URL:

http://web.engr.oregonstate.edu/~hwangk/

One Piece Database

Feedback by the peer reviewer (Mayu Morita):

Data Manipulation Queries 1. Are the queries syntactically correct? Disregard the part where input will be substituted as shown in the sample sample_database_manipulation_queries.sql. Minor errors on: SELECT `ID`, `name`, `leader_id`, `population`, `alignment`, `ship`, `arc_id`, FROM `crew`; Extra comma after `arc_id` INSERT INTO `crew_devil_fruit`(crew_id, devil_fruit_id) values((select crew_id from crew where name = [crew name]), (select devil_fruit_id from devil_fruit where ability = [ability]); Missing) to close out the values Checked on a syntax checker, including the bracketed area with fake data 2. Are there queries providing all functionalities as required by the CS340 Project Specs ? All of the queries required are present. HTML Pages 1. Does each functionality listed in the CS340 Project Specs have a corresponding HTML page? (It's okay to implement multiple functionalities on the same HTML page) All of the functionality is present required in the specs. 2. Is there a better way that data could be displayed on SHOW functionality pages? While it's not in the specs for step 3, and something that will most likely happen in next steps, will be to add colors and layouts that are a little more aesthetically pleasing. 3. Is there a better way that the forms for UPDATE and ADD functionalities could be implemented? Something to consider would be maybe making the options to be drop downs instead of free form boxes. I think functionally it works the samxe, but that way there isn't any issues of a user typing in a name incorrectly.

Feedback by the peer reviewer (Chelsea Egan):

DMQ 1. Are the queries syntactically correct? Disregard the part where input will be substituted as shown in the sample sample_database_manipulation_queries.sql. YES 2. Are there queries providing all functionalities as required by the CS340 Project Specs ? The only queries I didn't see was for searching, unless that was being implemented with one of the Select statements. I also didn't see a statement for deleting a many-to-many relationship. HTML 1. Does each functionality listed in the CS340 Project Specs have a corresponding HTML page? (It's okay to implement multiple functionalities on the same HTML page)I did not see a way to select from any tables except for character. I also am not sure if the Select a Character page satisfies the requirement for a search/filter functionality. Finally, I did not see an option to add or remove things from a many-to-many relationship. 2. Is there a better way that data could be displayed on SHOW functionality pages? None of the pages seem to display the contents of any of the tables. 3. Is there a better way that the forms for UPDATE and ADD functionalities could be implemented? It would be cool to have drop down options for those fields that are foreign keys to other tables. The update character page – it is not clear what I'm supposed to enter under "input update."

Feedback by the peer reviewer (Xavier Hollingsworth):

The attribute for each entity in the ERD are the same as those outlined in the database outline. The relationships of the entities are accurately reflected. The cardinality of the entities in the relationships are the same as they are described in the outline. The entity-relationship diagram is well drawn out but it is very small and convoluted, making it very hard to tell what is going on in the diagram. Although it is correctly drawn based on the general outline, it took me a long time to read and understand. Another thing is there are a lot of many-to-many relationships which causes problems in relational databases. I would recommend to change the several many-to-many relationships you have in your ERD. Having many-to-many relationships causes many problems in relational databases so I recommend creating a third entity which would be a composite entity. Between these three entities you would then have two one-to-many relationships instead of one many-to-many relationships. I recommend doing this for each many-to-many relationship in your diagram. Although it will make your diagram even larger and more convoluted it will fix the problems stated. For the schema, the relationship tables are correctly defined compared to the database outline. The foreign keys are present and correctly defined. The entitiy attributes match those defined in the outline. The schema looks well done to me although the arrows are sometimes hard to follow since they seem to go all over the place. The SQL file is very well written and easy to read. It appears syntatically correct and runs in phpmyadmin. The data types are appropriate for their corresponding attributes. The foreign keys are correctly defined although there seems to be no constraints. The relationship tables are not present and written so I would fix that and add any constraints you need to. Overall, you did good job on your Step 2 draft. Some things I would fix are the several many-to-many relationships you have, try and make the diagrams easier to read and larger, and create the relationship tables that are described in your outline.

