that each value of the prize ID must appear in at least one tuple of the GIVES table. One way to do this is according to the book page 75, to put the prize ID as a foreign key in REWARDS with a reference to Gives.

However, this is not consistent with the fact that the ID is not a candidate key in Gives. Finally, we leave it unimplementable.

Game{ ID_Game, Date_Game, Type_Game }

Reward{ ID_Reward, Name_Reward, Type_Reward, AFM }

Gives{ ID_Game, ID_Reward, AFM }

Variable	**Type**	**Domain**
ID_Reward	Int(10)	0 – 4294967295
ID_Game	smallint(5)	0 – 65535
AFM	Int(10)	0 – 4294967295

Obviously here the primary key is the combination of the candidate and foreign keys from the entities participating in the relationship.

A SOLUTION FOR TOTAL PARTICIPATION

Another idea to ensure full participation is the following:

Let two entities A and B, with a and b respectively as primary keys, be related by a many-to-many relationship R, and A has a full membership in R.

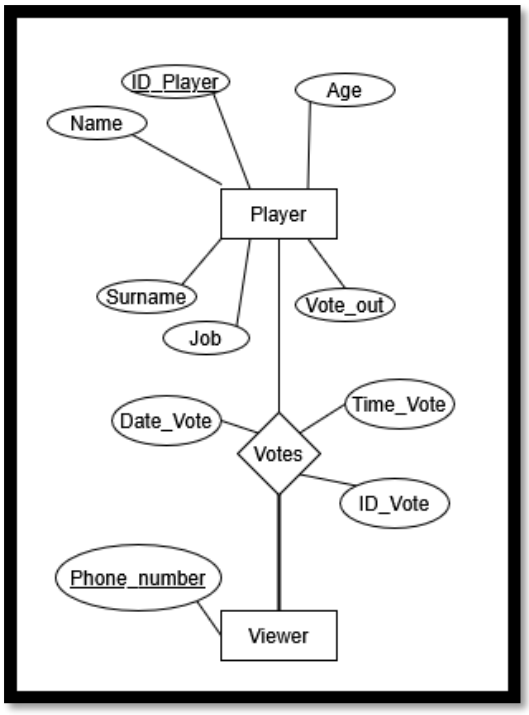
Then after the transaction is complete and all tuples are inserted into the table, we will check if each of the primary key values of A appears in at least one value of column a of table R. With SQL:

constraint total_part check (a in (select a from R));

set constraints *total_part* deferred;

"Viewer_votes" correlation

(Viewer votes)



We have a many-to-many relationship, just like before, so again 3 tables.

Viewer{Phone_Number}

Player{ ID_Player, Age, Name, Surname, Job, Vote_Out }

Viewer_votes{Phone_Number, ID_Player, Date_Vote, Time_Vote, ID_Vote}

We very reasonably claim that someone cannot vote multiple times at the same time, only once. That is why for each viewer, Date_Vote and Time_Vote are unique.

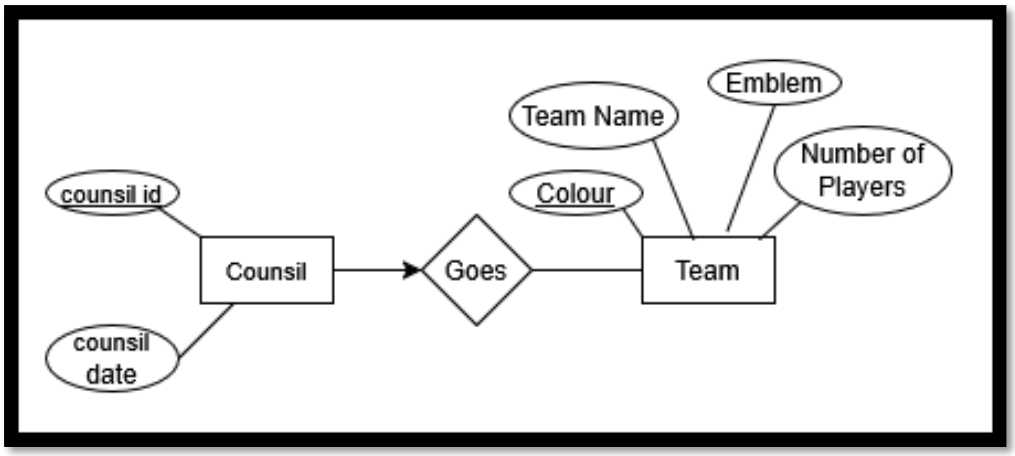
Variable	**Type**	**Domain**
Phone_Number	Int(10)	0 – 4294967295
Player_ID	Varchar(30)	30 characters maximum
Time_Vote	Time	
Date_Vote	Date	
ID_Vote	Int(8)	0 – 4294967295

Here the primary key consists of the combination of the keys Phone_Number (foreign key), Time_Vote and Date_Vote and ID_Player is defined as the candidate and foreign key.

