https://jlvr-cs340-project.herokuapp.com/

James Lipe
Vincent Rastello

Fun Sports Leagues – Final

Feedback by peer reviewer from step 3

- *Does the UI utilize a SELECT for every table in the schema?* In other words, data from each table in the schema should be displayed on the UI. Note: it is generally not acceptable for just a single query to join all tables and displays them.

Nearly. It appears there is no way to display information from a team's roster. Everything else appears to be in place. The way the information from tables is displayed is easy to read and well organized.

- *Does at least one SELECT utilize a search/filter with a dynamically populated list of properties?*

Yes. There is a way to view current teams and leagues on the corresponding pages.

- *Does the UI implement an INSERT for every table in the schema?* In other words, there should be UI input fields that correspond to each table and attribute in that table.

There is a way to insert into every table except for Rosters.

- *Does each INSERT also add the corresponding FK attributes, including at least one M:M relationship?* In other words if there is a M:M relationship between Orders and Products, INSERTing a new Order (e.g. orderID, customerID, date, total), should also INSERT row(s) in the intersection table, e.g. OrderDetails (orderID, productID, qty, price and line_total).

Yes, this is handled well for games and teams, as each requires a league id to be associated with it.

- *Is there at least one DELETE and does at least one DELETE remove things from a M:M relationship?* In other words, if an order is deleted from the Orders table, it should also delete the corresponding rows from the OrderDetails table, BUT it should not delete any Products or Customers.

There isn't a way to delete a player from a team currently.

- *Is there at least one UPDATE for any one entity?* In other words, in the case of Products, can productName, listPrice, qtyOnHand, e.g. be updated for a single ProductID record?

There appears to be no way to update any entities once created. For example it might be a good idea to be able to update a player's contact information if needed.

- *Is at least one relationship NULLable?* In other words, there should be at least one optional relationship, e.g. having an Employee might be optional for any Order. Thus it should be feasible to edit an Order and change the value of Employee to be empty.

It doesn't appear that there is an optional relationship. I think a way to implement this would be to have something like teams and leagues be optional. Maybe there is a scenario where a team has been created but it isn't associated with a league yet, or it can be removed from a league and also not be associated with a league

- *Do you have any other suggestions for the team to help with their HTML UI?*

I think the UI is good, it provides good functionality to create entities. It might be a good idea to add a way to put a player on a roster, and be able to edit rosters - maybe through having a "edit team" functionality. Also it might be good to have a way to edit and display entities like games (in the event that a game is canceled, time changed, or to see a schedule of upcoming games) or leagues (maybe a team switches leagues or you want to view the teams that are participating). I like the idea, you have a nice straight forward approach, and the current UI is easy to understand. I like how your team used a drop down menu to limit the foreign keys to what is available in the corresponding tables. Thank you for sharing, I wish you luck with your project!

Action based on feedback

1. We added the Rosters table to the front end

Upgrades to the Draft version

1. We deleted unique values off of the Rosters table from the foreign keys
2. We removed the emergency contact name and number attributes from the Players area

Feedback by the peer reviewer from step 2

1) Overview does a good job of describing the contents of the project but not necessarily the question it hopes to answer. The overview describes expected size of the use base and the entities that will be implemented. Schema could use a rework, the way the arrows are drawn is a little confusing. Names and syntax are consistent through out the outline. All of the relationships seem to be set up correctly. Overall I think this is a great start and I wish you guys good luck moving forward!

2) Hey Group #29!

https://jlvr-cs340-project.herokuapp.com/

- Does the overview describe what problem is to be solved by a website with DB back end?
  The overview doesn't really explicitly say what problem the database is solving. More explicit reference to how the database would store the necessary info would be helpful.
- Does the overview list specific facts?
  The overview is clear and describes specific facts about what is necessary for the database.
- Are at least four entities described and does each one represent a single idea to be stored as a list?
  Their are more than four entities and each encodes one idea.
- Does the outline of entity details describe the purpose of each, list attribute datatypes and constraints and describe relationships between entities?
  The outlines are good and describe the entities, they don't really reference constraints. It would be good to also have a note around constraints in the outline.
- Are 1:M relationships correctly formulated? Is there at least one M:M relationship?
  Cardinality is correctly formatted. The M:M relationships are correctly formatted as intersection entities.
- Is there consistency in a) naming between overview and entity/attributes b) entities plural, attributes singular c) use of capitalization for naming?
  The naming and capitalization seems consistent

3) **Does the overview describe what problem is to be solved by a website with DB back end?**

The overview provides a description of what is going to be store in the database. I suggest addressing the problems at first and then describe how it’s going to be solved by a website with a database.