Feedback by the peer reviewer (Jacob Carter):

Are the attributes for each entity in the ERD same as that described in the database outline? a. Yes, all the attributes are present and in the correct locations. 2. Is the participation of entities in the relationships same as that described in the outline? a. The participation is perfectly labelled. 3. Is the cardinality of entities in the relationships same as that described in the outline? a. The only question I have is regarding the cardinality between the crews and devil fruit. Your outline states “each crew has several devil fruit users” -- is this a situation in which there should be total participation? In other words, will there ever be a crew with no devil fruit? If not, the line should be bolded. 4. Is there something that could be changed/improved in the E R Diagram and/or the overall database design? a. Nothing of real substance that I can see. Maybe slightly more descriptive relationship descriptions (ie. “A Character wields/uses/etc. a Devil Fruit), but that’s very nitpicky of me. Good job!

The best peer review for a Schema would answer all of the following questions: 1. Are the relationship tables present where required and correctly defined, when compared with the database outline? a. Yes, the tables are present and correct, including the tables required for the potential many-to-many relationships. 2. Are foreign keys present where required and correctly defined, when compared with the database outline? a. Yes, foreign keys are defined and easy to

follow. 3. Do the entity attributes match those described in the outline. a. Yes, no problems here. 4. Is there something that could be changed/improved in the Schema and/or the overall database design? a. Again, not much from a substantive standpoint. The haphazardness of the lines could maybe be cleaned up by making a table (such as seen in the schema lecture) with the arrows running between rows. Even easier, copy the "designer" diagram from phpmyadmin. That being said, it is easy to trace as is and shouldn't hinder the project outside of a design standpoint.

The ideal peer review for a DDQ file would answer all of the following questions: 1. Is the SQL file syntactically correct? This can be easily verified by importing/copy-pasting it in phpmyadmin. (Do not forget to take backup of your own database before you do this!) a. The file is mostly correct. I have pasted the error message found in an image below so that you can work to fix it up. Are the data types appropriate considering the description of the attribute in the database outline? a. If populations are big enough to warrant it, consider switching from int to bigint. 3. Are the foreign keys correctly defined when compared to the Schema? a. Yes, they are well defined. You will probably want to define what will happen to them upon deletion and update to avoid issues moving forward. 4. Are relationship tables present when compared to the ERD/Schema? a. Yes, all tables are present and accounted for.

Overall, very good work. I like both the concept of your project and it's execution so keep it up. Best of luck with the rest of it!

Feedback by the peer reviewer (Aaron Journot):

Outline

Is the outline useful for you to understand broadly, the universe that the database is set in? Yes, the outline provides a good description of the Universe that the database will represent. I have never read One Piece, but I have played some of the games! Regardless, I think the description is accurate and detailed for the topic. Is there something else that you wish was mentioned in the outline to understand the topic better? No, the description is well written.

Database

Are there sufficient entities as required? Yes, there are 4 Entities, meeting the required 4. Are there sufficient number and types of relationships as required? Yes, there are 5 Relationships, surpassing the required 4 with at least one many-to-many. Is there a good reason to have the things described as entities to be entities? Or can they be just attributes of other entities? Yes, the entities described are necessary to be used as entities rather than attributes or relationships. The only entity that was difficult to see how it would fit into the database was Arcs. While I do understand the purpose and design of the Arc entity, it is hard to grasp the idea of a story arc as an 'element' of the universe. However, the descriptions of the attributes for each arc makes it describable to a point where it could be used as an entity. Do the various relationships make sense to you? Yes. The relationships are well defined and include details regarding how each entity interacts with the others. One question I have is whether or not Crews ever retain the devil fruit to pass around to crew members. In essence is there a crew-to-devil fruit relationship that needs defined or is the fruit, once picked up, not transferred to other people? Is there sufficient

information about each attribute to justify its presence? Absolutely. The attributes are well defined and provide all necessary info required to understand their purpose. Are the data types mentioned for attributes? It need not be an actual keyword from MySQL/MariaDB like "varchar" but is it sufficiently described as a string, number, date, etc.? In all instances, yes. Well done!

