

Kenny Mejia

Database Design Project

Data Dictionary

2018-11-29

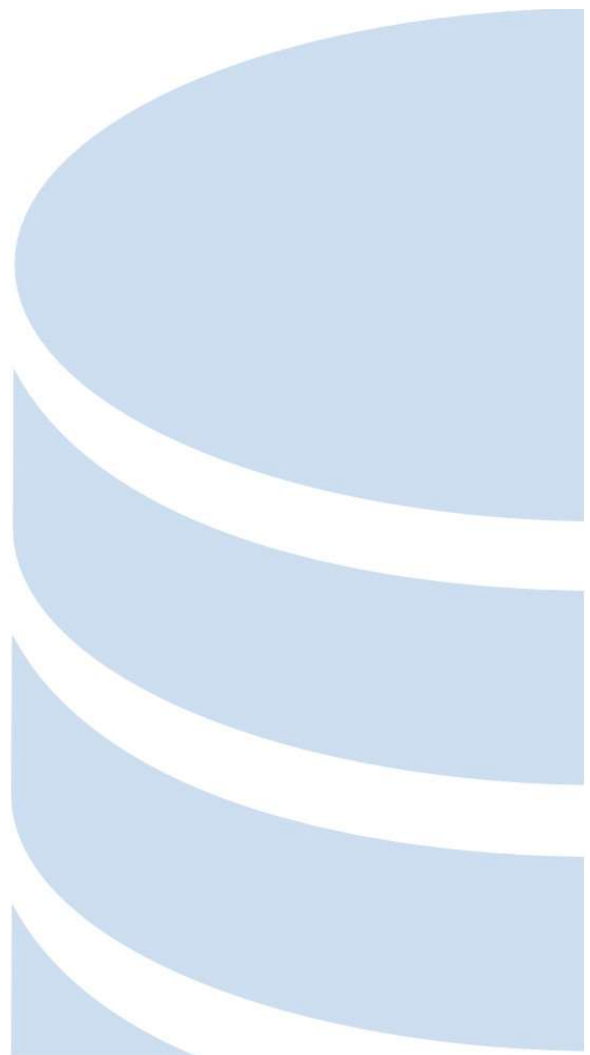























Table of contents

Executive Summary	4
Objectives	4
Future Enhancements	5
Known Problems	6
Entity Relational Diagram	7
1. Tables	8
1.1. Table: BRACKET (Bracket Table)	8
1.2. Table: BRACKETPLACEMENT (Bracket Placement Table)	9
1.3. Table: DEVELOPER (Developer Table)	10
1.4. Table: GAME (Game Table)	11
1.5. Table: GAMEMODE (Game Mode Table)	12
1.6. Table: MATCH (Match Table)	13
1.7. Table: PLAYER (Player Table)	14
1.8. Table: PLAYERRANKING (Player Ranking Table)	15
1.9. Table: REFEREE (Referee Table)	17
1.10. Table: REFEREES (Referees Table)	18
1.11. Table: SPONSOR (Sponsor Table)	19
1.12. Table: SPONSORSHIPS (Sponsorships Table)	20
1.13. Table: TEAM (Team Table)	21
1.14. Table: TEAMROSTER (Team Roster Table)	22
2. Sample Queries And Output	23

Legend

-  Primary key
-  Primary key disabled
-  User-defined primary key
-  Unique key
-  Unique key disabled
-  User-defined unique key
-  Active trigger
-  Disabled trigger
-  Many to one relation
-  User-defined many to one relation
-  One to many relation
-  User-defined one to many relation
-  One to one relation
-  User-defined one to one relation
-  Input
-  Output
-  Input/Output
-  Uses dependency
-  User-defined uses dependency
-  Used by dependency
-  User-defined used by dependency

Executive Summary

The following documentation displays a database that could be implemented for an esports ready video game. Esports also known as electronic sports or competitive/professional video gaming is a form of competition using video games. Most commonly, esports are played between professional players not just for bragging rights, but many also compete for large sums of money and just as important, fame and recognition. Starting in the late 2000's participation by professional gamers and spectartorship in esports events began to rise thanks to live streaming the events on platforms such as Twitch and YouTube. Esports finally began to become a significant factor in the video game industry as more and more game developers began focusing on the professional esports subculture. The most common genres that take on the form of esports are mainly fighting, FPS (First Person Shooter), RTS (Real Time Strategy), and MOBA (Multiplayer Online Battle Arena). Many competitions use promotions and relegation play with professional teams that are sponsored to take part in said competitions. The legitimacy of esports as a sports competition is still questioned but even organizations such as the International Olympic Committee has explored incorporating esports into future events. In this project proposal we will be taking a look at how a database could be incorporated into esports competitions and we will be using the popular MOBA League Of Legends or LOL as the game we will be building this database for. Future enhancements are also being considered which will be implemented as the database matures