- **Does the overview list specific facts?**

Yest, the overview provides a clear description of the constraints of the entities and attributes

- **Are at least four entities described and does each one represent a single idea to be stored a s a list?**

Yes, there are more than 4 entieres and they all represent a single idea.

- **Does the outline of entity details describe the purpose of each, list attribute datatypes and constraints and describe relationships between entities?**

Yes, the outline provides a clear description of the constraints, and the datatypes. However the relationship between Games and Team is not clear. There are two M:M relationships between Games and Teams; the direct relationship which is not valid and the other which is valid and that is intersected by the entity League. Also the entity Rosters and Matches are not included in the ER diagram.

- **Are 1:M relationships correctly formulated? Is there at least one M:M relationship?**

Yes, there is one M:M relationship which is Games:Teams intersected with the entity League

- **Is there consistency in a) naming between overview and entity/attributes b) entities plural, attributes singular c) use of capitalization for naming?**

Yes, there is a pattern of consistency

Overall, great work. Thanks for sharing !

Actions based on the feedback

Overall the feedback on our project was positive and we did not get many suggestions from the peer reviewers on things we should change. A consistent suggestion was to do a better job describing the problem our database is solving. We improved the description of the problem we are solving with this database.

Upgrades to the Draft version

Besides the clearer description of the problem we are solving with this database, we have updated some other things as well. The Matches table has been eliminated and we expanded the Games entity. Games will now include two attributes corresponding to team1 and team2. This works well because a Game will only consist of 2 teams, never more or less. The M:M relationship we had established between Games and Teams is now gone.

Another thing we updated based on feedback from our grader (thanks Andrew, the feedback is very helpful) was that we were using the “unique” attribute incorrectly. We had a misunderstanding that “unique” was only specific to the row, not the whole column. This has been updated. There was also a circular relationship with Teams that was confusing in our original draft, this has been removed with the elimination of Matches.

Project Outline and Database Outline - Final Version:

Organizing sports leagues across many different sports and leagues can be a big problem to manage. Using a piece of paper or Excel sheet can work fine if you’re only hosting a weekend tournament, but how about if you have hundreds of teams and thousands of players competing in leagues all throughout the year? That requires a database to keep track of all the different entities and the relationships between them.

Our database solves that problem. We are creating an organization called Fun Sports Leagues that hosts social and competitive sports leagues in 10 different sports. Players will be able to use our website (powered in part by a database) to signup for leagues and teams. Each sport will have a maximum of 8 different leagues going on at one time (for a maximum of 80 distinct leagues), and each league will have a maximum of 24 teams. The number of players on a team will vary depending on sport, however, the maximum number of players on a team will be capped at 30. Players will be allowed to be a member of as many teams as they are eligible to participate in.

Sports will be offered in a variety of different leagues. For example, basketball has categories for male, co-ed and female, as well as options for competitive or social. The database driven website will record *Leagues, Players, Teams, and Games.*

Leagues: records the type of sport and league specifications.

- leagueID: int, auto_increment, unique, not NULL, Primary Key
- sport: varchar, not NULL
- gender: varchar, not NULL
- level: int, not NULL
- startDate: date, not NULL
- endDate: date, not NULL
- Relationship: 1:M between Leagues and Games is implemented with leagueID as a foreign key inside Games.
- Relationship: 1:M between Leagues and Teams is implemented with leagueID as foreign key inside Teams.

Players: records the information about a player. Has multiple relationships with Teams.

- playerID: int, auto_increment, unique, not NULL, Primary Key
- firstName: varchar, not NULL
- lastName: varchar, not NULL
- gender: varchar, not NULL
- birthday: date, not NULL
- email: varchar, not NULL
- phoneNumber: varchar
- Relationship: M:M between Players and Teams is implemented with playerID and teamID as foreign keys in a join table called Rosters.
- Relationship 1:M between Players and Teams is implemented with playerID as a foreign key inside Teams called capitanID

https://jlvr-cs340-project.herokuapp.com/

Teams: records information about the team. Has multiple relationships with players, a relationship with Leagues, and a relationship with Games

- teamID: int, auto_increment, unique, not NULL, Primary Key
- leagueID: int, unique, Foreign Key
- capitanID: int, unique, Foreign Key (corresponds to playerID)
- teamName: varchar, not NULL
- wins: int, not NULL
- losses: int, not NULL
- Relationship: M:1 between Teams and Leagues is implemented with leagueID as foreign key inside Teams.
- Relationship: M:M between Players and Teams is implemented with playerID and teamID as foreign keys in a join table called Rosters.
- Relationship: 1:M between Players (team capitan) and Teams is implemented with playerID as foreign key in Teams.