It should be noted that the problem of total viewer participation in She Votes Again has not been solved.

"Goes" association

(He goes)



The association is 1-to-many with a key constraint for the council. This implementation is done with 2 tables (the GROUP key goes to the council), which certainly leads to the appearance of gaps in the council table (i.e. the councils after the union of the groups where there is no lost group to vote but everyone votes will have NULL in this attribute). However, we consider that there are not many and the peripheral memory they bind is acceptable. Otherwise, we would create 3 tables. We will apply the first method.

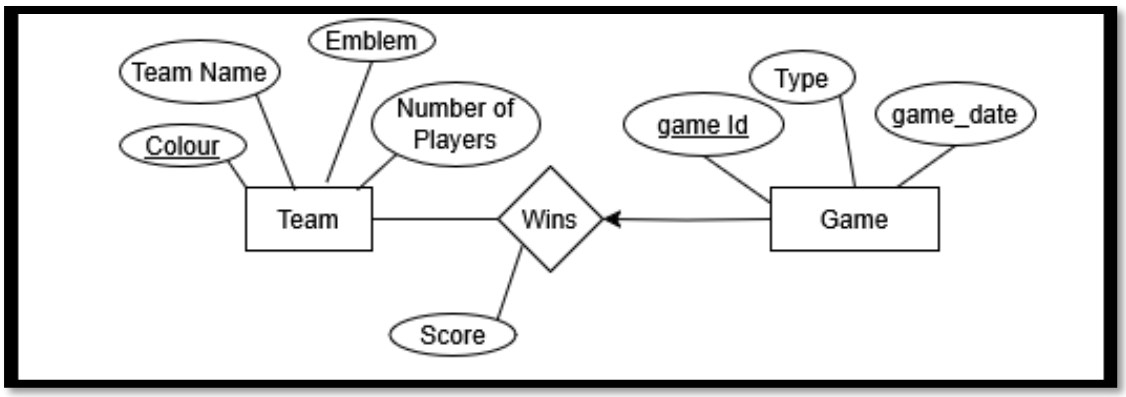
Team{ Color, Name, Badge, NumberofPlayers }

Council{ CouncilID, CouncilDate, Color }

Color can take a NULL value and essentially takes it after the groups are joined.

"Wins" correlation

(He wins)



A similar case to the previous one is the association “Wins” which is a one-to-many association with a key constraint on the one side. The constraint is that a competition does not have a winning team when its type is individual immunity. However, this does not create a gap problem for us since we consider that these competitions are few compared to the rest and we do not mind the NULL values that will appear. This association is one-to-many,

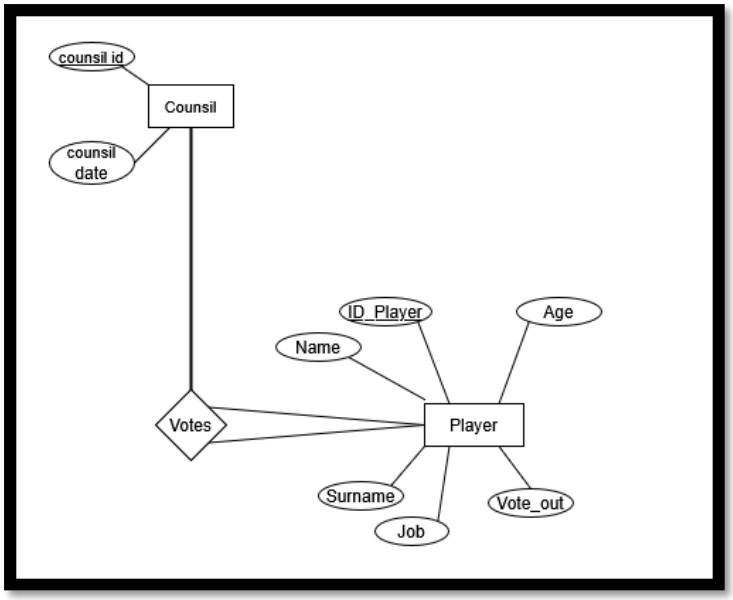
therefore only two tables are needed and thus the TEAM key and the attributes of the association (Score) are transferred to the COMPETITION table.

