Stage 2: Conceptual Database Design

Explanation

The following entities: **Events, Matches, and Game** were all established as individual entities instead of attributes of **Events** because the entities have different statistics or row elements associated with them. For example, the **Events** entity will have prize money associated with it while the **Matches** entity will have statistics for the entire match (compared to statistics for a single **Game** in that **Matches**).

The **Events** entity represents the regional event encompassing the Rocket League Championship series competition. The attributes we have for this entity consist of “event” (event name), “event_split” (event split / part of the season), “event_region” (event region), event_slug (octane.gg event URL), and the “event_tier” (event tier / level) contributing to the logistics of the event, and “event_id” (event identifier within the database). The **Events** entity uses a composite primary key consisting of “event_id, event_region, event_tier” to uniquely identify each entry. This composite key is chosen because the same event may be held in different regions simultaneously, in which the same event tier may be occurring. Regarding the cardinality for these events, we have a 1-many relationship for **Events-Game** (played_during) and **Events-Matches** (scheduled_in). These cardinalities came because events could be canceled, in which there are no matches or games for an event, or there could be an arbitrary number of games and matches depending on the decided upon bracket sizes for the events. Below is the event split by season taken from our kaggle dataset:

Fall Split

- Regionals 1, 2, 3 (All regions - Invitational Qualifiers, Closed Qualifiers and Main Events)
- Fall Major - Asia-Pacific Qualifier
- Fall Major - North America Tiebreaker (Complexity Gaming vs. Spacestation Gaming)
- Fall Major - Main Events

Winter Split

- Regionals 1, 2, 3 (All regions - Closed Qualifiers and Main Events)
- Winter Major - Asia-Pacific Qualifier
- Winter Major - Main Events

Spring Split

- Regional 1, 2, 3 (All regions - Closed Qualifiers and Main Events)
- Spring Major - Asia-Pacific Qualifier
- Spring Major - Main Events

World Championship

- Wildcard Stage (Play-In)
- Group Stage
- Playoffs

The **Matches** entity is the set of 5/7 games played between teams, with special treatment for the fall split finals played in 2 winning sets of best of 7 (Rocket League Championship Series 2021-2022). The entity has the “score” (score of best out of 5/7), “winner” (which team won) and “color” (winning team color) attributes defining the state of the match set. The primary key “match_id” allows us to uniquely identify the different match sets that were played within a certain region, along with “player_tag” which is used to identify the specific entries in the **Matches** table via player usernames. In addition, this entity has a foreign key to “team_name” which is used to bolster the legibility of the data entries within the table. In this manner, it is easy to see which teams won which matches along with the associated players. This entity has cardinalities of 1-many relationship for **Matches-Game**, many-1 relationship between **Matches-Events**, many-many between **Matches-Player**, and many-many between **Matches-Team**. The many-many relationships will have their own respective tables in accordance with the UML diagram—that is, “takes_part_in” and “competes_during” will exist and have information related to statistics. Regarding cardinality, there are 3 to 7 games possible in a match; 3/5 games if a single team wins all the games in a best of 5/7 and 5/7 games if both teams have equal amount of wins at the end of the 4th/6th match. Additionally, **Matches** belong to only one event. I.e. there can be many **Matches** for a singular event. Lastly, many teams participate in many matches and, consequently, so do many players.