Objectives

- Organize information efficiently
- Query the database for information that is relevant to the enterprise at the moment it is being accessed
- Provide a detailed description of the video game and game mode being used in the competition
- Provide a detailed description of each player participating in the current competition

- Provide a detailed description of each team participating in the current competition
- Provide detailed information on each match of the competition
- Provide brackets and standings as the competition unfolds
- Provide information on the referees that will be upholding the competitions rule set
- Provide information on each teams sponsorships

Future Enhancements

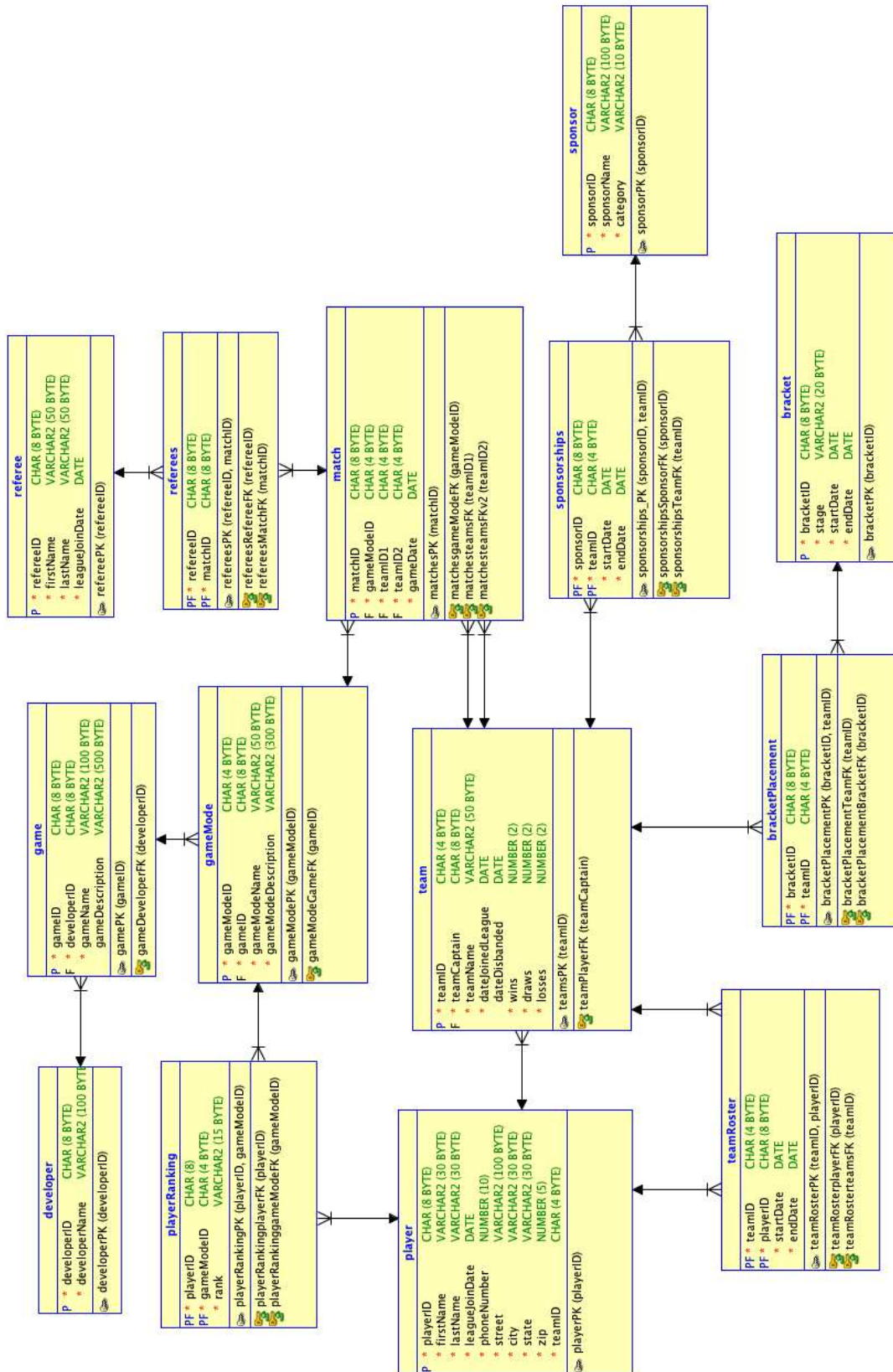
In popular games such as Blizzard's Overwatch and Riot Games' League Of Legends the characters chosen make all the difference when it comes to winning a match especially in a tournament. For example in the MOBA League Of Legends before characters are selected for a match the two teams ban a total of ten characters who they feel would pose a problem to their winning strategies. As a result the banning and selection of characters is just as important as the match itself. For this reason adding a table for characters chosen would benefit when wanting to go back and see what characters were chosen for a specific match. This would also greatly benefit the game developers as they could query the database to see which characters were popular picks and which characters were ignored. If the game developers use this information to balance "broken" (overpowered) characters, this would lead to a happier community of gamers that would enjoy going back to a well balanced game rather than an angry and upset community that would lead to "toxicity" within the community due to poor game balancing. This table would have another table extending to it, which would allow for information such as "kills" and "deaths" to be inserted by a per player basis. In other words we would be able to see a breakdown of how each player did in terms of "kills and deaths". With this information teams will be able to see how each member performed allowing for better planning in future matches, which would lead to a higher possibility of winning. With this same information game developers would be able to see how each character performed in each match. By comparing the performance of players using the same character in different matches, they would be able to see if the character needs to be adjusted for game balancing. With these enhancements this esports database will not only benefit the competition it is being used for but it will also benefit the game developers by allowing them to see what adjustments need to be made to the game in order for gamers to have an enjoyable