Do the data types make sense for the attributes? Yes, most of the data types appear to be well-thought and make sense for the application. Are constraints described for attributes? Do they make sense? The implementation details of the constraints do not matter right now. Yes, the constraints are noted in almost all of the attributes. Is there anything else that you wish the student did to make their outline better and more suitable for a CS340 Project? No, I think the universe is well defined and even without knowing more about the graphic novel (because I'm sure the games are different), I could probably describe to you the framework of the universe.

Feedback by the peer reviewer (Brendan Corazzin):

Nice work. The description is very detailed and I fully understand the mini-universe your database describes. Just a few comments: -You could still break name into first_name and last_name. Last_name could default to NULL and characters with single names would only have a "first name". This may be a better design because you don't know at this point if your application will ever need to reference just the first name of a character. You could write a regex function to split names, but this may be less reliable than storing first and last name separately. For example, if a character has a first name made up of two words that is split with a space, and a last name, you would have a three word name, like 'Mary Kate Olsen'. How would you get just the first name in this instance? You could hyphenate it, but if users ever contribute to the data, they may not follow the patterns you'd expect. -For the many to many relationships, I think you'll need separate tables to store this data. This is similar to the lecture example regarding the bsg_people, bsg_cert, and bsg_cert_people tables. I think the lecture is titled "keys". For example, the characters to crew relationship could be stored in a characters_crew table that holds a primary id key, a foreign key to the character, and a foreign key to the crew. You may have been planning this already, but it wasn't clearly outlined in the database description. -The devil fruit will need a primary key and the relationship should probably be stored in it's own table (as described in the previous comment). -Arc will also need a primary key. The intro attribute is unnecessary, as the arc id is stored with the character. To get all the characters associated with an arc, you would just need to write a query to do that. For example 'select * from characters where debut = arc_id', would give you that data. Another comment: I'd also consider naming attributes that reference other ids as 'X_id' where X is the thing it references. So instead of 'debut', you'd have arc_id. Naming clarity will help a lot when writing queries later on in the application code.

Actions based on the feedback (Mayu Morita):

Mayu's first notice was that two of my queries, one SELECT and one INSERT, had minor syntax issues. I have gone and addressed those issues in my code. Mayu also suggested that I maybe use CSS and things of the like to add color or other things to make the site more aesthetically pleasing. I chose not to make such edits, as this project is meant to test our

understanding our databases and not really the understanding of front end web design. Also, the class instructor addressed this on Piazza and stated that ”CSS is not expected but data should be presented at least in table form.” Therefore, I don’t believe it is necessary to change the appearance of my website. Ultimately, Mayu mentions that it might be better to make the input boxes drop down menus instead of free form boxes. I believe that this only applies for certain queries, and not necessarily for others such as “ADD”, which would require a free form box for the user to input the data. For the fields that I saw fit (mainly UPDATE functions), I have added a drop down menu from which users can choose which field they wish to update.

Actions based on the feedback (Chelsea Egan):

Chelsea mentioned that I did not have a search functionality, which I have since added to my DMQ file. This function uses the WHERE clause to search the database for certain selections based on the user’s input. I have also added DELETE functions for each table of my database, which includes many-to-many relationships. This addresses Chelsea’s second concern. I have also added ways to select (view) any table, and my search function should now fulfill the search/filter requirement. I have also added ways to add, remove, and update things or rows from my many-to-many relationships. I have also put text indicating where future data output will be placed when my website is connected to my database later on. For now these are simply placeholders. I have also added drop down menus for organization and also have updated the text to say “Input Updated Value” instead of “Input Update” for further clarification.

