02170 Mandatory Group Project

Overwatch League

Sarah2
Adil Numan Çelik s196590
Xiaoran Ma s196275
Ömer Yılmaz s196596
Haoying Yu s191952

2020.4.3

Contents

1. Statement of Requirements

We are planning to develop a database for the Overwatch League – 2018

The goal is to arrange all the match data in good structure, including information about teams, players, matches and match maps.

A detailed description of our database is shown below:

The Game League is organized in named teams.

Each team has a team name and is located in a certain city and a certain country, it also records the ID of players and information about its sponsors.

Each team consists of named players, and combat other teams in the matches.

Each player has an Unique ID, which is represented by their names. They play a certain role and get a ranking in each game. There are important dates related to players, date of birth and date of join.

Each match has a specific date, result MVP and the map of that match.

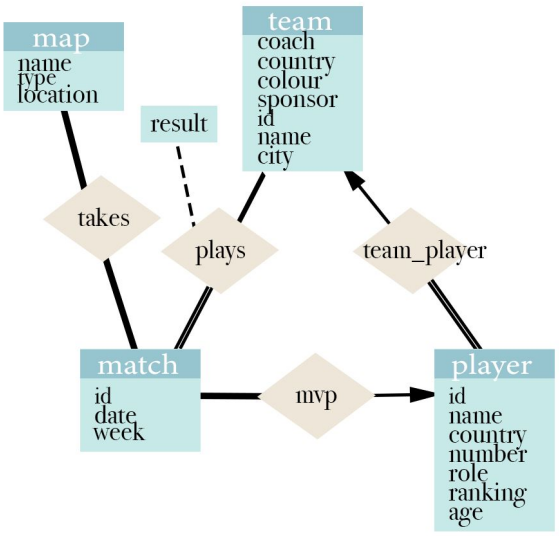
Each map has attributes about its name and location, and it's classified to a certain type.

There are relationship attributes between different entities.

This can be illustrated by Entity-Relationship Diagrams, which is presented in part

2. Conceptual Design

We have four entities, they are map, team, match and player. The attributes of maps are name, type and location. Map is the map pool of matches. And matches take in maps. Match has 3 attributes. Every match has their id, date and the week of the season. The relation between match and player is mvp. Every match has a mvp. The attributes of players are id, name, country, number, role, ranking and age. Players belong to teams. Team has 7 attributes: couch, country, colour, sponsor, id, name and city. Each match has 2 teams played. And then there will be a result of each plays.



3. Logical Design

3.1. Relation Schemas

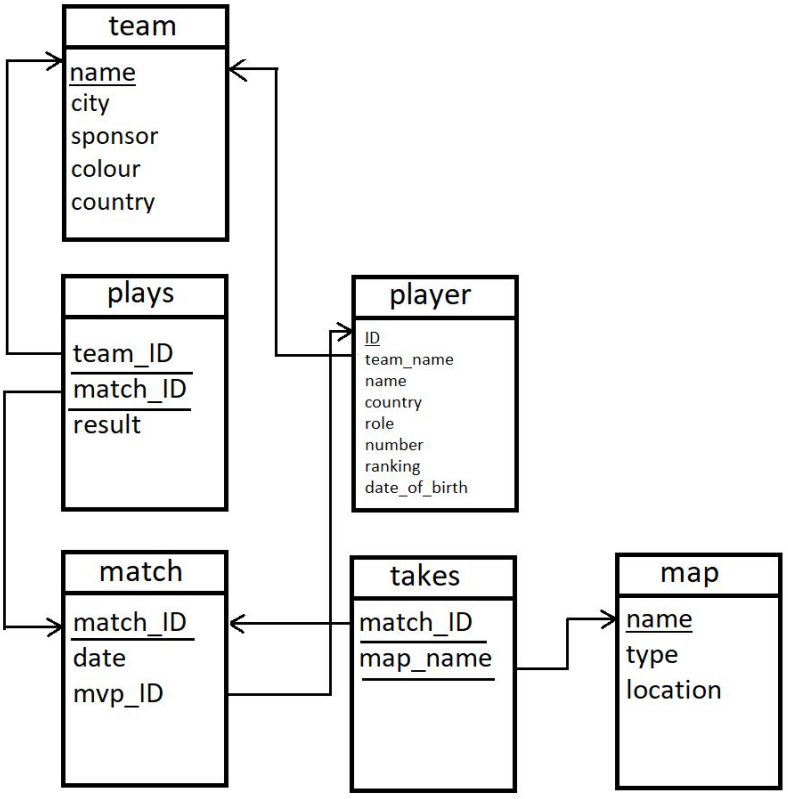
team (name, city, sponsor, colour, country)
player (ID, team.name NOT NULL, name, country, number, role, ranking, date_of_birth) foreign key team.ID references team on delete cascade
map (name, type, location)
match (ID, date, week, mvp.ID) foreign key mvp.ID references player on delete set null
plays(team.ID, match.ID NOT NULL, result) foreign key team.ID references team on delete set null, foreign key match.ID references match on delete cascade
takes(match.ID, map.name) foreign key match.ID references match on delete cascade, foreign key map.ID references map on delete cascade