experience.

Known Problems

This esports database is based on the direction that current esports tournaments have been taking over the last few years, meaning that this database is not meant to handle thousands of teams at a time. The best example of this would be the MLG or Major League Gaming model where hundreds or even thousands of teams sign up to be apart of one tournament. Your opponents are chosen at random and when the current round is over the teams with the most wins move onto the next round where the same process takes place until the finals are reached. By contrast my model is based on small seasonal matches based on brackets where teams in the same brackets face off against each other until the teams in the same bracket have faced each other more than once. Then the teams with the most wins continue onto the next bracket, this process continues until the finals are reached. MLG does not rely on referees like in my model instead they rely on the "honor system" where after the match has been completed both teams submit whether they won or lost the match. Many teams would not mind reporting false wins which is why proof is also submitted along with the report. This proof could consist of a simple screen capture or a complete replay of the match that took place. My model relies on the presence of official referees attending the matches being played. On the next iteration this database will be able to handle the more intense MLG model by allowing more teams to compete at random rather than strictly being based on brackets. We could also allow longer identification numbers so that the chances of them being duplicated or even the chances of running out of unique identification numbers does not become a concern.

Entity Relationship Diagram







1. Tables

1.1. Table: BRACKET (Bracket Table)

```
CREATE TABLE bracket (  
  bracketid CHAR(8 BYTE) NOT NULL,  
  stage VARCHAR2(20 BYTE) NOT NULL,  
  startdate DATE NOT NULL,  
  enddate DATE NOT NULL  
);  
ALTER TABLE bracket ADD CONSTRAINT bracketpk PRIMARY KEY ( bracketid );
```

Each team will be put into different brackets at the beginning of the season much like professional football and baseball. Each bracket will have a unique identification number that will help differentiate each bracket. The stage column will tell us in what stage of the season we are in such as qualifiers, semifinals, and finals. The table also has two date columns one for when the bracket starts taking effect and one for the date that the winner of each bracket is determined.

Columns



Name		Data type	Description / Attributes
	BRACKETID	CHAR(8 BYTE)	Unique identification number given to each bracket
	STAGE	VARCHAR2(20 BYTE)	stage of the competition that each bracket is in such as qualifiers, semi finals, and finals
	STARTDATE	DATE	date that the bracket takes effect
	ENDDATE	DATE	date that the team with the most wins is deemed the winner of the bracket

1.2. Table: BRACKETPLACEMENT (Bracket Placement Table)

```
CREATE TABLE bracketplacement (  
    bracketid CHAR(8 BYTE) NOT NULL,  
    teamid CHAR(4 BYTE) NOT NULL  
);  
  
ALTER TABLE bracketplacement ADD CONSTRAINT bracketplacementpk PRIMARY KEY ( bracketid,  
teamid );  
  
ALTER TABLE bracketplacement  
    ADD CONSTRAINT bracketplacementbracketfk FOREIGN KEY ( bracketid )  
        REFERENCES bracket ( bracketid );  
  
ALTER TABLE bracketplacement  
    ADD CONSTRAINT bracketplacementteamfk FOREIGN KEY ( teamid )  
        REFERENCES team ( teamid );
```

The bracket placement table exists between the team table and the bracket table so that we may be able to associate one table with the other. Teams will be placed in different brackets as the competition progresses, so the further a team progresses through the competition the higher the bracket that they will be placed in. Many teams can belong to one bracket and one team can belong to many brackets. The table consists of two primary foreign keys being bracketID and teamID so that we may have unique rows being returned when querying data out of the database.