Actions based on the feedback (Xavier Hollingsworth):

The main changes that Xavier suggested were to organize my ERD to be more clear, make changes to my numerous many-to-many relationships so that they aren’t so plentiful, add foreign key constraints to my code, and to add relationship tables. I have edited my ERD to be much clearer and spread out my ERD so that each entity has more space for visibility. I believe my relationships tables are correct, so I did not make any changes to that (have 3 relationship tables for each of my 3 many-to-many relationships). I also did some research and also asked fellow students on Piazza, who believe that 3 many-to-many relationships is not excessive and does not require any change. So I will not change anything regarding this. I have also edited my code to reflect the foreign key constraints so that on delete, they will be set to NULL and on update they will cascade.

Actions based on the feedback (Jacob Carter):

Jacob did not have too many change suggestions. He had one question regarding my crew-devil fruit relationship, which warranted no change. He suggested that I edit some of my relationship names to be more specific, and I have done so. Jacob also suggested that maybe I change “population” from int to bigint, but the numbers would be finely contained within int’s limitations. I have also changed my SQL code to alleviate the errors specified. I was unable to figure out all of the errors in time.

Actions based on the feedback (Aaron Journot):

Aaron mentioned how the “Arc” entity seems non-concrete and a bit strange as an entity, but Arcs are essentially synonymous to physical locations within the series, as each arc is associated with a brand new geographical location. I have chosen to keep this entity name. Also, Aaron asked of the crews ever retain the fruit to pass around to crew members. This cannot occur because the devil fruit can only be used by one user at a time. I have added this to my description of the devil fruit entity.

Actions based on the feedback (Brendan Corazzin):

Brendan suggested that I break the name attribute up into last and first names. The reason I choose still not to do this is because some, if not most characters do not go by their birth given name, but rather by nicknames or titles. It seems illogical to me for that system to work with a first name, last name system. For example, a character of the name Big Mom would be first name Big, last name Mom. This doesn’t make sense to me, so I will keep it as one name attribute. Brendan also suggested splitting the many-to-many relationships into separate tables. I had this thought, but did not know how to implement it within the outline. This aspect of my database will be more visible within the ERD and schema section of part 2. Brendan also suggested that I include a primary key for the Devil’s Fruit and Arc entities, which I had forgotten to do. Removed the redundancy of having an intro attribute when the debut attribute does the same thing. Adjusted my outline to change certain attribute names to “xxx_id”, for more clarity of relationship between tables.

Upgrades to the Draft version:

I added a couple of more constraint descriptions to the data types (for example, max character limits). The biggest change is that I added another relationship (crews have devil fruit). This relationship describes the total of devil fruit wielders that each crew has. Before this relationship, each entity had some sort of relationship with every other except for the relationship between crew and devil fruit, so I decided to add it. Other than that, most of my changes were from the help of my peer reviewers. I also realize that many-to-many relationships and their attributes are not well-stated in my outline, but my ERD and schema will develop on those. For now, I have moved the attributes associated with the many-to-many relationships to be found underneath each respective relationship.

Upgrades to the Draft Version (Part 2):

Regarding the outline, I have very little upgrades at this stage of the project. The biggest of these changes would be the updating of my ERD to allow for more clarity, which was mentioned in the peer review actions taken. I believe that this was not wholly necessary but will help future readers to understand what my diagrams portray. My SQL file was also updated to have minimal and to reflect foreign key constraints on delete and update, which are very big changes. I still could not figure out how to fix the error that says “foreign key constraint is incorrectly formed”. Other than those listed, I made very minor changes to the outline as a whole.

Upgrades to the Draft Version (Part 3):

This time around, I made quite a few changes to both my HTML files and my DMQ file. Starting with the DMQ file, I added several new functionalities for my database, the most notable ones being a new search function for searching a character by their text name, update functions for every single table with drop down menus for each desired field, and also functionalities across the board for relationship tables. My DMQ file was very skeletal and basic when I turned it in for the draft version, and it is now more fleshed out.