Team{ Color, Name, Badge, NumberofPlayers }

Game{ID_Game,Date_Game,Type_Game, Color, Score }

"Player_votes" association

(Player votes)



The association “Votes” is a ternary association in which the entities “Council” and “Player” participate. For this association we have that a player can vote for many players (but only one in each council), a player can vote in many councils, a player can be voted for by many players and in many councils and finally, in a council many players can vote and be voted for, but there must be at least one pair (voter, voted). This association is many-to-many, so we have three tables, “Player”, “Council” and “Votes”.

Player{ID_Player, Age, Name, Surname, Job, Vote_Out }

Council{ ID_Council, Date_Council, Color}

Player_Votes{ID_Council, ID_Voter, ID_Voted}

Variable	**Type**	**Domain**
ID_Council	smallint(6)	0 - 65535
Voter ID	smallint(5)	0 - 65535
ID_Voted	smallint(5)	0 - 65535

As we said in a council a player votes 1 time. So for each different ID_Council the ID_Voter will appear at most once (none after the player leaves and then). For this reason we take as primary key {ID_Council, ID_Voter}.

Regarding the constraints of the "Votes" table, we have that the ID_Council variable refers to the ID_Council variable of the "Council" table, in which it is a primary key, is a foreign key and is updated according to the "Council" table. The ID_Voter and ID_Voted variables refer to the ID_Player variable of the "Player" table, in which it is a primary key, are foreign keys and are updated according to the "Player" table.

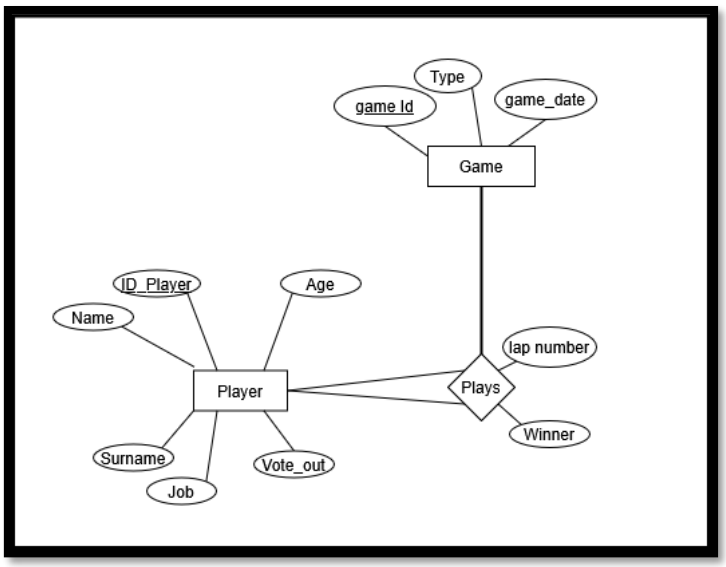
Below is an example of voting:

Council	**Voter**	**Voted**
1	Scree	Bo
1	Danos	Bo
1	Costas	Danos
1	Bo	Spaliara
1	Spaliara	Bo

Looking at the table, we see that by knowing the council and the person voting, we can find where their vote goes.

"Plays" association

(Plays)



And this association is a ternary relationship, here between the entities “Player” and “Match”. As can be seen from the figure, a player can play in a match against many players, a player can play in many matches, and in a match at least one pair of players {player1, player2} will play. In the association there is the attribute round number, and we consider

that in each round only one pair of players plays. Since here too the association is many-to-many, it will be reflected in three tables, “Player”, “Match” and “Plays”.

Player{ID_Player, Age, Name, Surname, Job, Vote_Out }

Game{ID_Game, Date_Game, Type_Game}

Plays{ID_Opponent1, ID_Opponent2, ID_Game, Round_ID, Winner}

Variable	Type	Domain
ID_Game	Smallint(5)	0 - 65535
Round_ID	Tinyint(3)	0 - 255
Winner	Smallint(5)	0 - 65535
ID_Opponent1	Smallint(5)	0 - 65535
ID_Opponent2	Smallint(5)	0 - 65535

The association "Plays" has as primary key the combination of Round and ID_Game so that a battle between player1 and player2 is uniquely defined. However, a player can play multiple times in a match with the same opponent, but in a different round.