Columns



Name		Data type	Description / Attributes
	BRACKETID	CHAR(8 BYTE)	Unique identification number given to each bracket References: BRACKET
	TEAMID	CHAR(4 BYTE)	Each teams unique clan tag (name abbreviation) References: TEAM

1.3. Table: DEVELOPER (Developer Table)

```
CREATE TABLE developer (  
    developerid CHAR(8 BYTE) NOT NULL,  
    developername VARCHAR2(100 BYTE) NOT NULL  
);  
ALTER TABLE developer ADD CONSTRAINT developerpk PRIMARY KEY ( developerid );
```

Without game developers the games that thousands compete in and enjoy every year would not exist. Games are often created by one developer and then sold to another, this is why the developer table exists outside the game table. If a game was bought by another developer, then only the developerID field inside the game table would need to be updated. The game developer table consists of an developerID and a developerName field. The developerID is the primary key and is used to give each developer a unique identification number to differentiate each developer from one another. The developerName field, as it sounds, is used to hold the name of the developer.

Columns





Name		Data type	Description / Attributes
	DEVELOPERID	CHAR(8 BYTE)	Unique identification number given to each game developer
	DEVELOPERNAME	VARCHAR2(100 BYTE)	Game developers name

1.4. Table: GAME (Game Table)

```
CREATE TABLE game (  
  gameid          CHAR(8 BYTE) NOT NULL,  
  developerid     CHAR(8 BYTE) NOT NULL,  
  gamename        VARCHAR2(100 BYTE) NOT NULL,  
  gamedescription VARCHAR2(500 BYTE)  
);  
  
ALTER TABLE game ADD CONSTRAINT gamepk PRIMARY KEY ( gameid );  
  
ALTER TABLE game  
  ADD CONSTRAINT gamedeveloperfk FOREIGN KEY ( developerid )  
    REFERENCES developer ( developerid );
```

The game table describes the game used during the competition. Each game has multiple game modes and depending on the competition there may be more than game mode being used during the competition, a custom game mode with custom rules may even be used. For this reason we do not have a game mode field inside the game table. The gameId, which is a unique identification number given to each game is the primary key for the table. The developerID is a foreign key to the developer table's developerID. Games are often created by one developer and then sold to another, this is why the developerID as a foreign key, so that updates may be easier to carry out. The gameName field would hold the name of the game and the gameDescription table would hold an optional description of the game.

Columns





Name		Data type	Description / Attributes
	GAMEID	CHAR(8 BYTE)	Unique identification number given to each game
	DEVELOPERID	CHAR(8 BYTE)	Unique identification number given to each developer References: DEVELOPER
	GAMENAME	VARCHAR2(100 BYTE)	Name of the game being used during the competition
	GAMEDESCRIPTION	VARCHAR2(500 BYTE)	Optional description of the game, the description may include number of active players, ESRB rating, and genre Nullable

1.5. Table: GAMEMODE (Game Mode Table)

```
CREATE TABLE gamemode (  
  gamemodeid      CHAR(4 BYTE) NOT NULL,  
  gameid          CHAR(8 BYTE) NOT NULL,  
  gamemodename    VARCHAR2(50 BYTE) NOT NULL,  
  gamemodedescription VARCHAR2(300 BYTE) NOT NULL  
);  
  
ALTER TABLE gamemode ADD CONSTRAINT gamemodepk PRIMARY KEY ( gamemodeid );  
  
ALTER TABLE gamemode  
  ADD CONSTRAINT gamemodegamefk FOREIGN KEY ( gameid )  
    REFERENCES game ( gameid );
```

The game mode table is made up of 4 fields, the primary key is the gameModeID which is a unique identification number given to each game mode. The gameID is a foreign key to the game table's gameID, this is because some games may have the same game modes such as basic three versus three or five versus five games. GameModeName would contain the name of the game mode and gameModeDescription would describe the game mode being played in case special rules were added for the competition.

Columns






Name		Data type	Description / Attributes
	GAMEMODEID	CHAR(4 BYTE)	Unique identification number given to each game mode
	GAMEID	CHAR(8 BYTE)	Unique identification number given to each game References: GAME
	GAMEMODENAME	VARCHAR2(50 BYTE)	Name of the game mode being played
	GAMEMODEDESCRIPTION	VARCHAR2(300 BYTE)	Description of the game mode, special rules may be defined here if special rules are used

