Group 5

Royden Lynch

Gerald Mouton

Seth Richard

Nick Settoon

Business Description

Project Objectives

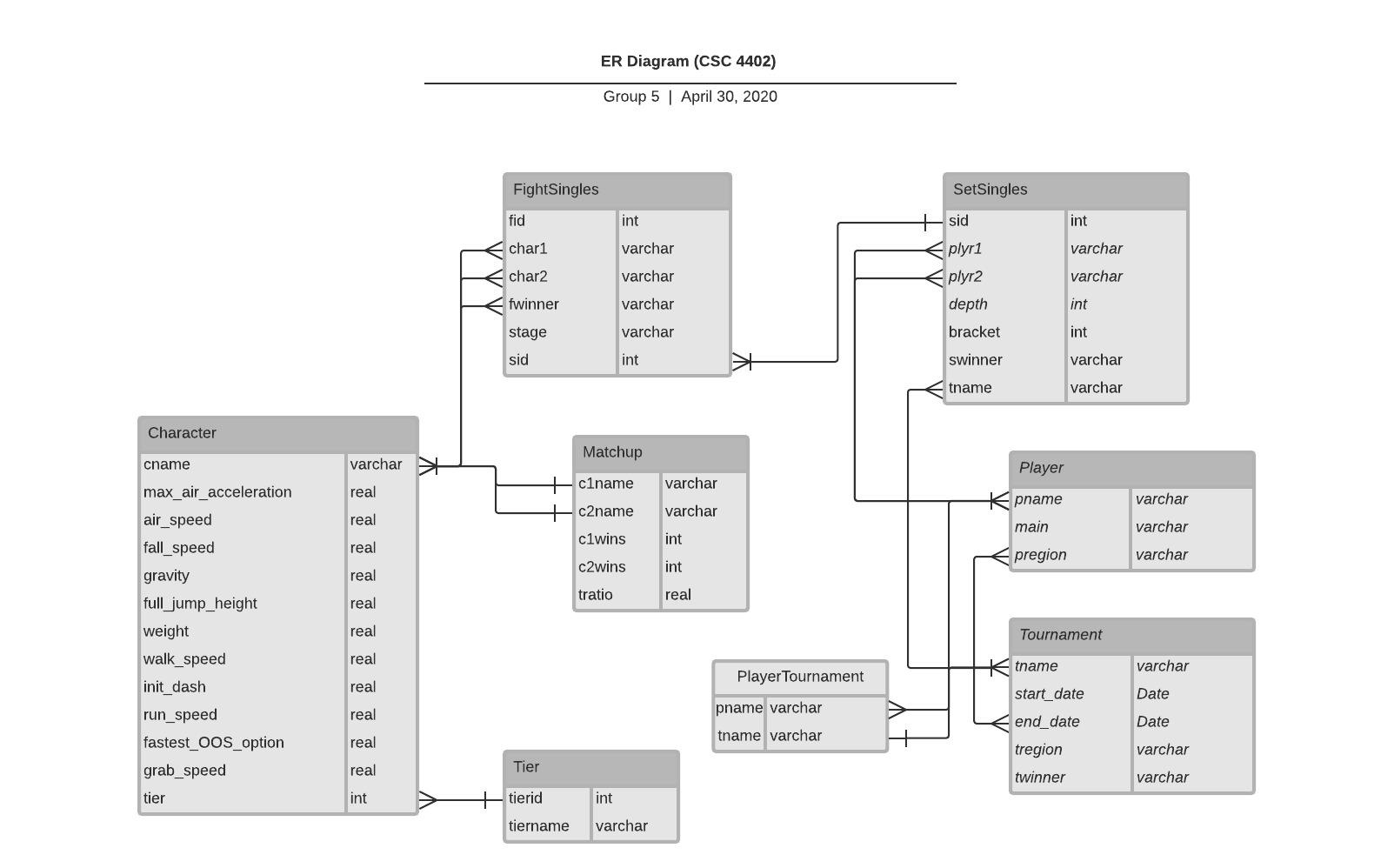
Esports is one of the most rapidly growing sports in the world right now. This form of entertainment and competition expands across the globe, and the world’s best come together to compete in epic battles and showcase their skills. One of the more popular games is Super Smash Bros. Ultimate. The game lets players choose any one of the 75 characters. These characters come from Nintendo’s most popular games including the Mario Bros. Series, Pokemon, and Legend of Zelda Series. Each esports player chooses their favorite character to battle one another in a fight for survival. The esport tournaments consist of players battling in a double elimination style bracket with a best-of-five or best-of-three series deciding who to crown the best Super Smash Bros. Ultimate player. There is much data to keep track of with these tournaments, such as character attributes, fight and set outcomes, player statistics, and more. Our group plans to tackle and sort a large proportion of this data and store it in a database to make for an easy and organized view of results. In all, our project objective is to create an efficient and consistent database to answer any user request regarding all of the Super Smash Bros. Ultimate data we collect.