As for my HTML file, I made the corresponding changes necessary to accommodate the changes I made in my DMQ file. I also categorized each link based on its function, and overall made the site look a bit cleaner. The website, much like my DMQ file, was at first very barebones but now has a lot more content.

Project Outline:

The topic of my project this term will be the Japanese graphic novel series One Piece (there is also an adapted animated series). One Piece describes a fictional and expansive fantasy world where Gol D. Roger, the Pirate King, initiated a revolution of piracy and inspired the next generation of pirates to go on an expedition to find "One Piece", the treasure that he claimed to have hidden during his journey. I picked this topic because One Piece is one of my favorite pieces of fictional literature, and the expansiveness of the universe allows for a very thorough and detailed database. My database will not include every single character of the series (there are simply too many), but will include every character that is recurring and/or deemed influential to the main plot of the series. There will be an attribute called "devil fruit", which refers to a plot point in the series where magical fruit spread across the world grant the consumer superhuman abilities when eaten. Some characters have these abilities, and for these characters they will be listed as an attribute, otherwise the attribute will read NULL.

Database Outline, in Words:

My database will have the following entities:

- Characters – The character entity will be the foundation of this database. This entity will describe each individual character of the series, and will have the following attributes:
    - *id*: This will be the primary key attribute for the character entity. This is a number that will be assigned to each character when they are included in the database. This number will auto-increment with each inclusion, so that each character has its own unique id. The data type will be int.
    - *name*: This is the name of the character. Initially, I thought I would divide this attribute into first and last name, but certain characters in the series only have a single-word alias, and for organization and universality's sake, I went with the one name. This attribute cannot be NULL (a character must have a name) and there is no default. The data type will be string (max 50 char).

    - *arc_id:* This attribute describes when a character was introduced into the series. This cannot be NULL, as each character must be introduced at a certain point. The data type will be the id of an arc entity.
    - *bounty:* This attribute describes the bounty of a character. Due to the pirate nature of this series, several characters are considered outlaws and therefore have bounties. If a character does have a bounty, it will be listed here. If a character is law-abiding or yet to receive a bounty, they will have NULL. The data type will be int.
- Crew - This entity can be described as the group with which the character is associated with. Within this entity, you can find pirate crews, organizations, hometowns, or even NULL if a character has no affiliations. This entity has the following attributes:
    - *id:* The primary key attribute for the crew entity. This number will auto-increment each time a crew is added to the database, and will serve as the unique number identifying each crew. The date type will be int.
    - *name*: This is the name of the crew. This attribute cannot be NULL (a crew must have a name) and there is no default. The data type will be string (max 50 char).
    - *leader_id:* The leader attribute will be the id of the character who leads the group. For a pirate crew, this is their captain. A crew can potentially not have a leader (either it is a group effort or the leader has not yet been revealed), and in this case leader will be NULL. The data type will be an id that points to a character.
    - *population:* This attribute gives the population of the crew. This will not include minor or unnamed characters (for example, the "marine" crew will list only the major marines, otherwise there would be thousands). Cannot be NULL. The data type will be int.
    - *alignment*: The alignment attribute will only have three options: good, bad, or neutral. This describes, from the point of view of the protagonist of the series, whether the crew is good or bad. One Piece is unique in that the protagonist is a pirate, so several pirate crews will be good and certain law-enforcing organizations, such as the Navy, can be labeled as bad. Cannot be NULL. The data type will be string (max 10 char).
    - *ship*: This attribute will give the name of a crew's ship. If the crew is not a pirate crew or the name is unknown, it will be NULL. The data type will be string (max 50 char).
    - *arc_id:* This attribute describes when a crew was introduced into the series. This cannot be NULL, as each crew must be introduced at a certain point. The data type will be the id of an arc entity.
- Devil Fruit – This entity describes the numerous "devil fruit" that exist in the One Piece universe. As previously stated, these fruits give their consumer magical, superhuman abilities. A devil fruit can only be used by one user at a time. This entity has the following attributes:
    - *id:* This is the primary key attribute that will create unique ids for each fruit. This is an auto-incrementing integer.