3.2. Discussion

Player has total participation with the team and the relation between them is many-to-one. So instead of having a team_player relation, the player's relation schema has a not null foreign key: team.name, referencing the name attribute of the team.
Age is not a proper attribute to hold in a database since it changes every year. So we replaced it with date_of_birth.

Team actually has a name as an identifier attribute so we do not need an ID. Hence, ID is removed and name is the primary key.
The mvp relation between match and player is many-to-one. So we prefered match to have an attribute called mvp.ID which is a foreign key referencing player.
We have two relation schemas of relation sets: takes, plays. They have primary keys of their entity sets as foreign keys. The reason for the not null constraint in match.ID of plays is total participation of match in plays relation.

3.3. Relation Diagram



4. Normalization

Let's analyze tables one by one.

4.1. Team(name, city, sponsor, colour, country)

All values are atomic, and the primary key is only one attribute, so it is in 2NF. You can think that the country depends on the name via the city which is a

violation for 3NF. However, there are cities with the same name in different countries. For example, Perth in Australia and Scotland.

4.2. player(ID, team.name, name, country, number, role, ranking, date_of_birth) foreign key team.ID references team

All values are atomic, and the primary key is only one attribute, so it is in 2NF. Since there is no dependency between other attributes, it is also in 3NF already.

4.3. Map(name, type, location)

All values are atomic, and the primary key is only one attribute, so it is in 2NF. Since there is no dependency between other attributes, it is also in 3NF already.

4.4. Match(ID, date, week, mvp.ID) foreign key mvp.ID references player

All values are atomic, and the primary key is only one attribute, so it is in 2NF. Week depends on the date which is a violation for 3NF.
Normalization:

Match3NF(ID, date, mvp.ID) foreign key mvp.ID references player foreign key date references Date_Week

Date_Week(date, week)

4.5. Plays(team.name, match.ID, result) foreign key team.ID references team, foreign key match.ID references match

All values are atomic, so it is in 1NF. Result is dependent on both match.ID and team.ID because the result is win or not. It is in 2NF. Since there is only one non-primary attribute it is also in 3NF.

4.6. Takes(match.ID, map.name) foreign key match.ID references match, foreign key map.name references map

All values are atomic, and the primary key consists of all attributes, so it is in 2NF. Since there is no non-primary attribute it is also in 3NF.