Games: records information about an individual game including location and time. Has relationships with Leagues, Teams.

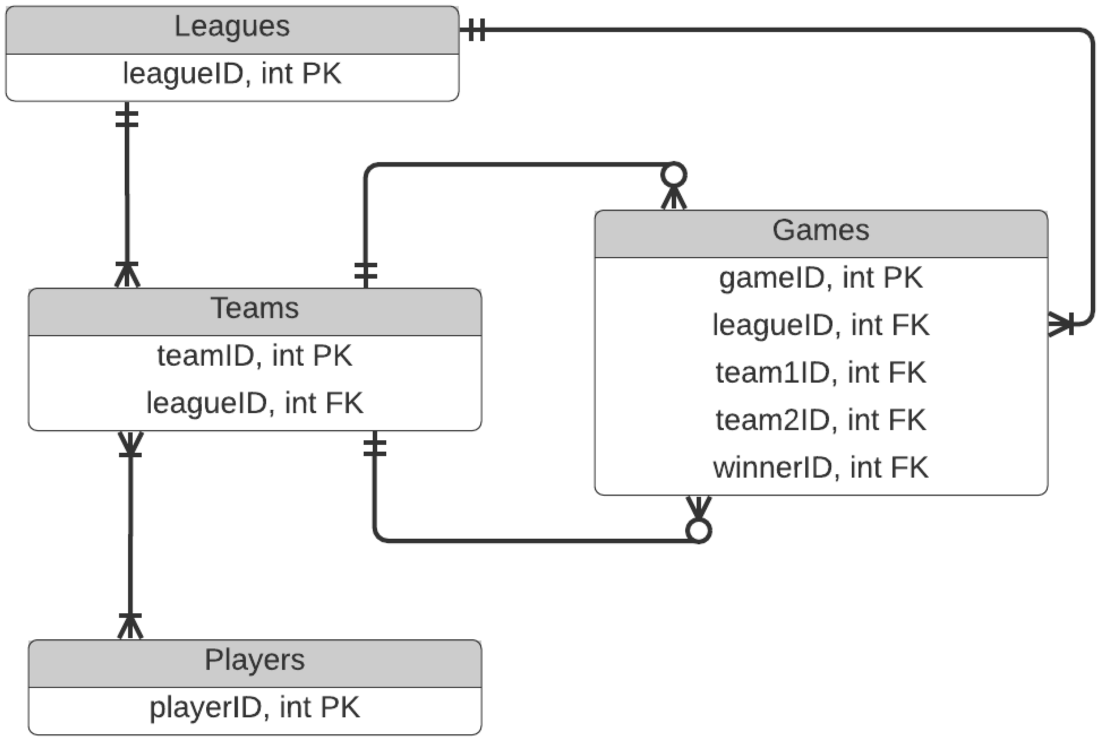
- gameID: int, auto_increment, unique, not NULL, Primary Key
- leagueID: int, unique, not NULL, Foreign Key
- gameLocation: varchar, not NULL
- team1ID: int, not NULL (corresponds to teamID)
- team2ID: int, not NULL (corresponds to teamID)
- gameDate: date, not NULL
- gameTime: time, not NULL
- isPlayoff: boolean (or TINYINT)
- gameWinnerID: int, (corresponds to teamID)
- Relationship M:1 with Leagues is implemented with leagueID as foreign key in Games.
- Two M:1 relationships with Teams, with two teamID's entered as foreign key in Games.
- Relationship M:1 with Teams, with gameWinnerID entered as foreign key in Games

Rosters: (intersection table) records information about the relationship between players and teams.

