CS2300 Project Phase 2

Andrew Henningsen & Evan Wilcox

November 7, 2018

1 Revised Problem Statement

The Overwatch League is a professional esports league for the game of Overwatch developed by and for Blizzard Entertainment. Twelve different teams compete in the Pacific and Atlantic Divisions to make it into the playoffs to win a Grand Prize of $1,000,000. Keeping track of all the different players and teams is a difficult task, especially when players are traded or dropped or teams get eliminated from competing. Fans often desire a way to compare different players or in the case that they miss a match, they can view a recap on the go.

2 Revised conceptual database design

Our database consists of storing information of various different entities:

The "Person" Entity consists of a unique identifying id, a name, and a handle.

There are three different types of "Person", One of which is a "Player". This entity contains a role, a number, a location that they are from, and a link to a picture. In addition, a Player can play on many different teams, either the standard twelve teams, or teams for a specific country.

The second type is "Personnel", casters/personnel play an integral part in the Overwatch experience and thus we must store data on them. Personnel include a type(Host, Caster, or Analyst) and a link to a picture.

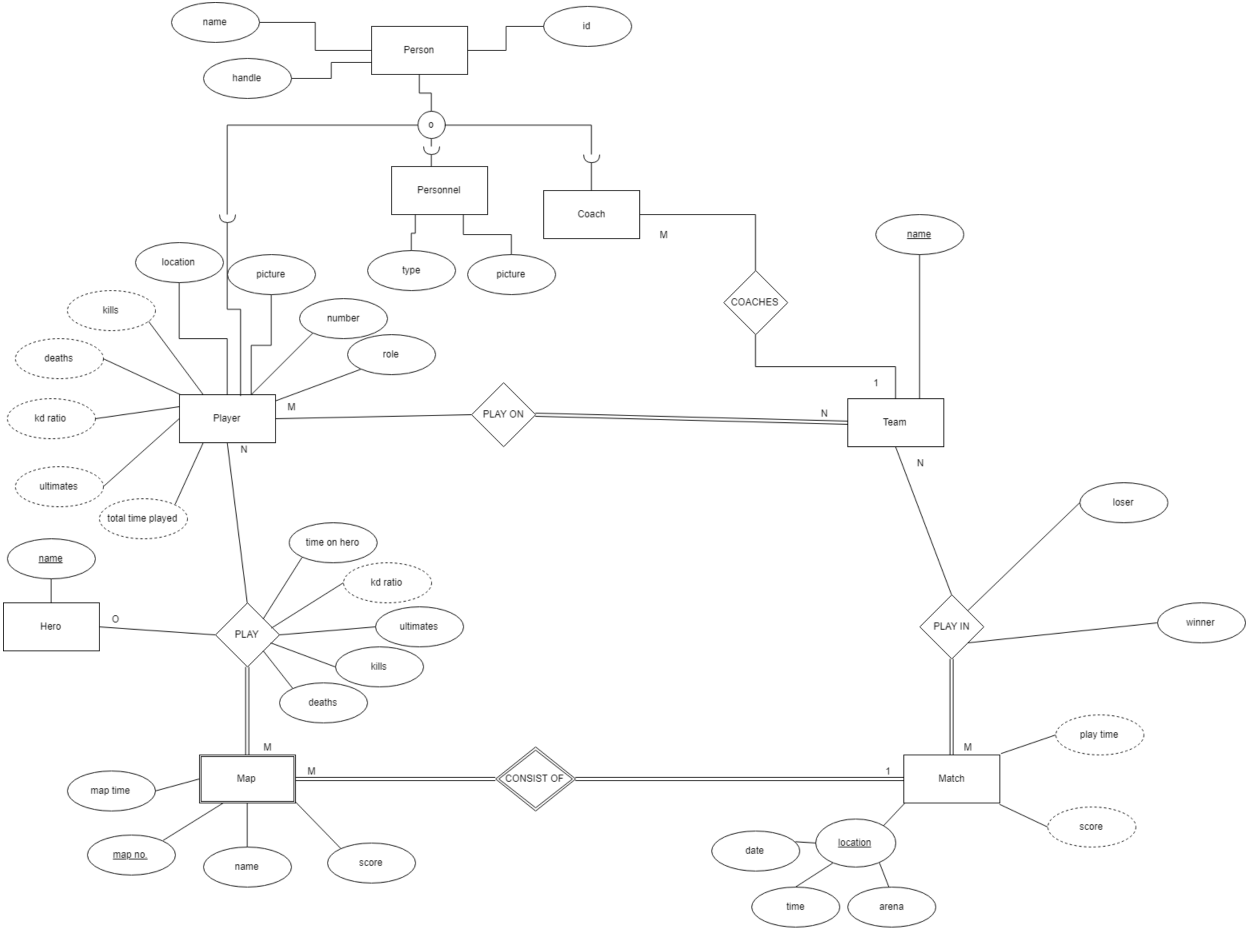
The final type of person is a "Coach". Coaches coach a "Team".

The "Team" entity includes a unique id, a division, a name and a link to a picture.