1.6. Table: MATCH (Match Table)

```
CREATE TABLE match (  
  matchid    CHAR(8 BYTE) NOT NULL,  
  gamemodeid CHAR(4 BYTE) NOT NULL,  
  teamid1    CHAR(4 BYTE) NOT NULL,  
  teamid2    CHAR(4 BYTE) NOT NULL,  
  gamedate   DATE NOT NULL  
);  
  
ALTER TABLE match ADD CONSTRAINT matchespk PRIMARY KEY ( matchid );  
  
ALTER TABLE match  
  ADD CONSTRAINT matchesgamemodefk FOREIGN KEY ( gamemodeid )  
    REFERENCES gamemode ( gamemodeid );  
  
ALTER TABLE match  
  ADD CONSTRAINT matchsteamskfk FOREIGN KEY ( teamid1 )  
    REFERENCES team ( teamid );  
  
ALTER TABLE match  
  ADD CONSTRAINT matchsteamskv2 FOREIGN KEY ( teamid2 )  
    REFERENCES team ( teamid );
```

A match in the competition would consist of two teams playing a game mode in which the winner may be decided after a certain amount of wins. It is possible for teams to face each other more than once which is why we use the matchID as a primary key to differentiate each match. We also have a date field so that we can see when each match took place. GameModeID is a foreign key to the gameModeID in the gameMode table, the field would tell us what kind of game mode was being used during the match as different game modes may be used during each match. The last two fields teamID1 and teamID2 would tell us which teams are facing each other during the match, both are foreign keys to the teamID in the team table.

Columns











	Name	Data type	Description / Attributes
	MATCHID	CHAR(8 BYTE)	Unique identification number given to each match
	GAMEMODEID	CHAR(4 BYTE)	Unique identification number given to each game mode References: GAMEMODE
	TEAMID1	CHAR(4 BYTE)	Unique identification number given to each team References: TEAM
	TEAMID2	CHAR(4 BYTE)	Unique identification number given to each team References: TEAM
	GAMEDATE	DATE	The date that each match took place

1.7. Table: PLAYER (Player Table)

```
CREATE TABLE player (  
  playerid      CHAR(8 BYTE) NOT NULL,  
  firstname     VARCHAR2(30 BYTE) NOT NULL,  
  lastname      VARCHAR2(30 BYTE) NOT NULL,  
  leaguejoindate DATE NOT NULL,  
  phonenumber    NUMBER(10) NOT NULL,  
  street        VARCHAR2(100 BYTE) NOT NULL,  
  city          VARCHAR2(30 BYTE) NOT NULL,  
  state         VARCHAR2(30 BYTE) NOT NULL,  
  zip           NUMBER(5) NOT NULL,  
  teamid        CHAR(4 BYTE) NOT NULL  
);  
ALTER TABLE player ADD CONSTRAINT playerpk PRIMARY KEY ( playerid );
```

The player table describes every aspect of the person taking part in the competition. Each player is given a unique player identification number represented by playerID, which is the primary key for the player table. When each player registers for the season they must give their first name, last name, as well as their address which is further broken down by street, city, state, and zip. The player must also register a phone number so they may be reached when important matters arise concerning the player throughout the competition. Each player has their team's identification associated with them so that we can see which players belong to a certain team.

Columns

Name		Data type	Description / Attributes
	PLAYERID	CHAR(8 BYTE)	Unique identification number given to each player
	FIRSTNAME	VARCHAR2(30 BYTE)	First name of the player
	LASTNAME	VARCHAR2(30 BYTE)	Last name of the player
	LEAGUEJOINDATE	DATE	Date that the player joined the league
	PHONENUMBER	NUMBER(10, 0)	The players phone number
	STREET	VARCHAR2(100 BYTE)	The players street address
	CITY	VARCHAR2(30 BYTE)	The players city of residence
	STATE	VARCHAR2(30 BYTE)	The players state of residence
	ZIP	NUMBER(5, 0)	The players zip code
	TEAMID	CHAR(4 BYTE)	The team that the player belongs to






1.8. Table: PLAYERRANKING (Player Ranking Table)

```
CREATE TABLE playerranking (  
  playerid CHAR(8) NOT NULL,  
  gamemodeid CHAR(4 BYTE) NOT NULL,  
  rank VARCHAR2(15 BYTE) NOT NULL  
);  
ALTER TABLE playerranking ADD CONSTRAINT playerrankingpk PRIMARY KEY ( playerid, gamemodeid  
);  
  
ALTER TABLE playerranking  
  ADD CONSTRAINT playerrankinggamemodefk FOREIGN KEY ( gamemodeid )  
    REFERENCES gamemode ( gamemodeid );  
  
ALTER TABLE playerranking  
  ADD CONSTRAINT playerrankingplayerfk FOREIGN KEY ( playerid )  
    REFERENCES player ( playerid );
```

In games such as the popular MOBA League Of Legends each player is given a rank consisting of a tier ranging from iron (being the lowest) to challenger (being the highest) and each tier is broken down into five divisions simply named, one, two, three, four, and five. League Of Legends has three game modes including two, five players against five players (5 V. 5) as well as three players against 3 players (3 V. 3). In each game mode a player is assigned a different rank, for example a player may be silver 3 in one game mode, but may be diamond 1 in another game mode. This is why the player ranking table sits between the game mode table and the player table.