- playerTeamID: int, auto_increment, unique, not NULL, Primary Key
- playerID: int, not NULL, Foreign Key
- teamID: int, not NULL, Foreign Key

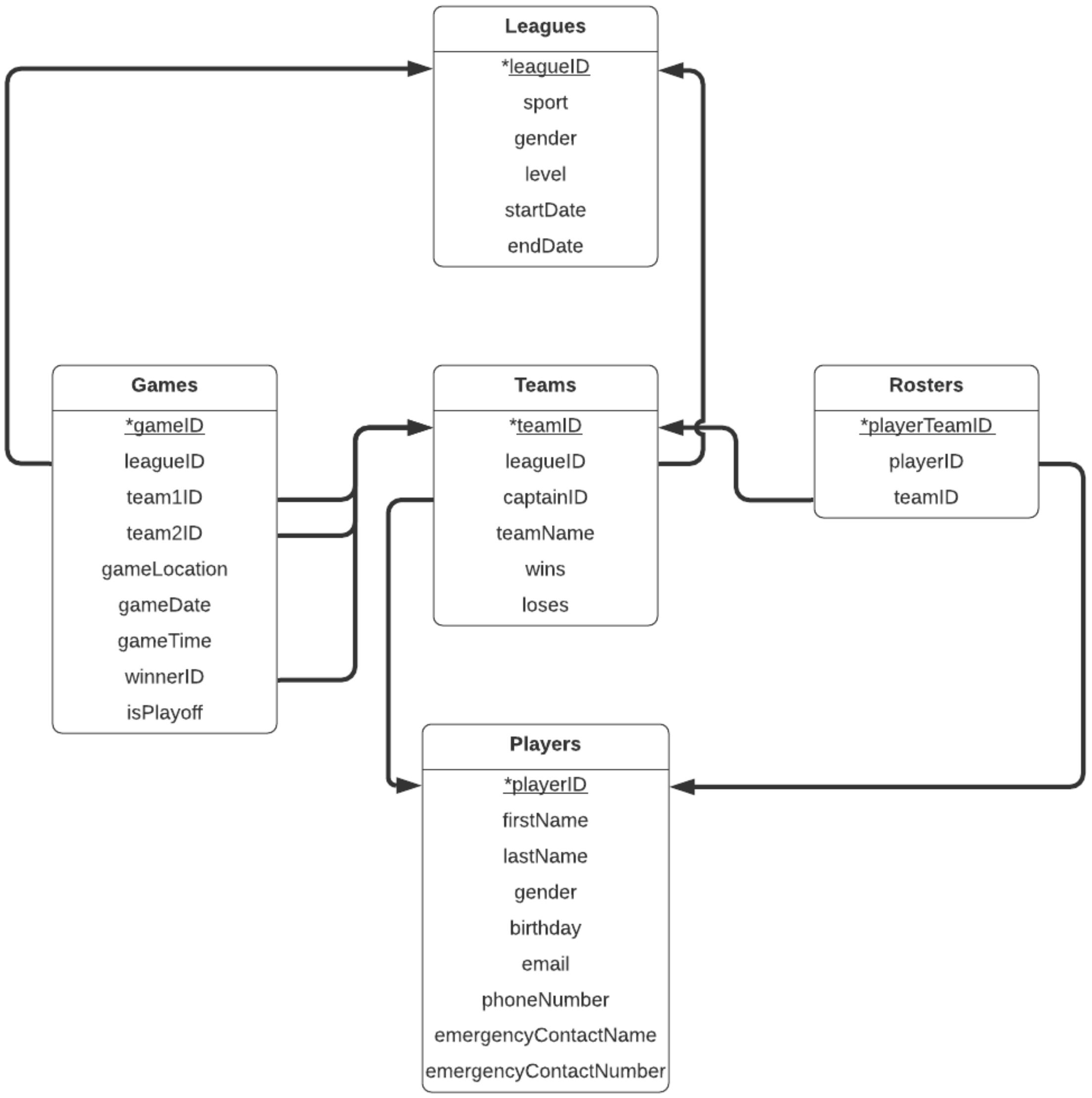
https://jlvr-cs340-project.herokuapp.com/

c) Entity-Relationship Diagram



c) Schema

https://jlvr-cs340-project.herokuapp.com/


Leagues
*leagueID
sport
gender
level
startDate
endDate
Games
*gameID
leagueID
team1ID
team2ID
gameLocation
gameDate
gameTime
winnerID
isPlayoff
Teams
*teamID
leagueID
captainID
teamName
wins
loses
Rosters
*playerTeamID
playerID
teamID
Players
*playerID
firstName
lastName
gender
birthday
email
phoneNumber
emergencyContactName
emergencyContactNumber