Business Requirements

The database must be formatted in a way that can satisfy these requirements by the user.

* We need the database to correctly determine who has a better head to head matchup between two esports players.
* The database must be able to account for the increase in growth of entries as the popularity of the sport continues to grow.
* Be able to rank the players by the number of set wins they have in order to create a “standings” of the players.
* Determine the fictional characters that give a player the biggest advantage and highest chance to win a set.
* Be able to determine if where a player is from has any influence on their performance.
* Determine which players had the highest participation, meaning ther participated in the most tournaments.
* What are the most popular regions for Ultimate, and what areas need more promotion and advertisement.
* Find which characters are considered to be the favorite amongst players and are used the most to fight in sets.

ER Model



Constraints

* FightSingles:
  + Each fighter and winner in a fight must be a Character entry.
  + Each fight must be a valid Matchup entry.
  + Each fight must have a parent set.
* SetSingles:
  + Both players and the winner in a set must be valid Player entries.
  + The set must be from a Tournament entry.
* Tournament:
  + Every tournament must have a winner from the Player table.
* Player:
  + A player’s main character should be a valid Character entry
* PlayerTournament:
  + No player should be paired with a tournament more than once
  + Every player name must be a valid Player entry
  + Every tournament name must be a valid Tournament entry
* Character:
  + Each character is in a valid Tier
* Matchup:
  + Each Matchup is a valid cross product of the Character table with itself
  + All matchups are unique from the perspective of the c1name field. I.e. (c1name, c2name) may be the same Matchup as (c2name, c1name). See Relational Data Model below for reasoning.

Assumptions

* Each tournament should have one, and only one, winner.
* Player names are unique
* Character names are unique
* Tournament names are unique
* There are a finite number of possible matchups

Relational Data Model

For our database design we have divided our data into eight different tables. Our main goal when designing our relational database was to achieve a certain level of data integrity and attempt to avoid data redundancy. We took measures to avoid redundant information by creating multiple smaller tables to make our database more efficient. As a result, each of our tables achieves BCNF. Our database includes a table for characters, fight singles, character matchups, players, players in each tournament, player set singles, character tier, and lastly the names of the tournaments from which we have collected data.

The Character table holds unique information about each character including their cname(character name), max\_air\_acceleration, air\_speed, fall\_speed, gravity, full\_jump\_height, weight, walk\_speed, init\_dash, run\_speed, fastest\_OOS\_option, grab\_speed, and tierid. In this table the primary key is the character name because each character has their own unique name and the foreign key is the tierid which references tierid in the Tier table which is used to determine the skill of a character compared to the rest of them. The table is in BCNF because cname is the only primary key or candidate key.

The next table is FightSingles which keeps a record of fights between two characters at a certain stage as well as the fight winner. The attributes of this table are fid(fight id), char1(character 1), char2(character 2), fwinner(fight winner), stage, and sid(set id). The primary key in this table is fid and each fight between two characters is given a specific id number. The foreign key in this table is sid because it references sid in SetSingles which maps the characters used to a player in a specific set. No combination of the other attributes determines anything in the table. Therefore, fid is the only candidate key in the table, because of this it is in BCNF.