Columns






		Name	Data type	Description / Attributes
		PLAYERID	CHAR(8 BYTE)	Unique identification number given to each player References: PLAYER
		GAMEMODEID	CHAR(4 BYTE)	Game mode that is being referenced References: GAMEMODE
		RANK	VARCHAR2(15 BYTE)	Rank given to the player in the game mode being referenced

1.9. Table: REFEREE (Referee Table)

```
CREATE TABLE referee (  
  refereeid      CHAR(8 BYTE) NOT NULL,  
  firstname     VARCHAR2(50 BYTE) NOT NULL,  
  lastname      VARCHAR2(50 BYTE) NOT NULL,  
  leaguejoindate DATE NOT NULL  
);  
ALTER TABLE referee ADD CONSTRAINT refereepk PRIMARY KEY ( refereeid );
```

The referee table describes the referee that would be taking part upholding the rules and regulations during the competition. Each referee is given a unique identification number that would be placed in the refereeID column. The table also consists of a firstName and lastName field which would be filled in with the associated referee's first name and last name. Lastly we have a date field that would be filled in with the date that the referee joined the league.

Columns





Name		Data type	Description / Attributes
	 REFEREEID	CHAR(8 BYTE)	Unique identification number given to each referee
	FIRSTNAME	VARCHAR2(50 BYTE)	The first name of the referee
	LASTNAME	VARCHAR2(50 BYTE)	The last name of the referee
	LEAGUEJOINDATE	DATE	The date that the referee joined the league

1.10. Table: REFEREES (Referees Table)

```
CREATE TABLE referees (  
  refereeid CHAR(8 BYTE) NOT NULL,  
  matchid CHAR(8 BYTE) NOT NULL  
);  
ALTER TABLE referees ADD CONSTRAINT refereespk PRIMARY KEY ( refereeid, matchid );  
  
ALTER TABLE referees  
  ADD CONSTRAINT refereesmatchfk FOREIGN KEY ( matchid )  
    REFERENCES match ( matchid );  
  
ALTER TABLE referees  
  ADD CONSTRAINT refereesrefereefk FOREIGN KEY ( refereeid )  
    REFERENCES referee ( refereeid );
```

In a typical video game competition there are multiple referees to uphold the rules of each match. The referees table allows us to associate many referees to a match and it also allows us to associate many matches to a referee. The table is made up of two fields one being the refereeID which would tell us which referee is present during the match and the matchID which would tell us what match the referee is a part of.

Columns




Name		Data type	Description / Attributes
	 REFEREEID	CHAR(8 BYTE)	Unique identification number given to each referee References: REFEREE
	 MATCHID	CHAR(8 BYTE)	Unique identification number given to each match References: MATCH

1.11. Table: SPONSOR (Sponsor Table)

```
CREATE TABLE sponsor (  
  sponsorid CHAR(8 BYTE) NOT NULL,  
  sponsorname VARCHAR2(100 BYTE) NOT NULL,  
  category VARCHAR2(10 BYTE) NOT NULL  
);  
ALTER TABLE sponsor ADD CONSTRAINT sponsorpk PRIMARY KEY ( sponsorid );
```

The sponsor table tells us about the sponsors that each team has as that is a big part of each teams budget during the competition. Each sponsor will have a sponsorID which is a unique identification number what will allow us to differentiate each sponsor. The sponsorName field will be filled in with the sponsors name. The category field will be filled in with the category that the sponsor falls under. This would include categories such as clothing, energy drinks, or pc and console accessories.

Columns





Name		Data type	Description / Attributes
	SPONSORID	CHAR(8 BYTE)	Unique identification number given to each sponsor
	SPONSORNAME	VARCHAR2(100 BYTE)	Name of the sponsor
	CATEGORY	VARCHAR2(10 BYTE)	Category that the sponsor (depends on the items that they are promoting through the team)

1.12. Table: SPONSORSHIPS (Sponsorships Table)

```
CREATE TABLE sponsorships (  
    sponsorid CHAR(8 BYTE) NOT NULL,  
    teamid CHAR(4 BYTE) NOT NULL,  
    startdate DATE NOT NULL,  
    enddate DATE NOT NULL  
);  
  
ALTER TABLE sponsorships ADD CONSTRAINT sponsorships_pk PRIMARY KEY ( sponsorid, teamid );  
  
ALTER TABLE sponsorships  
    ADD CONSTRAINT sponsorshipssponsorfk FOREIGN KEY ( sponsorid )  
        REFERENCES sponsor ( sponsorid );  
  
ALTER TABLE sponsorships  
    ADD CONSTRAINT sponsorshipsteamfk FOREIGN KEY ( teamid )  
        REFERENCES team ( teamid );
```

The sponsorships table sits between the team table and the sponsor table. It allows us to associate a sponsor and a team,, a sponsor may want to sponsor more than one team and a team may have more than sponsor. The sponsorID and the teamID are unique identification numbers for each sponsor and team. They are also primary keys as well as foreign keys. The last two fields are dates that would define when the sponsorship began as well as when the sponsorship would end.