Regarding the constraints, we have that the variable ID_Game refers to the variable ID_Game of the table "Game" in which it is the primary key, is a foreign key and is updated according to the table "Game". The variables ID_Opponent1 and ID_Opponent2 refer to the variable ID_Player of the table "Player" in which it is the primary key, are foreign keys and are updated according to the table "Player".

Below we have a table as an example:

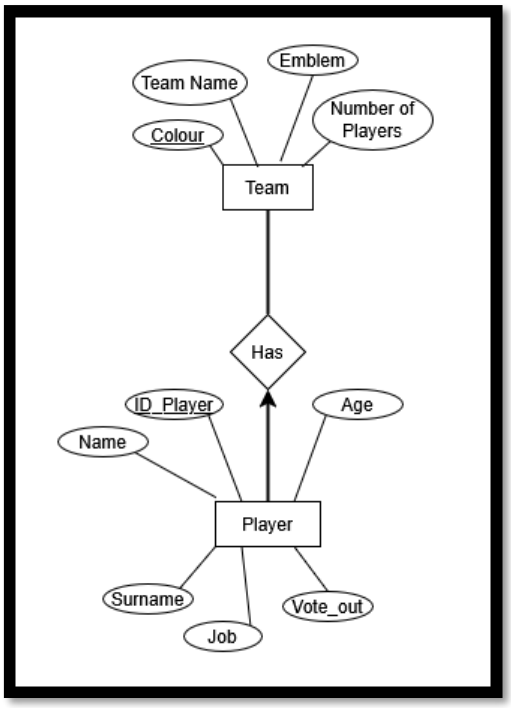
Game	Round	Player1	Player2	Winner
1	1	Danos	Spaliara	Spaliara
1	2	Laura	Scree	Scree
1	3	Danos	Bo	Danos
1	4	Laura	Scree	Laura
2	1	Danos	Spaliara	Spaliara

In the example table we see the pair Laura-Sara twice, for the same event but we see that they are in a different round. We also see the pair Danos-Spaliaras twice in the same round, but of course in a different event.

Once again we cannot guarantee that in the PLAYING table the ID of all events will appear at least once.

"Has" association

(He has)



In this case we have a 1-to-many relationship, in which both the many and the one have full participation. Specifically, a player belongs to exactly one team and a team has at least one player. According to our rules the player table will also take the team key as a foreign key as shown below:

Team{Color, Name, Badge, NumberofPlayers}

Player{ID_Player, Age, Name, Surname, Job, Vote_Out, Color}

By prohibiting the Color variable in the "Player" table from taking a NULL value, we achieve that each player has a team, that is, we achieve the total participation of the player. On the contrary, the total participation of the "Team" is not implemented.

Entities

"Sponsor" entity

(Sponsor)

The entity "Sponsor" is part of the association "offers". In the entity table, 3 variables are defined, none of which are inherited by another entity.

Variable	**Type**	**Domain**
AFM	Int(10)	0 – 4294967295

Brand	Varchar(30)	30 characters maximum
Field	Varchar(30)	30 characters maximum

The primary key is defined as the AFM, which will be clearly unique for each sponsor, and the Brand as the unique key.

"Reward" entity

(Prize)

The entity "Reward" is part of the association "offers". In the entity table, 4 variables are defined, of which 3 refer to the entity "Reward" (ID_Reward, Name_Reward and Type_Reward) and 1 to the entity "Sponsor" (AFM).

Variable	**Type**	**Domain**
ID_Reward	Int(10)	0 – 4294967295
Name_Reward	Varchar(30)	30 characters maximum
Type_Reward	Varchar(30)	30 characters maximum
AFM	Int(10)	0 – 4294967295

The primary key is defined as the combination of the candidate keys ID_Reward and AFM (which is inherited as a key from the Sponsor entity).

Regarding the entity constraints, the AFM variable is a foreign key and refers to the "Sponsor" table. Finally, the prize type must take one of the four values: 'food', 'Trip', 'Communication', 'Entertainment'.