5. Implementation

Below you can find the sql statements used to create tables and views.

```
DROP TABLE IF EXISTS Plays;
DROP TABLE IF EXISTS Takes;
DROP TABLE IF EXISTS Match_of_Teams;
DROP TABLE IF EXISTS Player;
DROP TABLE IF EXISTS Date_Week;
DROP TABLE IF EXISTS Team;
DROP TABLE IF EXISTS Map;
Drop View IF EXISTS AGE;


Create Table Team (
team_name VARCHAR(50) primary key,
city VARCHAR(50),
sponsor VARCHAR(50),
colour VARCHAR(30),
country VARCHAR(50));

Create Table Player (
ID Varchar(50) Primary key ,
team_name VARCHAR(50) nOT nULL,
full_name Varchar(50),
country VARCHAR(50),
player_number INT,
player_role Enum("Support","DPS","Tank"),
ranking INT,
date_of_birth date,
foreign key (team_name) references Team(team_name) on delete cascade);

Create Table Map (
map_name Varchar(50) Primary key,
map_type varchar(50),
location varchar(50) );

Create Table Date_Week (
        date_of_play date Primary key,
   week_of_play int);

Create Table Match_Of_Teams (
ID INT Primary key Auto_Increment,
date_of_play date,
mvp_ID varchar(50),
```

```
foreign key (mvp_ID) references Player(ID) on delete set null,
foreign key (date_of_play) references Date_Week(date_of_play) on delete set null);

Create Table Plays(
match_ID INT NOT NULL,
team_name Varchar(50),
result Enum("Win","Lose"),
primary key (team_name, match_ID),
foreign key (team_name) references Team(team_name) on delete cascade,
foreign key (match_ID) references Match_Of_Teams(ID) on delete cascade);

Create Table Takes(
match_ID Int Not Null,
map_name Varchar(50),
primary key (match_ID, map_name),
foreign key (match_ID) references match_of_teams(ID) on delete cascade,
foreign key (map_name) references  map(map_name) on delete cascade);

CREATE VIEW Age AS
SELECT ID, TIMESTAMPDIFF(YEAR, date_of_birth,  CURDATE()) as Age
FROM Player;
```

6. Database Instance

6.1. Team

team_name	city	sponsor	colour	country
Boston Uprising	Boston	Gillette	Blue	USA
Dallas Fuel	Dallas	Jack in the Box	Black	USA
Houston Outlaws	Houston	T-Mobile	Green	USA
London Spitfire	London	Logitech G	Blue	United Kingdom
New York Excelsior	New York	T-Mobile	Black	USA
Philadelphia Fusion	Philadelphia	ARRIS	Yellow	USA
San Francisco Shock	San Francisco	NULL	Gray	USA
Seoul Dynasty	Seoul	SIDIZ	Black	South Korea
Shanghai Dragons	Shanghai	NULL	Red	China
NULL	NULL	NULL	NULL	NULL

6.2. Player

ID	team_name	full_name	country	player_number	player_role	ranking	date_of_birth
ado	Shanghai Dragons	Gihyeon Chon	South Korea	99	DPS	NULL	1999-03-16
aimgod	Boston Uprising	Minseok Kwon	South Korea	31	Support	NULL	1999-03-31
akm	Dallas Fuel	Dylan Bignet	France	10	DPS	NULL	1995-12-11
altering	Shanghai Dragons	Yage Cheng	China	13	Support	NULL	1994-03-22
anamo	New York Excelsior	Taesung Jung	South Korea	97	Support	NULL	1997-09-02
bani	Houston Outlaws	Christopher Benell	Canada	3	Support	NULL	1993-09-28
bdosin	London Spitfire	Seungtae Choi	South Korea	22	Support	NULL	1996-08-18
boombox	Philadelphia Fusion	Isaac Charles	United Kingdom	42	Support	NULL	1997-04-17
bunny	Seoul Dynasty	Junhyeok Chae	South Korea	8	DPS	NULL	1998-05-15
carpe	Philadelphia Fusion	Jaehyeok Lee	South Korea	18	DPS	4697	1998-10-14
closer	London Spitfire	Wonsik Jung	South Korea	4	Support	NULL	1998-05-16
coolmatt	Houston Outlaws	Matt Iorio	USA	69	Tank	NULL	1989-07-29
custa	Dallas Fuel	Scott Kennedy	Australia	10	Support	4707	1993-11-22
Danteh	San Francisco Shock	Dante Cruz	USA	6	DPS	79	1990-03-24
dayfly	Philadelphia Fusion	Jeonghwan Park	South Korea	10	Support	NULL	1999-01-15
diya	Shanghai Dragons	Weida Lu	China	0	DPS	NULL	1999-07-30
dreamka...	Boston Uprising	Jonathan Sanchez	USA	28	DPS	4373	1998-11-09
eqo	Philadelphia Fusion	Josue Corona	Israel	15	DPS	NULL	1999-05-10
fissure	London Spitfire	Chanhyung Baek	South Korea	1	Tank	NULL	1999-02-26
fiveking	Shanghai Dragons	Zhaoyu Chen	China	9	Support	NULL	1997-12-08
fleta	Seoul Dynasty	Byungsun Kim	South Korea	1	DPS	NULL	1999-09-02
fury	London Spitfire	Junho Kim	South Korea	19	Tank	NULL	1999-11-25
gamsu	Boston Uprising	Youngjin Noh	South Korea	1	Tank	NULL	1994-02-26
Gesture	London Spitfire	Jae-hee Hong	South Korea	7	Tank	NULL	1998-01-12
harryhook	Dallas Fuel	Jonathan Tejedo...	Spain	9	Support	4659	1991-08-10
hotba	Philadelphia Fusion	Hongjun Choi	South Korea	7	Tank	NULL	1999-11-30