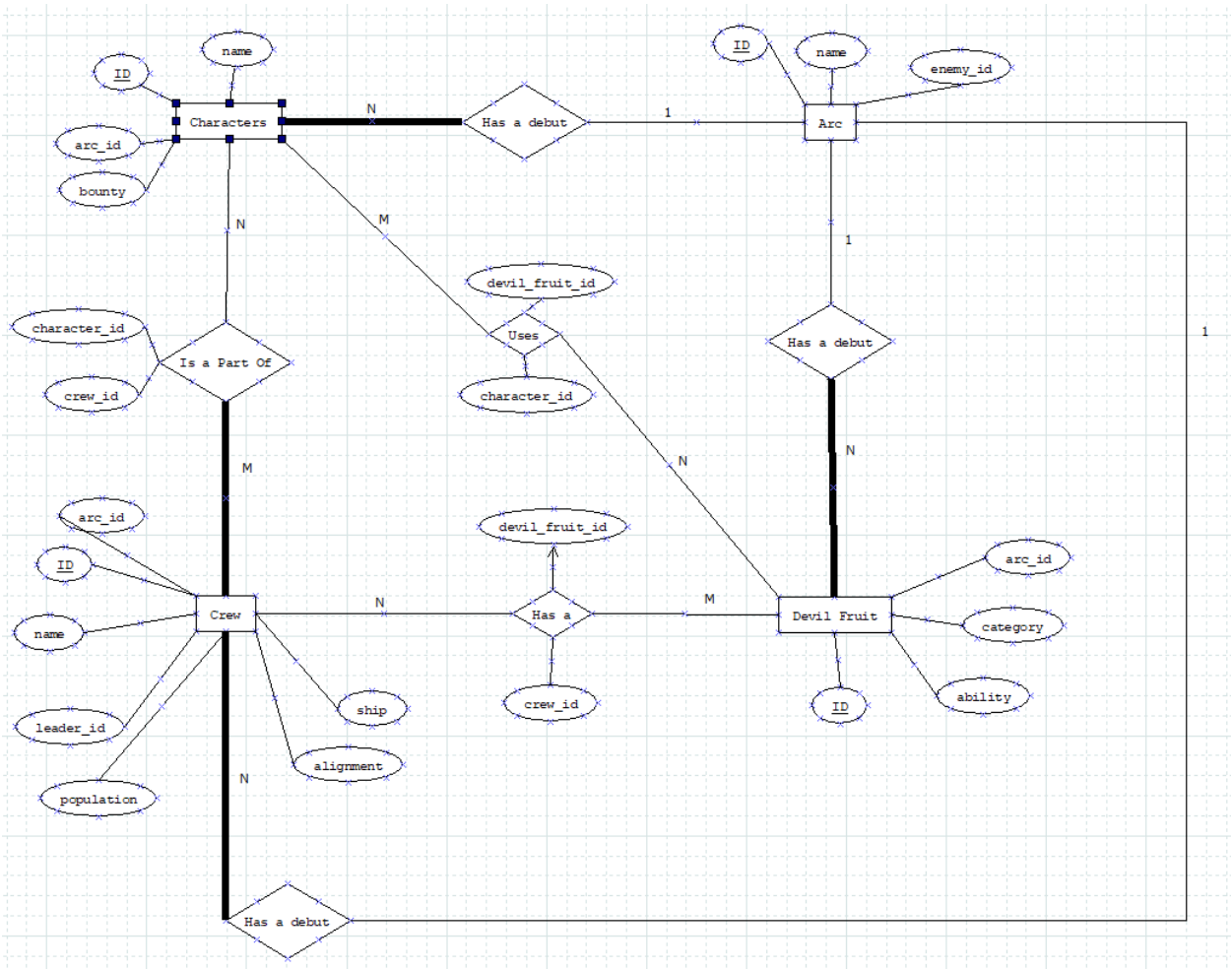
    - *ability:* This defines what ability the fruit bestows upon its consumer. This cannot be NULL, as each fruit has some form of power. The data type will be a string (max 50 char).
    - *category:* This defines the type of fruit. There are three main categories of fruit: paramecia, logia, and zoan. This cannot be NULL as a fruit must have a category. The data type will be a string (max 50 char).
    - *arc_id:* This attribute describes when a devil fruit was introduced into the series. This cannot be NULL, as each devil fruit must be introduced at a certain point. The data type will be the id of an arc entity.
- Arc – This entity describes the arcs of the One Piece story. By "arc", I mean individual, separable sub-stories that comprise the series. This entity has these attributes:
    - *id:* This is the primary key attribute that will create unique ids for each arc. This is an auto-incrementing integer.
    - *name:* This is the name of the arc. This cannot be NULL as each arc has a name. The data type will be a string (max 50 char).
    - *enemy_id:* This will be the big, main antagonist of each arc. For pirate crews, this will be noted as simple the name of the captain. For other antagonists with a less specific "main" enemy, there can be multiple listed. This cannot be NULL. The data type will be the id or id's of the antagonists.

