INST733-IM01: Database Design-Spring 2023

Project Final Report

Date: 05/9/2023

Team - 9: Emily in Paris Fans

Team members: Ainur Abilbayeva, Angela Tseng, Yogesh Boricha

Project Name: Emily in Paris Outfits Database

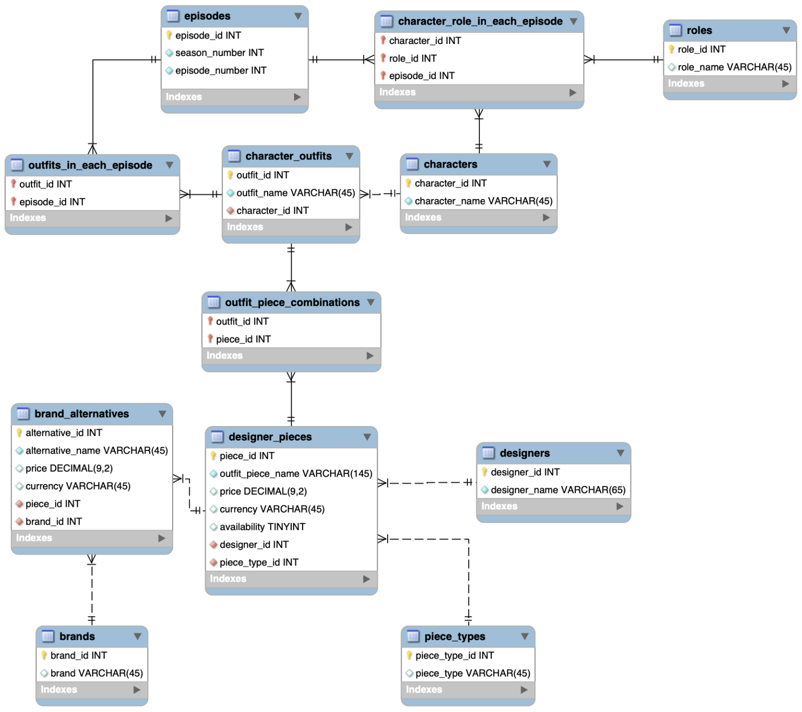
1. Introduction

The global sensation that is Emily in Paris has enamored audiences with its portrayal of a young American woman navigating life and work in the romantic city of Paris. One of the show's notable features is the unique and fashionable outfits worn by the lead character, Emily.

Emily in Paris has also gained massive popularity on social media platforms like Instagram, where fans have created dedicated fan pages for the show's outfits, particularly Emily's. A database centered on Emily's outfits can help track the popularity of different styles and brands featured in the show, which can be valuable for marketers and retailers looking to leverage the show's popularity.

This final report discusses the objectives and deliverables accomplished throughout the project. In addition, the paper also covers the methodology of designing the ERD, normalization, data population, and query generation. Afterwards, changes in design and challenges encountered will be discussed, along with a reflection on lessons learned and potential improvements to the system. By revisiting the successes and difficulties of the project, this report will summarize the development of the Emily In Paris Outfits project, ultimately providing a convenient centralized location for the outfits represented in the Netflix show.

2. Database Design and Implementation



2.1 Logical Design

Figure 1. EERD for Emily in Paris Outfits Database

The Emily In Paris Outfits database consists of 12 different tables. Of these, there are a few intermediary tables representing many to many relationships.

- **episodes:** A table summarizing all the seasons and episodes in the Emily in Paris show. In order to avoid repetition and to define the primary key, episode_id has been added to this table.
- **characters:** Table to list all the major characters in the show.
- **roles:** This table defines all the role types in a show. Since there can be only primary, secondary or general characters in a season, this table contains only three rows.
- **character_role_in_each_episode:** linking table to define roles of characters in each episode of the show. As shown in the EERD, this table summarizes the many-to-many relationship between three source tables.
- **character_outfits:** source table for the outfit names in the show.
- **outfits_in_each_episode:** linking table between character outfits and episodes, since several outfits can occur in one episode, and one outfit can be worn by a character in several episodes having this table is important.