Columns









Name		Data type	Description / Attributes
	SPONSORID	CHAR(8 BYTE)	Unique identification number given to each sponsor References: SPONSOR
	TEAMID	CHAR(4 BYTE)	Unique identification number given to each team References: TEAM
	STARTDATE	DATE	Date the sponsorship beings
	ENDDATE	DATE	Date that the sponsorship ends

1.13. Table: TEAM (Team Table)

```
CREATE TABLE team (  
  teamid          CHAR(4 BYTE) NOT NULL,  
  teamcaptain     CHAR(8 BYTE) NOT NULL,  
  teamname        VARCHAR2(50 BYTE) NOT NULL,  
  datejoinedleague DATE NOT NULL,  
  datedisbanded   DATE,  
  wins            NUMBER(2) NOT NULL,  
  draws           NUMBER(2) NOT NULL,  
  losses          NUMBER(2) NOT NULL  
);  
  
ALTER TABLE team ADD CONSTRAINT teamspk PRIMARY KEY ( teamid );  
  
ALTER TABLE team  
  ADD CONSTRAINT teamplayerfk FOREIGN KEY ( teamcaptain )  
    REFERENCES player ( playerid );
```

Each team in the competition has a unique “clan tag” or abbreviation of their team name, represented by teamID, which is also why it is the primary key for the team table. Each team must have team captain so that order is maintained during each match and throughout the competition. The teamCaptain column is a foreign key to the playerID column in the player table since a team captain is essentially just a player just more responsibilities. Each team has a name as well as the date that they joined the competition. In the team table we keep track of how many total games were wins, draws, or losses. One column that is not mandatory is the team table is the dateDisbanded column, only when a team has officially withdrawn from the competition do we fill in this column.

Columns





Name		Data type	Description / Attributes
	TEAMID	CHAR(4 BYTE)	Unique identification number given to each team
	TEAMCAPTAIN	CHAR(8 BYTE)	Identification number of the player that is the captain of the team References: PLAYER
	TEAMNAME	VARCHAR2(50 BYTE)	Name of the team
	DATEJOINEDLEAGUE	DATE	Date that the team joined the league
	DATEDISBANDED	DATE	Date the team is disbanded Nullable
	WINS	NUMBER(2, 0)	Number of wins in the competition
	DRAWS	NUMBER(2, 0)	Number of draws in the competition
	LOSSES	NUMBER(2, 0)	Number of losses in the competition

1.14. Table: TEAMROSTER (Team Roster Table)

```
CREATE TABLE teamroster (  
    teamid    CHAR(4 BYTE) NOT NULL,  
    playerid  CHAR(8 BYTE) NOT NULL,  
    startdate DATE NOT NULL,  
    enddate   DATE NOT NULL  
);  
  
ALTER TABLE teamroster ADD CONSTRAINT teamrosterpk PRIMARY KEY ( teamid, playerid );  
  
ALTER TABLE teamroster  
    ADD CONSTRAINT teamrosterplayerfk FOREIGN KEY ( playerid )  
        REFERENCES player ( playerid );  
  
ALTER TABLE teamroster  
    ADD CONSTRAINT teamrosterteamsfk FOREIGN KEY ( teamid )  
        REFERENCES team ( teamid );
```

Each team in the competition is made up of many players, which is why we have the team roster table sitting in between the team table and the player table. In this table we have teamID which is a foreign key to the team table but it is also being used as a primary key in the team roster table. We also have playerID which is a foreign key to the player table but it is also being used as a primary key in the team roster table. In this table we also have dates of when the roster took effect and when the roster became obsolete.