6.3. Date_Week

date_of_play	week_of_play
2018-01-10	1
2018-01-11	1
2018-01-12	1
2018-01-13	1
2018-01-17	2
2018-01-18	2
2018-01-19	2
2018-01-20	2
2018-01-24	3
2018-01-25	3
2018-01-26	3
2018-01-27	3
2018-01-31	4
2018-02-01	4
2018-02-02	4
2018-02-03	4
2018-02-07	5
2018-02-08	5
2018-02-09	5
2018-02-10	5

6.4. Map

map_name	map_type	location
Blizzard World	Hybrid	USA
Dor Dorado	Escort	Mexico
Eichenwalde	Hybrid	Germany
Hanamura	Assault	Japan
Hollywood	Hybrid	USA
Horizon Lunar Colony	Assault	The Moon
Ilios	Control	Greece
Junkertown	Escort	Australia
King's Row	Hybrid	United Kingdom
Lijiang Tower	Control	China
Nepal	Control	Nepal
Numbani	Hybrid	Numbani
Oasis	Control	Egypt
Route 66	Escort	USA
Temple of Anubis	Assault	Egypt
Volskaya Industries	Assault	Russia
Watchpoint: Gibraltar	Escort	Base

6.5. Match_of_teams

ID	date_of_play	mvp_ID
1	2018-01-10	Fleta
2	2018-01-11	Carpe
3	2018-01-11	jjonak
4	2018-01-12	Danteh
5	2018-01-13	Gesture
6	2018-01-13	Jjonak
7	2018-01-17	Carpe
8	2018-01-17	Muma
9	2018-01-18	Jake
10	2018-01-19	Miro
11	2018-01-19	Profit
12	2018-01-20	Danteh
13	2018-01-24	Bdosin
14	2018-01-24	ryujeh...
15	2018-01-25	Gamsu
16	2018-01-25	Carpe
17	2018-01-26	Saeby...
18	2018-01-26	Carpe
19	2018-01-26	Mickie
20	2018-01-27	Dream...
21	2018-01-31	Muma
22	2018-02-01	ShaDo...
23	2018-02-01	Libero
24	2018-02-01	Gesture
25	2018-02-02	zunba
26	2018-02-03	Profit

6.6. Plays

match_ID	team_name	result
3	Boston Uprising	Lose
10	Boston Uprising	Lose
12	Boston Uprising	Lose
15	Boston Uprising	Win
20	Boston Uprising	Win
29	Boston Uprising	Win
33	Boston Uprising	Lose
1	Dallas Fuel	Lose
9	Dallas Fuel	Lose
11	Dallas Fuel	Lose
19	Dallas Fuel	Win
20	Dallas Fuel	Lose
22	Dallas Fuel	Lose
27	Dallas Fuel	Lose
28	Dallas Fuel	Win
2	Houston Outlaws	Lose
6	Houston Outlaws	Lose
8	Houston Outlaws	Win
9	Houston Outlaws	Win
21	Houston Outlaws	Win
25	Houston Outlaws	Lose
30	Houston Outlaws	Win
33	Houston Outlaws	Win
5	London Spitfire	Win
11	London Spitfire	Win
13	London Spitfire	Win