The Matchup table evaluates every possible character versus character matchup and determines which character has the advantage over another. The attributes of the Matchup table are c1name, c2name, c1wins, c2wins, and tratio. The primary key of this table is c1name, c2name and the foreign keys are c1name and c2name which both reference cname in the Character table. We wanted this table stored in the database in order to make future queries simpler. Doing a cross-product on character and combining with fight counts every time we wanted to use that dataset wasn’t a good idea. The table is in BCNF because (c1name, c2name) is the only candidate key. In the Matchup table, no combination of (c1wins, c2wins, and tratio) is a determinate. C1name or c2name alone are not determinants, so the table is in 2NF. Because of the previous two facts, the table ends up in BCNF.

The Player table describes each esports player that has participated in the tournaments we have collected data from. The table attributes are pname(player name), main(their most used character), and pregion(state or country a player is from). The primary key of Player is pname and the foreign key is main which references cname in Character. The table is in BCNF because pname determines both of the other attributes.

The PlayerTournament table lists every player that played in a specific tournament. The attributes of PlayerTournament are pname(player name) and tname(tournament name). The primary key is cname, tname and the foreign keys is pname as it references pname in Player. The table is in BCNF as there are no non-primary attributes.

The SetSingles table lists every fight between two players, the winner, and the tournament they were competing in. The attributes of this table are sid(set id), plyr1, plyr2, depth, swinner(set winner), bracket, and tname(tournament name). The primary key of the table is sid because each set is given a unique number and the foreign key is tname which references one of the tournaments(tname) in Tournaments. The table is in BCNF normal form. The candidate keys are (p1,p2,depth,tname), (swinner,depth,tname), (sid) no proper subset of those keys determines (bracket), the only non-primary-attribute. This makes the table 2NF. In addition, (bracket) doesn’t determine anything in the table, so the table is in 3NF. It’s also BCNF because all the dependencies in the table have a super key as their determinate.

The Tier table defines the scale used to grade characters on their overall skill compared to the other characters. The attributes of Tier are tierid and tiername. Tierid is the primary key of Tier and there is no foreign key in this table. The table is in BCNF as the table’s purpose is to act as a grading scale to work with the Character table to rank a character.

Lastly, the Tournament table describes the location, dates, and winner of a specific tournament. The attributes include tname(tournament name), start\_date, end\_date, tregion, and twinner. The primary key of Tournaments is tname since each tournament has a unique number and the foreign key is twinner which references a pname(player name) in Player. The table is in BCNF as there are no partial and NPA->NPA dependencies. All determinants are either (tname), (start\_date), (end\_date). These are superkeys and thus Tournament is in BCNF.