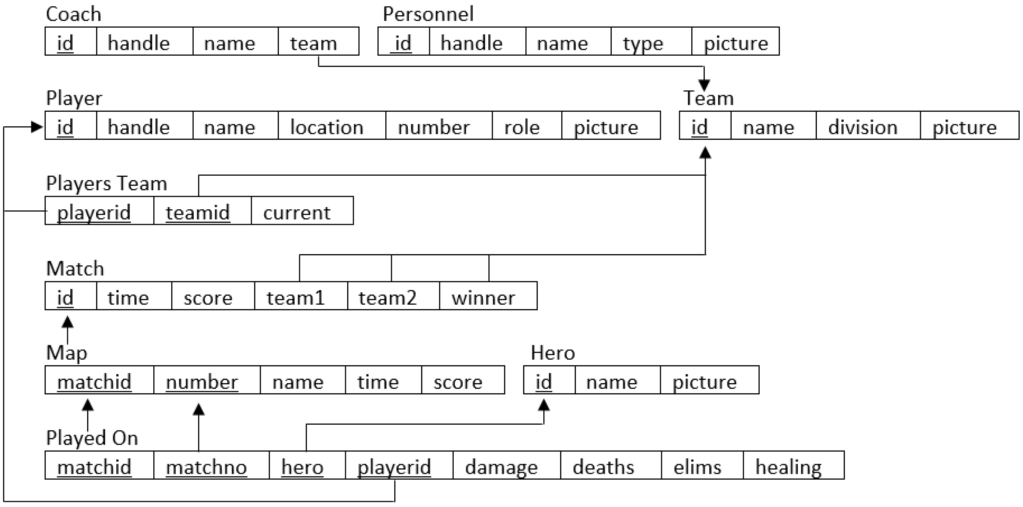
A "Match" is played between two teams, a team1 and a team2. In addition, a match has a uniquely identifying id, a final score, a time and a winner.

A "Map" is what is considered a part of a Match. There are 4 maps with an additional tie-breaker map if necessary. A "Map" cannot exist without a Match and thus is considered a weak entity. A map contains the name of the in game world being played on, a time, a score, a type, and a link to a picture.

There are 28 (currently) "Hero"s each with a uniquely identifying id, a name, and a picture.
A Player players on a map with specific hero. They can, however switch heroes and thus it is important to know that this ternary relationship depends on the hero, player and map. In each of these relationships, we require to know how many eliminations, deaths, damage, and healing was done.



3 Logical database design


Coach
id handle name team
Personnel
id handle name type picture
Player
id handle name location number role picture
Team
id name division picture
Players Team
playerid teamid current
Match
id time score team1 team2 winner
Map
matchid number name time score
Hero
id name picture
Played On
matchid matchno hero playerid damage deaths elims healing

Foreign Keys
Map.matchid references Match.id
PlayersTeam.playerid references Player.id
PlayersTeam.teamid references Team.id
PlayedOn.matchid, PlayedOn.matchno references Map.matchid, Map.number
PlayedOn.hero references Hero.id
PlayedOn.playerid references Player.id

Table	**Attribute**	**Datatype**	**Constraint**	**Description**
Team	id	int	Primary Key	Unique Team ID.
Team	name	varchar	Not Null	Team's name.
Team	division	varchar		Division the team is in.
Team	picture	varchar		Link to Team logo.
Player	id	int	Primary Key	Unique Player ID.
Player	handle	varchar	Not Null	Player's in-game name.
Player	name	varchar	Not Null	Player's actual name.
Player	location	varchar		Where player is from.
Player	no	int		Player's team number.
Player	role	varchar	Not Null	Favored role of player.
Player	picture	varchar	Not Null	Link to picture of player.
Players Team	playerid	int	Foreign Key	Foreign Key to Player.
Players Team	matchid	int	Foreign Key	Foreign Key to Team.
Players Team	current	int	Not Null	Indicates player's current team.
Hero	id	int	Primary Key	Unique Hero ID.
Hero	name	varchar	Not Null	Hero's name.
Hero	picture	varchar	Not Null	Link to picture of Hero.
Match	id	int	Primary Key	Unique Match id.
Match	time	int	Not Null	When the match took place.
Match	score	varchar	Not Null	Score of the match.
Match	team1	int	Foreign Key	Foreign Key to Team.
Match	team2	int	Foreign Key	Foreign Key to Team.
Match	winner	int	Foreign Key	Foreign Key to Team.
Map	matchid	int	Foreign Key	Foreign Key to Match.
Map	number	int	Primary Key	Unique map number.
Map	name	varchar	Not Null	Where the map was played
Map	time	int	Not Null	Length of map.
Map	score	varchar	Not Null	Score of map.
Played On	matchid	int	Foreign Key	Foreign Key to Match.
Played On	matchno	int	Foreign Key	Foreign Key to Map.
Played On	playerid	int	Foreign Key	Foreign Key to Player.
Played On	hero	int	Primary Key	Hero that Player played.
Played On	damage	int		Damage done by Player.
Played On	deaths	int		Amount of times Player died.
Played On	elims	int		Amount of elims Player got.
Played On	healing	int		Healing done by Player.
Coach	id	int	Primary Key	Unique Coach ID.
Coach	handle	varchar		In-game handle.
Coach	name	varchar	Not Null	Coach's name.
Coach	team	int		Team Coach coaches.
Personnel	id	varchar	Primary Key	Unique Personnel ID.
Personnel	handle	varchar		In-game handle.
Personnel	name	varchar	Not Null	Personnel's name.
Personnel	type	varchar	Not Null	Caster, Host, or Analyst.
Personnel	picture	varchar	Not Null	Link to picture.