6.7. Takes

match_ID	map_name
1	Horizon Lunar Colony
1	Ilios
1	Junkertown
1	Numbani
2	Horizon Lunar Colony
2	Ilios
2	Junkertown
2	Numbani
3	Horizon Lunar Colony
3	Ilios
3	Junkertown
3	Numbani
4	Horizon Lunar Colony
4	Ilios
4	Junkertown
4	Numbani
5	Horizon Lunar Colony
5	Ilios
5	Junkertown
5	Numbani
6	Horizon Lunar Colony
6	Ilios
6	Junkertown
6	Numbani
7	Eichenwalde
7	Horizon Lunar Colony

7. SQL Data Queries

7.1. Order by

The scripts below select the winner/ loser of each match and then order them in an ascending order.

```
select match_ID, team_name as Winner from plays where result = 'win' order by match_ID;
select match_ID, team_name as Loser from plays where result = 'lose' order by match_ID;
```

The outputs of the queries are shown below.

match_ID	Winner
1	Seoul Dynasty
2	Philadelphia Fusion
3	New York Excelsior
4	San Francisco Shock
5	London Spitfire
6	New York Excelsior
7	Philadelphia Fusion
8	Houston Outlaws
9	Houston Outlaws
10	Seoul Dynasty
11	London Spitfire
12	San Francisco Shock
13	London Spitfire
14	Seoul Dynasty
15	Boston Uprising
16	Philadelphia Fusion
17	New York Excelsior
18	Philadelphia Fusion
19	Dallas Fuel
20	Boston Uprising
21	Houston Outlaws

match_ID	Loser
1	Dallas Fuel
2	Houston Outlaws
3	Boston Uprising
4	Shanghai Dragons
5	Philadelphia Fusion
6	Houston Outlaws
7	San Francisco Shock
8	Shanghai Dragons
9	Dallas Fuel
10	Boston Uprising
11	Dallas Fuel
12	Boston Uprising
13	San Francisco Shock
14	Shanghai Dragons
15	London Spitfire
16	New York Excelsior
17	Seoul Dynasty
18	Shanghai Dragons
19	San Francisco Shock
20	Dallas Fuel
21	San Francisco Shock

7.2. Group by

The script below selects the average, maximum and minimum value of the evaluation of players of different ages.

```
select age, AVG(ranking) as average_ranking, MAX(ranking) as maximum_ranking,
MIN(ranking) as minimum_ranking from Player natural join Age group by age;
```

The outputs of the queries are shown below.

Age	average_ranking	maximum_ranking	minimum_ranking
20	4671.5000	4724	4619
21	4446.5000	4697	4343
22	4640.5000	4690	4591
23	4603.0000	4666	4540
24	4645.0000	4645	4645
25	4635.5000	4676	4595
26	4536.5000	4707	4373
27	4468.0000	4468	4468
28	4653.0000	4659	4647
29	NULL	NULL	NULL
30	79.0000	79	79

Since there is no record of players whose age is 29, the values of 29-year-old players in this table are NULL.

7.3. Join

The scripts below select the natural join results of Player, Team and Age, which provide users an integral view of the players, including their teams and constantly updated ages.

```
select * from Player natural join Team natural join Age;
```

The outputs of the queries are shown below.