The relationships in my database are:

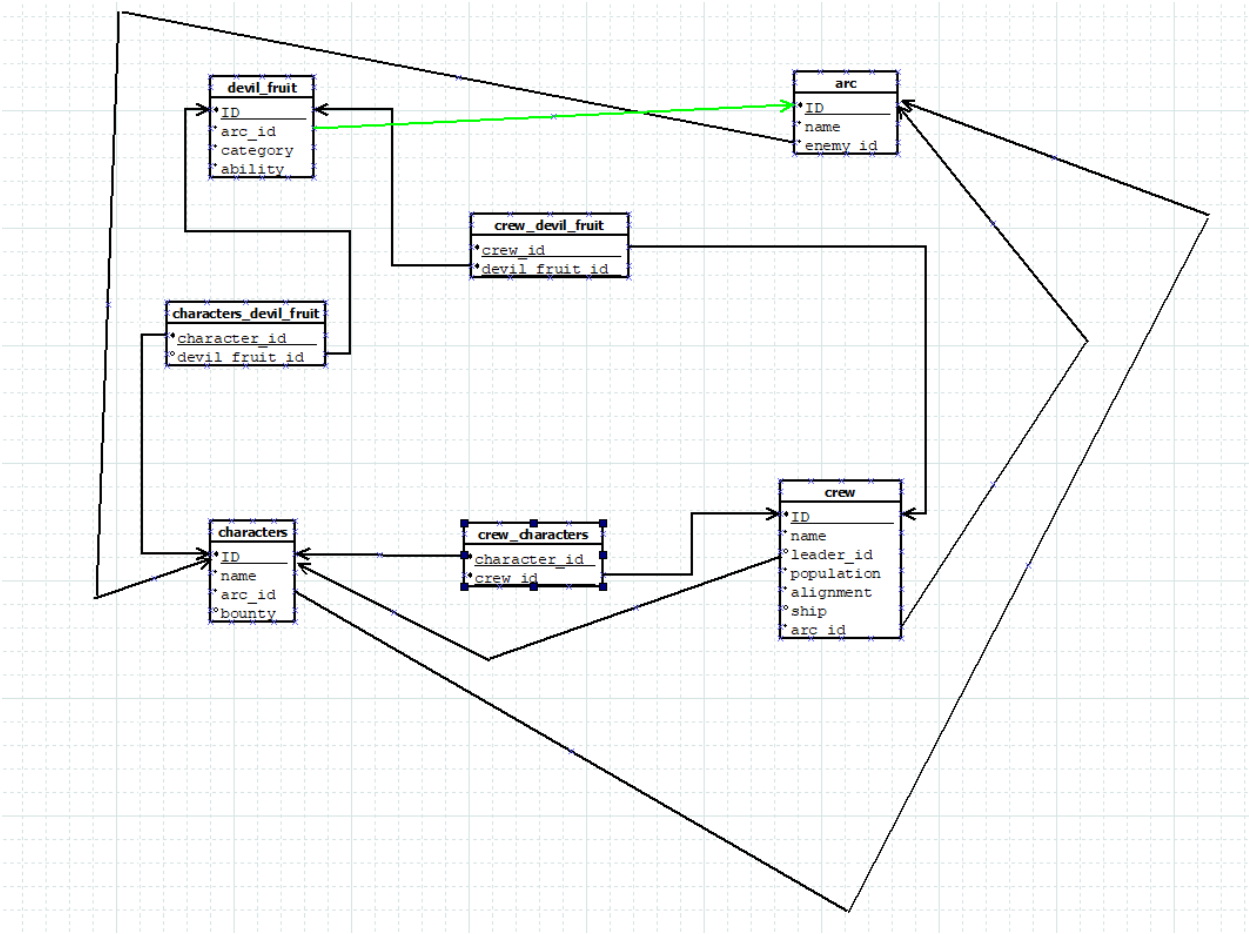
- Characters have crews – A character can be a part of multiple crews (could have left a crew for another, or simply be part of multiple at once) and a crew can have multiple members. The Crew and Character entities are often a one-to-many relationship, but they can occasionally be many-to-many.
    - *crew_id:* This attribute describes the affiliations a character has within the series. Depending on the type of character, this can vary greatly. For example, a pirate's crew_id attribute will name his/her pirate crew. A marine, on the other hand, will simply list "Navy". There can also be characters with no affiliation, and these will have NULL. This attribute will contain the id of the crew entity that this character is affiliated with. A character's affiliation must either be NULL or exist within the crew entity defined in our database. The data type will be an id that directs to a crew.
    - *character_id:* This is a record of a crew's characters. This cannot be NULL, as a crew must have at least one character. The data type will be an id of a character(s).
- Characters have devil fruit – In general, a character can only have one devil fruit. There is one exception, a character named Blackbeard who has shown to mysteriously be able to wield multiple fruit abilities. Also, a fruit can have multiple users at different points in time. Therefore, this relationship between Character and Devil Fruit is normally one-to-one, but it can also sometimes be one-to-many or many-to-many.