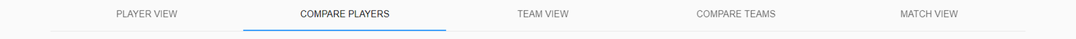
Using SQLite datatypes.

4 Application program design

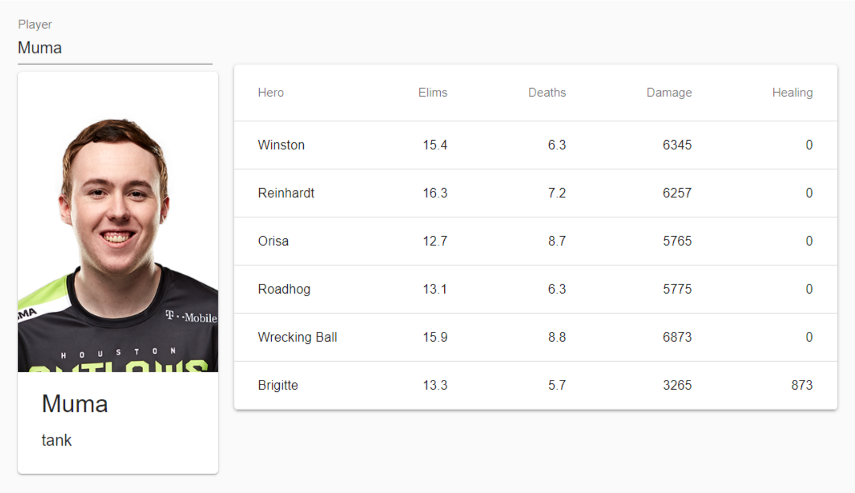
```
Node.JS backend server:
The following functions will be hosted as webpages return a json formatted object
    compare_players(player1, player2)
        stats_between_players = database_query
        return stats_between_players

    compare_teams(team1, team2)
        stats_between_teams = database_query
        return stats_between_teams

    player_stats(player)
        player_stats = database_query
        return player_stats

    team_stats(team)
        team_stats = database_query
        return team_stats

    match_stats(match)
        match_stats = database_query
        return match_stats


React Front End:
    Structure:
        Player View
            queries database and displays data beautifully

        Compare Player View
            queries database and displays data beautifully

        Team View
            queries database and displays data beautifully

        Compare Team View
            queries database and displays data beautifully

        Match View
            queries database and displays data beautifully
```

5 User interface design

Header:



Player View:

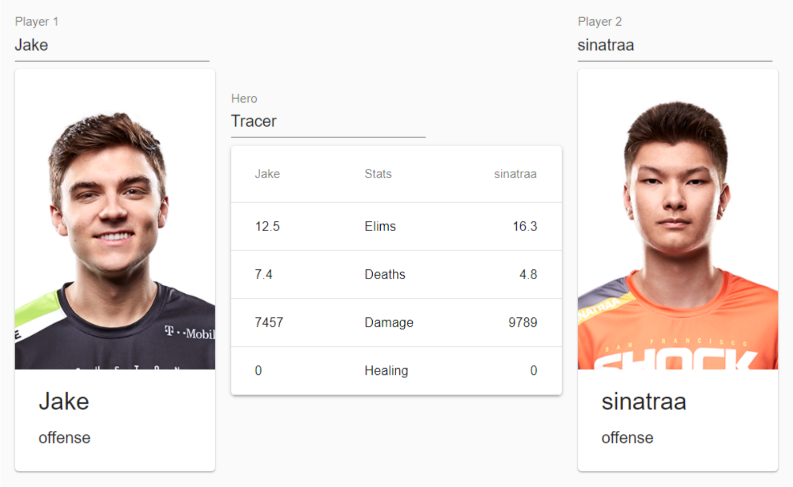


Player
Muma

Muma
tank

Hero	Elims	Deaths	Damage	Healing
Winston	15.4	6.3	6345	0
Reinhardt	16.3	7.2	6257	0
Orisa	12.7	8.7	5765	0
Roadhog	13.1	6.3	5775	0
Wrecking Ball	15.9	8.8	6873	0
Brigitte	13.3	5.7	3265	873

Compare Players:



Player 1
Jake

Player 2
sinatraa

Hero
Tracer

Jake	Stats	sinatraa
12.5	Elims	16.3
7.4	Deaths	4.8
7457	Damage	9789
0	Healing	0

Jake
offense

sinatraa
offense

With similar views for the teams.