ID	team_name	country	full_name	player_number	player_role	ranking	date_of_birth	city	sponsor	colour	Age
dreamkazper	Boston Uprising	USA	Jonathan Sanchez	28	DPS	4373	1998-11-09	Boston	Gillette	Blue	21
seagull	Dallas Fuel	USA	Brandon Larned	7	DPS	4468	1992-07-28	Dallas	Jack in the Box	Black	27
coolmatt	Houston Outlaws	USA	Matt Iorio	69	Tank	NULL	1989-07-29	Houston	T-Mobile	Green	30
jake	Houston Outlaws	USA	Jacob Lyon	76	DPS	4540	1996-07-28	Houston	T-Mobile	Green	23
muma	Houston Outlaws	USA	Austin Wilmot	17	Tank	NULL	1998-08-17	Houston	T-Mobile	Green	21
rawkus	Houston Outlaws	USA	Shane Flaherty	1	Support	4676	1994-06-16	Houston	T-Mobile	Green	25
Danteh	San Francisco Shock	USA	Dante Cruz	6	DPS	79	1990-03-24	San Francisco	NULL	Gray	30
bunny	Seoul Dynasty	South Korea	Junhyeok Chae	8	DPS	NULL	1998-05-15	Seoul	SIDIZ	Black	21
fleta	Seoul Dynasty	South Korea	Byungsun Kim	1	DPS	NULL	1999-09-02	Seoul	SIDIZ	Black	20
miro	Seoul Dynasty	South Korea	Jinhyuk Gong	3	Tank	4645	1996-02-26	Seoul	SIDIZ	Black	24
ryujehong	Seoul Dynasty	South Korea	Jehong Ryu	14	Support	4647	1991-09-05	Seoul	SIDIZ	Black	28
tobi	Seoul Dynasty	South Korea	Jinmo Yang	4	Support	4373	1994-02-26	Seoul	SIDIZ	Black	26
zunba	Seoul Dynasty	South Korea	Joonhyeok Kim	5	Tank	4591	1997-10-29	Seoul	SIDIZ	Black	22
altering	Shanghai Dragons	China	Yage Cheng	13	Support	NULL	1994-03-22	Shanghai	NULL	Red	26
diya	Shanghai Dragons	China	Weida Lu	0	DPS	NULL	1999-07-30	Shanghai	NULL	Red	20
fiveking	Shanghai Dragons	China	Zhaoyu Chen	9	Support	NULL	1997-12-08	Shanghai	NULL	Red	22
roshan	Shanghai Dragons	China	Wenhao Jing	26	Tank	NULL	1997-05-11	Shanghai	NULL	Red	22
xushu	Shanghai Dragons	China	Junjie Liu	66	Tank	NULL	1995-01-01	Shanghai	NULL	Red	25

8. SQL Table Modifications

8.1. Update

The following scripts show when a team changed their colour or their sponsor. First scripts and tables show London Spitfire changed their colour from blue to Azure. And the second one shows Shanghai Dragons changing their sponsor to Razer.

```
UPDATE team SET colour = 'Azure'
WHERE team_name = 'London Spitfire';
```

```
UPDATE team SET sponsor = 'Razer'
WHERE team_name = 'Shanghai Dragons';
```

The table before using UPDATE

team_name	city	sponsor	colour	country
Boston Uprising	Boston	Gillette	Blue	USA
Dallas Fuel	Dallas	Jack in the Box	Black	USA
Houston Outlaws	Houston	T-Mobile	Green	USA
London Spitfire	London	Logitech G	Blue	United Kingdom
New York Excelsior	New York	T-Mobile	Black	USA
Philadelphia Fusion	Philadelphia	ARRIS	Yellow	USA
San Francisco Shock	San Francisco	NULL	Gray	USA
Seoul Dynasty	Seoul	SIDIZ	Black	South Korea
Shanghai Dragons	Shanghai	NULL	Red	China

After

team_name	city	sponsor	colour	country
Boston Uprising	Boston	Gillette	Blue	USA
Dallas Fuel	Dallas	Jack in the Box	Black	USA
Houston Outlaws	Houston	T-Mobile	Green	USA
London Spitfire	London	Logitech G	Azure	United Kingdom
New York Excelsior	New York	T-Mobile	Black	USA
Philadelphia Fusion	Philadelphia	ARRIS	Yellow	USA
San Francisco Shock	San Francisco	NULL	Gray	USA
Seoul Dynasty	Seoul	SIDIZ	Black	South Korea
Shanghai Dragons	Shanghai	Razer	Red	China

team_name	city	sponsor	colour	country
Boston Uprising	Boston	Gillette	Blue	USA
Dallas Fuel	Dallas	Jack in the Box	Black	USA
Houston Outlaws	Houston	T-Mobile	Green	USA
London Spitfire	London	Logitech G	Azore	United Kingdom
New York Excelsior	New York	T-Mobile	Black	USA
Philadelphia Fusion	Philadelphia	ARRIS	Yellow	USA
San Francisco Shock	San Francisco	NULL	Gray	USA
Seoul Dynasty	Seoul	SIDIZ	Black	South Korea
Shanghai Dragons	Shanghai	Razer	Red	China