    - *devil_fruit_id:* This is the attribute that describes the devil fruit ability of a character. If a character has one of these abilities, it will be listed. There are, however, “normal” human beings in the series as well, who do not have such an ability. For those characters, this attribute will default to NULL. The data type will be the id of a devil fruit entity.
    - *character_id:* This is a record of the specific fruit’s users. Each fruit only has exactly one user at a time (if a user dies, the fruit is reincarnated into a new fruit somewhere in the world). There are fruits within the series that have been used by more than one user (not simultaneously). This can potentially be NULL, as fruits can remain unfound or uneaten. The data type will be an id of a character(s).
- Characters have debut arcs – Every character has made their debut into the series at some point in time. The relationship of Character to Arc is many-to-one, as a character only has one debut arc but that arc can be the debut of several characters.
- Crews have debut arcs – Every crew has made their debut at some point. The relationship of Crew to Arc is many-to-one, as a crew only has one debut arc but that arc can debut several crews.
- Devil Fruit have debut arcs – Every devil fruit has to make a debut at some point. The relationship of Devil Fruit to Arc is many-to-one, as a devil fruit is only debuted once but that arc can debut several devil fruit.
- Crews have devil fruits – Each crew has several devil fruit users. This relationship is many-to-many, as many crews can have one devil fruit (not all at once) and a crew can have many devil fruits.
    - *devil_fruit_id:* This is the attribute that describes the devil fruit ability of a character. If a crew has one of these abilities, it will be listed. For those crews without devil fruit abilities, this attribute will default to NULL. The data type will be the id of a devil fruit entity.
    - *crew_id:* This will keep track of which crew has which fruit. For those fruit without crews, this will be NULL. The data type will be the id of a crew.

Entity-Relationship Diagram



Schema



Green line is colored because I was unable to find a way to diagram this schema without line overlap. Color differentiates the line.