"Player" entity

(Player)

The entity "Player" is part of the association "Has", "Plays", "Player Votes", "Viewer Votes". 7 variables are defined in the entity table, of which 6 refer to the entity "Player" and 1 to the entity "Team" (Color).

Variable	**Type**	**Domain**
Player_ID	Smallint(5)	0 – 4294967295
Age	Tinyint(2)	0 – 255
Name	Varchar(20)	20 characters maximum
Surname	Varchar(20)	20 characters maximum
Job	Varchar(30)	30 characters maximum
Vote_out	Date	'2017-02-13' - '2017-07-05'
Color	Varchar(15)	15 characters maximum

The primary key is the variable ID_Player, the unique key is the variable Name, and the candidate key is the variable Color, which is inherited from the team entity and takes a value depending on the team to which the player belongs.

The constraints of this entity concern the Color variable which is a foreign key and refers to the "Team" table. The constraints concern the restriction on the player's age which cannot be less than 18 or greater than 70. Also, the search date of the player's participation in the game must belong to the time frame in which the game was filmed.

Entity "Council"

(Council)

The entity “Council” is part of the “Goes” relationship in which we have a key constraint, with the key of the entity “Group” also going to “Council”. In the “Council” entity table, 3 variables are defined, two of which are related to the council (ID_Council, Date_Council) and one is related to the group (Color).

Variable	**Type**	**Domain**
ID_Council	Smallint(6)	0 - 65535
Date_Council	Date	'2017-02-13' - '2017-07-05'
Color	Varchar(15)	15 characters maximum

In the entity table "Council", the variable ID_Council is defined as the primary key, which is unique for each council. We have also defined the variable Date_Council as a key (candidate).

Regarding the constraints contained in the entity table "Council", we initially have that the Color variable is a foreign key and refers to the table "Group". Furthermore, the Date_Council variable can also only take values within the previously defined scope.

"Game" entity

(Race)

The entity “Game” is part of the relationship “Wins” and its table carries the attributes and constraints of this relationship. The entity table defines 5 variables, 3 of which refer to “Game” (ID_Game, Date_Game, Type_Game), 1 to “Wins” (Score) and 1 to “Team” (Color).

Variable	**Type**	**Domain**
ID_Game	Smallint(5)	0 -65535
Date_Game	Date	'2017-02-13' - '2017-07-05'
Type_Game	Varchar(20)	'Personal_Immunity', 'Team_Immunity', 'Communication', 'Reward'
Score	Varchar(10)	10 characters maximum
Color	Varchar(15)	15 characters maximum

In the table of the entity “Match”, the variable ID_Game is defined as the primary key, which is unique for each match. Also, the combination (`Type_Game`,`Date_Game`) is defined as the unique key since we cannot have two matches of the same type on one day,

for example two individual immunities on one day. Also, we have defined the variable Color as the key since it is the primary key for the entity “Team”.

Regarding the constraints contained in the “Match” entity table, we initially have that the Color variable is a foreign key and refers to the “Team” table. Another constraint is that the Type_Game variable can only take specific values, which are also its scope. Finally, the Date_Game variable can also only take values within the scope previously defined.

Both color and score take the value NULL when the GameType is "individual immunity".

"Team" entity

(Group)

The entity "Team" is part of three relationships ("Goes", "Wins" and "Has") from which, however, no attributes and constraints are transferred. 4 variables are defined in the entity table (Color, Name, Badge, NumberOfPlayers).

Variable	**Type**	**Domain**
Color	Varchar(15)	'Red' , 'Blue', 'White', 'Green', 'Purple', 'Black', 'Yellow', 'Orange', 'Pink', 'Brown'
Name	Varchar(20)	20 characters maximum
Badges	Varchar(20)	20 characters maximum
NumberOfPlayers	Tinyint(3)	0 – 255

The primary key is the Color variable and the unique keys are the Name and Badge variables. The only constraint in this entity concerns the Color variable which can only take values from the definition field given above.

"Viewer" entity

(Viewer)

The entity "TV Viewer" is part of the relationship "Votes" from which it does not inherit any attributes. It only has one variable, Phone_Number, which is obviously its primary key.

Variable	**Type**	**Domain**
Phone_Number	Int(10)	0 – 4294967295

Regarding constraints, a restriction is placed so that the Phone_Number variable can only take values that exclusively start with 69- and have a total of 10 digits (for mobiles) and that start with 2- and have a total of 10 digits (for landlines).