8.2. Delete

The script below shows deleting a player from the player list when he is retired or leaving the team.

```
DELETE FROM player
WHERE ID = 'XQC';
```

The table before using UPDATE

ID	team_name	full_name	country	player_number	player_role	ranking	date_of_birth
zunba	Seoul Dynasty	Joonhyeok Kim	South Korea	5	Tank	4591	1997-10-29
xushu	Shanghai Dragons	Junjie Liu	China	66	Tank	NULL	1995-01-01
XQC	Dallas Fuel	Félix Lengyel	Canada	12	Tank	NULL	1995-11-12
tobi	Seoul Dynasty	Jinmo Yang	South Korea	4	Support	4373	1994-02-26
taimou	Dallas Fuel	Timo Kettunen	Finland	13	DPS	4598	1993-08-30
striker	Boston Uprising	Namjoo Kwon	South Korea	7	DPS	NULL	1990-12-04
Shadow...	Philadelphia Fusion	George Gushcha	Russia	18	DPS	4690	1997-04-24
seagull	Dallas Fuel	Brandon Larned	USA	7	DPS	4468	1992-07-28

After

ID	team_name	full_name	country	player_number	player_role	ranking	date_of_birth
zunba	Seoul Dynasty	Joonhyeok Kim	South Korea	5	Tank	4591	1997-10-29
xushu	Shanghai Dragons	Junjie Liu	China	66	Tank	NULL	1995-01-01
tobi	Seoul Dynasty	Jinmo Yang	South Korea	4	Support	4373	1994-02-26
taimou	Dallas Fuel	Timo Kettunen	Finland	13	DPS	4598	1993-08-30
striker	Boston Uprising	Namjoo Kwon	South Korea	7	DPS	NULL	1990-12-04
Shadow...	Philadelphia Fusion	George Gushcha	Russia	18	DPS	4690	1997-04-24
seagull	Dallas Fuel	Brandon Larned	USA	7	DPS	4468	1992-07-28
saebye...	New York Excelsior	Jongryeol Park	South Korea	9	DPS	NULL	1995-12-23

9. SQL Programming

9.1. Functions

The scripts below define a function which counts the number of winning/lost games.

```
DELIMITER //
CREATE FUNCTION count_winning(vTeam_name varchar(50)) RETURNS INT
Begin
    DECLARE vWinningCount INT;
    SELECT COUNT(*) INTO vWinningCount FROM team NATURAL JOIN plays
    WHERE (team_name = vTeam_name) and (result = 'Win');
    RETURN vWinningCount;
END;//
DELIMITER ;
```

```
DELIMITER //
CREATE FUNCTION count_lose(vTeam_name varchar(50)) RETURNS INT
Begin
    DECLARE vLoseCount INT;
    SELECT COUNT(*) INTO vLoseCount FROM team NATURAL JOIN plays
    WHERE (team_name = vTeam_name) and (result = 'Lose');
    RETURN vLoseCount;
END;//
DELIMITER ;
```

After the functions are defined, they can be called using the select commands.

One example for each function is shown below (Command scripts and results)

①

```
SELECT 'Boston Uprising' as 'team_name', count_winning('Boston Uprising') as 'winning_count';
```

team_name	winning_count
Boston Uprising	3

②

```
SELECT 'Boston Uprising' as 'team_name', count_lose('Boston Uprising') as 'lost_count';
```

team_name	lost_count
Boston Uprising	4

By calling these two functions, users can easily get to know the numbers of winning and lost games of a specific team.