Data Manipulation - Query Table

|  |  |  |  |
| --- | --- | --- | --- |
| # | Query | SQL Code | Resulting Records |
| 1. | This query ranks players by the number of tournaments, in our data set, that they went to. | SELECT pname as Player,  COUNT(\*) as Tourneys  FROM PlayerTournament  NATURAL JOIN Player  GROUP BY pname  ORDER BY Tourneys  DESC; |  |
| 2. | This query ranks matchups, a fight between two characters independant of players, by the number of occurrences of that matchup. | SELECT \*, c1wins + c2wins as Total\_Fights  FROM Matchup  WHERE Total\_Fights >= 1  ORDER BY Total\_Fights DESC; |  |
| 3. | This query lists players by the number of fights they have won. | SELECT swinner AS name,  COUNT(fid) AS  fight\_wins  FROM FightSingles  NATURAL INNER JOIN  SetSingles  GROUP BY swinner  ORDER BY fight\_wins  DESC; |  |
| 4. | This query ranks the regions in our data set by the number of players that are based in that region. | SELECT pregion AS region, COUNT(pname) AS num\_players  FROM Player  GROUP BY pregion  HAVING num\_players >= 3  ORDER BY num\_players DESC; |  |
| 5. | This query ranks regions by the number of tournaments that were held there. | SELECT tregion AS region, COUNT(tname) AS Num\_Tournaments  FROM Tournament  GROUP BY tregion  ORDER BY Num\_Tournaments DESC; |  |
| 6. | This query ranks which characters are most popularly used in Super Smash Bros. Ultimate. | SELECT cname AS Character, tiername as Tier, Total\_Played  FROM (SELECT Character AS cname, sum(Play\_rate) as Total\_Played  FROM (SELECT char1 as Character, COUNT() as Play\_rate  FROM FightSingles NATURAL JOIN SetSingles NATURAL JOIN Tournament  GROUP BY char1  UNION  SELECT char2 as Character, COUNT() as Play\_rate  FROM FightSingles NATURAL JOIN SetSingles NATURAL JOIN Tournament  GROUP BY char2)  GROUP BY Character  ORDER BY Total\_Played DESC) NATURAL JOIN Character NATURAL JOIN Tier; |  |
| 7. | This query ranks players by the number of sets they have won in all of the tournaments in our data set. | SELECT Plyr, IFNULL(Set\_wins,0) as  Set\_wins  FROM (SELECT Plyr  FROM (SELECT plyr1 as Plyr  FROM SetSingles  GROUP BY plyr1  UNION  SELECT plyr2 as Plyr  FROM SetSingles  GROUP BY plyr2))  LEFT JOIN  (SELECT swinner, COUNT(\*) as  Set\_wins  FROM SetSingles  GROUP BY swinner) on Plyr  = swinner  ORDER BY Set\_wins DESC, Plyr ASC; |  |
| 8. | Tournaments are structured like trees only horizontally. With this in mind, we assigned different depths to sets at different rounds of a tournament. Meaning that as you go from Grand Finals down the depth increases.  This query ranks the players by how early they were eliminated from the tournament. With 1 meaning they made it to the Grand Finals. | SELECT plyr1 as Player, MIN(depth) as  Round\_Eliminated  FROM SetSingles NATURAL JOIN  Tournament  GROUP BY Player  UNION  Select plyr2 as Player, MIN(depth) as  Round\_Eliminated  FROM SetSingles NATURAL JOIN  Tournament  GROUP BY Player  ORDER BY Round\_Eliminated; |  |
| 9. | In Super Smash Bros, some characters are better than others. Therefore, not all fights take the same level of skill.  This query ranks each player by the number of fights that they have won when the character that they are playing has a lower tier than the opponent's character.  This essentially means how many fights has a player won when their character was at a disadvantage. | SELECT Player,  SUM(Wins\_At\_Disadvantage) as  Wins\_At\_Disadvantage  FROM (SELECT plyr1 as Player,  COUNT() as Wins\_At\_Disadvantage  FROM(SELECT char1, char2,  fwinner, plyr1, plyr2  FROM FightSingles NATURAL  JOIN SetSingles)  WHERE (SELECT tratio  FROM Matchup  WHERE c1name = char1 AND  c2name = char2) > 1  AND char1 = fwinner  GROUP BY plyr1  UNION  SELECT plyr2 as Player, COUNT() as  Wins\_At\_Disadvantage  FROM(SELECT char1, char2,  fwinner, plyr1, plyr2  FROM FightSingles  NATURAL JOIN SetSingles)  WHERE (SELECT tratio  FROM Matchup  WHERE c1name = char2 AND  c2name = char1) > 1  AND char2 = fwinner  GROUP BY plyr2)  GROUP BY Player  ORDER BY Wins\_At\_Disadvantage DESC; |  |
| 10. | This query answers the question “Which character performs the best against others on average?”  The lower the number the better they should perform. | SELECT c1name, adv\_tier as c1\_tier, c2name, t2.tiername as c2\_tier, tratio  FROM (SELECT c1name, c2name, tratio, t1.tiername as adv\_tier  FROM Matchup as mu INNER JOIN Character as char on mu.c1name = char.cname NATURAL JOIN Tier as t1) as view1 INNER JOIN Character as char2 on view1.c2name = char2.cname NATURAL JOIN Tier as t2  WHERE tratio < 1  ORDER BY tratio; |  |