Columns

Name		Data type	Description / Attributes
	TEAMID	CHAR(4 BYTE)	Unique identification number from each team References: TEAM
	PLAYERID	CHAR(8 BYTE)	Unique identification number given to each player References: PLAYER
	STARTDATE	DATE	Date that the roster takes effect
	ENDDATE	DATE	Date that the roster is no longer in effect

2. Sample Queries And Output

```
SELECT D.DEVELOPERID,D.DEVELOPERNAME AS DEVELOPER,G.GAMEID,G.GAMENAME AS GAME,
M.GAMEMODEID,M.GAMEMODENAME AS GAMEMODE,M.GAMEMODEDESCRIPTION AS DESCRIPTION
FROM DEVELOPER D
INNER JOIN GAME G
ON G.DEVELOPERID = D.DEVELOPERID
INNER JOIN GAMEMODE M
ON M.GAMEID = G.GAMEID
WHERE G.GAMEID !='NULL';
```

DEVELOPERID	DEVELOPER	GAMEID	GAME	GAMEMODEID	GAMEMODE	DESCRIPTION
98661941	Riot Games	68679929	League Of Legends	3870	Summoners Rift	The gold standard for competitive League of Legends

```
SELECT B.BRACKETID AS BRACKET, B.TEAMID AS CLANTAG,T.TEAMNAME AS TEAM, T.WINS
FROM BRACKETPLACEMENT B
INNER JOIN TEAM T
ON T.TEAMID = B.TEAMID
WHERE B.BRACKETID = 25705094 ORDER BY WINS DESC;
```

BRACKET	CLANTAG	TEAM	WINS
25705094	RNG	Royale Never Giveup	68
25705094	C9	Cloud 9	67
25705094	IG	Invictus Gaming	67
25705094	EFOX	Echo Fox	46

```
SELECT T.TEAMNAME AS TEAM, T.WINS, T.DRAWS, T.LOSSES
FROM TEAM T
INNER JOIN BRACKETPLACEMENT P
ON T.TEAMID = P.TEAMID
INNER JOIN BRACKET B
ON P.BRACKETID = B.BRACKETID
WHERE B.BRACKETID = 25703094;
```

TEAM	WINS	DRAWS	LOSSES
Counter Logic Gaming	54	11	8
Golden Gamers	56	20	5
Optic Gaming	63	10	24
Team Liquid	34	9	21

```

SELECT P.PLAYERID AS ID, P.FIRSTNAME || ' ' || P.LASTNAME AS FULLNAME, P.TEAMID AS TEAM
FROM PLAYER P
INNER JOIN PLAYERRANKING R
ON P.PLAYERID = R.PLAYERID
WHERE R.RANK = 'Challenger' ORDER BY TEAM DESC;

```

ID	FULLNAME	TEAM
10459875	Mariam Checchetelli	TLIQ
82903363	Ronnie Ivel	TLIQ
99052761	Natasha Swiffin	TLIQ
96725261	Chucho Sellek	RNG
51949324	Annemarie Braganca	RNG
71161686	Lenka Broadberry	OPTG
97634970	Romona Olivie	GG
92703785	Jeanne Rontsch	EFOX
72543581	Wash Fullick	CLG
11070575	Danica Rimer	CLG
61366023	Eddi Marieton	C9
65255345	Elfrida Batrim	C9

```

SELECT P.PLAYERID, P.FIRSTNAME || ' ' || P.LASTNAME AS FULLNAME, P.TEAMID
FROM PLAYER P
INNER JOIN MATCH M
ON P.TEAMID = M.TEAMID1
WHERE M.GAMEDATE like '%16-NOV-18%' AND M.TEAMID1 = 'EFOX';

```

PLAYERID	FULLNAME	TEAMID
44387003	Ermin MacGille	EFOX
49649938	Gwendolen Mascall	EFOX
50858032	Kerrill Duley	EFOX
92703785	Jeanne Rontsch	EFOX
19698459	Hobie Deeley	EFOX


```
SELECT * FROM MATCH WHERE TEAMID1 = 'OPTG' OR TEAMID2 = 'OPTG';
```

⚡ MATCHID	⚡ GAMEMODEID	⚡ TEAMID1	⚡ TEAMID2	⚡ GAMEDATE
55234044	3870	OPTG	TLIQ	15-NOV-18
55334044	3870	OPTG	TLIQ	16-NOV-18
55434044	3870	OPTG	TLIQ	17-NOV-18
17623058	3870	CLG	OPTG	18-NOV-18
17523058	3870	CLG	OPTG	19-NOV-18
17423058	3870	CLG	OPTG	20-NOV-18

```
select P.PLAYERID AS ID, P.FIRSTNAME || ' ' || P.LASTNAME AS FULLNAME, P.TEAMID AS TEAM
from PLAYER P
inner join TEAMROSTER R
on P.PLAYERID = R.PLAYERID
WHERE R.TEAMID = 'C9';
```

⚡ ID	⚡ FULLNAME	⚡ TEAM
74137655	Wyatan Dearness	C9
61366023	Eddi Marieton	C9
65255345	Elfrida Batrim	C9
33425449	Sheree Piotrowski	C9
10185954	Stanley Vamplus	C9