9.2. Procedures

The scripts below define a procedure that takes a player name as input, and store the nationality of the player to the output variable.

```
DELIMITER //
CREATE PROCEDURE GetNationality
(IN vplayername VARCHAR(50), OUT vNationality VARCHAR(50))
BEGIN
    SELECT Country INTO vNationality FROM Player
    WHERE ID = vplayername;
end;//
DELIMITER ;
```

By calling the procedure and then selecting the output variable, users can get the nationality of a specific player. Here is an example (Scripts and results).

```
CALL GetNationality('tobi', @Nationality);
SELECT 'tobi' AS player_name, @Nationality AS Nationality;
```

player_name	Nationality
tobi	South Korea

9.3. Transactions

The scripts below define a procedure that includes a transaction. It takes two IN parameters. These parameters are the IDs of the players to swap teams. The OUT parameter is the status of the transaction.

```
DELIMITER //
CREATE PROCEDURE player_trade (
IN vPlayer1 varchar(50), vPlayer2 varchar(50), OUT vStatus VARCHAR(45))
BEGIN
DECLARE OldTeam1, NewTeam1, OldTeam2, NewTeam2 varchar(50);
START TRANSACTION;
SET SQL_SAFE_UPDATES = 0;
SET OldTeam1 = (SELECT team_name FROM player WHERE id = vPlayer1);
SET OldTeam2 = (SELECT team_name FROM player WHERE id = vPlayer2);
Set NewTeam1 = OldTeam2;
Set NewTeam2 = OldTeam1;
UPDATE Player SET team_name = NewTeam1 WHERE ID = vPlayer1;
UPDATE Player SET team_name = NewTeam2 WHERE ID = vPlayer2;
IF NewTeam1=OldTeam2 and NewTeam2=OldTeam1
THEN SET vStatus = 'Transaction Trade committed!'; COMMIT;
ELSE SET vStatus = 'Transaction Trade rollback'; ROLLBACK;
END IF;
END; //
DELIMITER ;
```

Below you can find the example usage of this procedure.

Before calling the transcation procedure "player_trade":

```
Select id, team_name from player where player.id="jake" or player.id="tobi";
```

id	team_name
jake	Houston Outlaws
tobi	Seoul Dynasty

After calling the transaction procedure "player_trade":

```
Call player_trade("jake", "tobi",@tradestatus);
Select @tradestatus
```

@tradestatus
Transaction Trade committed!

```
Select id, team_name from player where player.id="jake" or player.id="tobi";
```

id	team_name
jake	Seoul Dynasty
tobi	Houston Outlaws

9.4. Triggers

```
DELIMITER //
CREATE TRIGGER Plays_before_Insert
Before INSERT ON Plays FOR EACH ROW
BEGIN
declare number_of_teams int;
Set number_of_teams = (select count(*) from plays where match_id = new.match_id);
IF number_of_teams > 1 then
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Only two teams can play a match.';
End IF;
END; //
DELIMITER ;

Insert into Plays Values (1, "Seoul Dynasty", "Lose");
Insert into Plays Values (2, "Boston Uprising", "Win");
```

Here are the triggers and some illegal queries against this trigger. Whenever you try to add a team to **plays**, it checks the database with the same id, if there are more than 1 teams playing this match it gives an error:

```
181  18:50:30  Insert into Plays Values (1, "Seoul Dynasty", "Lose")   Error Code: 1644. Only two teams can play a match.
182  18:50:32  Insert into Plays Values (2, "Boston Uprising", "Win")  Error Code: 1644. Only two teams can play a match.
```

9.5. Events

Every player's ranking will be re-calculated in a new season. So the scripts below show after each season (one month), the player's ranking will be set to null.

```
SHOW VARIABLES LIKE 'event_scheduler';

SET GLOBAL event_scheduler = 1;

CREATE EVENT changeranking
    ON SCHEDULE
      EVERY 1 MONTH
      DO
      UPDATE player SET player_ranking = NULL ;
```

Variable_name	Value
event_scheduler	ON

```
143  21:57:27  CREATE EVENT changeranking  ON SCHEDULE  EVERY 1 MONTH  DO...  0 row(s) affected  0.011 sec
```