The **Game** entity is, within the match set, normally a 3 versus 3, 5 minute long game between 2 different teams with a possibility of going to overtime if the score were to be tied when the clock strikes zero. The entity has the attributes “winner” (denoting the winner of the game) and “platform” (denoting the primary platform the game was played on) The primary key “game_id”, along with the foreign keys “player_tag” linking the game_id to a player and “team_name” linking the game_id to a team, makes each game uniquely distinguishable. Player and team performance is linked and recorded on a match-by-match basis, necessitating the creation of this entity and the use of foreign keys linking the **Player** and **Team** entities to **Game**. Next, regarding the cardinalities for this entity, the linking with the **Player** and **Team** entities is done in a manner that there is a many-many relationship between **Game-Player** and a many-many relationship between **Game-Team**. These relationships will have their own tables in accordance with the UML diagram—that is, “participates_in” and “competes_in” will exist and have information related to statistics. The reasoning for these cardinalities is because there are 2 teams of 3 players participating in each game of Rocket League. Next, are the relationships between **Game-Matches** as many-1 and **Game-Events** as many-1. These cardinalities are due to how there can be many games belonging to the same match, and, similarly, many games belonging to the same event.

The **Team** entity defines the teams that played a particular set of **Games**, a single **Team** has played multiple **Matches** throughout **Events**. The entity has the attributes “team_region” (denoting the region the team is based in), “team_slug” (the team’s unique URL), and “team_name” (the name team uses in game). The primary key “team_id” makes each team uniquely distinguishable. The cardinalities for this entity are such that there is a 1-many relationship between **Team-Player** and a many-many relationship between **Team-Game**. The reasoning for the cardinalities are that there are 3 participating players for each team in a game of Rocket League and two participating teams for each game of Rocket League.

The **Player** entity is an individual participating in the RLCS tournament. The **Player** will belong to a particular **Team** and play matches for that team in **Events**. The entity has the attributes “player_tag” (denoting the player_tag used in game), “player_name” (the name of the player as they are most commonly known (not always used)), “player_country” (the country the player represents), and “player_slug” (the player’s unique URL). The primary key “player_id” makes each player uniquely distinguishable. Lastly, regarding the cardinalities of this entity, the linking with the **Player** and **Game** entities is done in a many-many relationship between **Player-Game** and a many-1 relationship between **Player-Team**. The many-many relationships will have their own respective tables in accordance with the UML diagram—that is, “participates_in” and “takes_part_in” will exist and have information related to statistics. The reasoning for these cardinalities is that any player can play an arbitrary number of games and a player can only belong to a single team.

Normalize

Rocket League Normalization:

All of the Rocket League entities already have relations that are in accordance with 3NF normalization:

Player relations are already in 3NF

- player_id → player_name
- player_id → player_slug
- player_id → player_tag
- player_id → player_country
- player_slug → player_id
- player_tag → player_id
- candidate_key: player_id

Team relations are already in 3NF

- team_id → team_region
- team_id → team_slug
- team_id → team_name
- team_slug → team_id

- team_name → team_id
- candidate_key: team_id

Game relations are already in 3NF

- game_id, player_tag → team_name
- game_id → winner
- game_id → platform
- candidate_key: game_id, player_tag

Matches relations are already in 3NF

- match_id, player_tag → color
- match_id, player_tag → team_name
- match_id, player_tag → winner
- match_id, player_tag → score
- match_id → platform
- candidate_key: match_id, player_tag

Events relations are already in 3NF

- event_id → event_split
- event_id → event_region
- event_id → event
- event_slug → event_id
- candidate_key: event_id, event_region, event_tier

Convert to Relational Schema

Rocket League Relational Schema:

Player(player_id: INT [PK], player_name:CHAR(255), player_slug:CHAR(255), player_tag:CHAR(255), player_country:CHAR(2))

Team(team_id: INT [PK], team_region:CHAR(10), team_slug:CHAR(255), team_name: CHAR(255))

Game(game_id: INT [PK], player_tag: CHAR(255) [PK], winner: BOOL, platform: CHAR(10), team_name: CHAR(255) [FK])

Matches(match_id: INT [PK], player_tag: CHAR(255) [PK], score: INT, color: CHAR(10), winner: BOOL, platform: char(10), team_name: CHAR(255) [FK])

Events(event_id: INT [PK], event_tier: CHAR(25) [PK], event_region: CHAR(25) [PK], event_split: CHAR(10), event: CHAR(25), event_slug: CHAR(255))