- **designer_pieces:** table defining the outfit pieces of a certain outfit. This is the most important table in our database (containing more than 100 rows), defining all the pieces in 30 outfits described in the database.
- **outfit_piece_combinatoins:** linking table to define the outfit pieces, which belong to a certain outfit (for example, one outfit can have a bag, dress, shoe and some accessories). This table is significant to have, because one outfit can have several desginer_pieces, and pieces can be repeatedly used in several outfits.
- **designers:** pieces of the outfit have been designed by a unique designer, and this table identifies designers for each outfit piece.
- **brands:** similar to table above, this table identifies brand names and their id.
- **brand_alternatives:** since most of the designer_pieces are expensive for fans of the show, we provided alternatives tables to present identical cheaper outfit pieces from brands.
- **piece_types:** this table has been the result of the normalization, and provides the piece types summary for the outfit pieces in this database.

Assumptions

Below we provided the assumptions, which have been made to create this database:

- One episode can have many characters, and many characters can be in an episode.
- A certain character's role can change throughout the show (in each episode)
- One episode can have multiple character outfits, and multiple outfits can be found in a single episode.
- A character can have many outfits, but an outfit is only worse once by a character. Additionally, there is no requirement for a character to have a registered outfit in the database.
- An outfit can consist of multiple designer pieces, and a designer piece can be reused in multiple outfits.
- A designer can have many pieces, but a piece can only belong to one designer. A designer can have a piece registered in the database, but this is an optional relationship.
- Likewise, a piece can only be of one type of clothing or accessory, but a type can be associated with multiple pieces.

- A designer piece can have many alternatives, but it can also not have any alternatives. However, an alternative can only be associated with one piece.
- The same applies to brands. A brand can have multiple alternatives, but it can also have none. However, an alternative can only apply to one brand.

The database was designed as such to support user search of characters from certain episodes. By querying for a character or an episode, the user can access the outfits worn. From there, the user can also discover any alternatives or other information about the outfit, such as designer or brand.

2.2 Physical Database

The physical design of the database can be found through an SQL script attached. This was achieved through exporting the data on MySQL Workbench. When run, the resulting schema includes all necessary tables, imported data, and views. Attention to import order was necessary in this creation, since there are foreign keys that need to be imported in order before the next table is imported.

2.3 Sample Data

Sample data has been collected and entered manually into a spreadsheet primarily from @emilyinparisoutfits, a fanmade Instagram account dedicated to record outfits worn in the Netflix show “Emily in Paris.” Each post on the account contains information about each individual piece, the cost, character that wore it, and episode that the outfit was seen in. Information from these posts and various other supplementary fashion sources were used to populate the data in the tables.

Some different instagram accounts have been used to update tables and include more information as well. For example, the alternatives table has been taken from @themrsdrago. Since most of the clothes have been designed by worldwide famous fashion designers, Wendy Drago provided some similar, but affordable options of clothing for every Emily fan. These alternatives are included in the brand_alternatives table.

All of the source tables in the emily_in_paris_outfits database meet the minimum requirement of 20 rows, except for the roles table, which has been explained in the previous section. Since there can be only primary, secondary or general characters in a season, this table contains only three rows. As for the brands and brand_alternatives tables, there weren't many options of brands to choose the cheap options, which will be similar to the designer_pieces at the same time. These were the major reasons to have only 20 to 25 rows for the tables mentioned.

For all of the other options, the database has 12 tables, and a minimum of 20 rows in source tables, with a minimum of 50 records for linking tables meeting the requirements for sample data. In fact, for the designer_pieces we included more than 500 rows, however this required us to add information on other tables. For this reason, the information has been provided for 30 outfits, having almost 200 outfit pieces for them.

2.4 Views/Queries

View Name	Req. A	Req. B	Req. C	Req. D	Req. E
Q1	✓	✓		✓	
Q2	✓	✓		✓	
Q3	✓				
Q4	✓		✓		
Q5	✓		✓		
Q6	✓		✓		
Q7	✓	✓	✓		
Q8	✓		✓	✓	✓

Q1: View of all outfits worn by a specific character in season 2

Q2: View of all outfits worn by a specific character in S3 E2

Q3: View of all outfit pieces and their prices by designer and piece type

Q4: View of all outfits and their total cost by character

Q5: View of all brands and the number of outfits that feature their pieces

Q6: View of all designers and the number of outfits they created in the show

Q7: View of the average price of outfit pieces by designer and piece type.

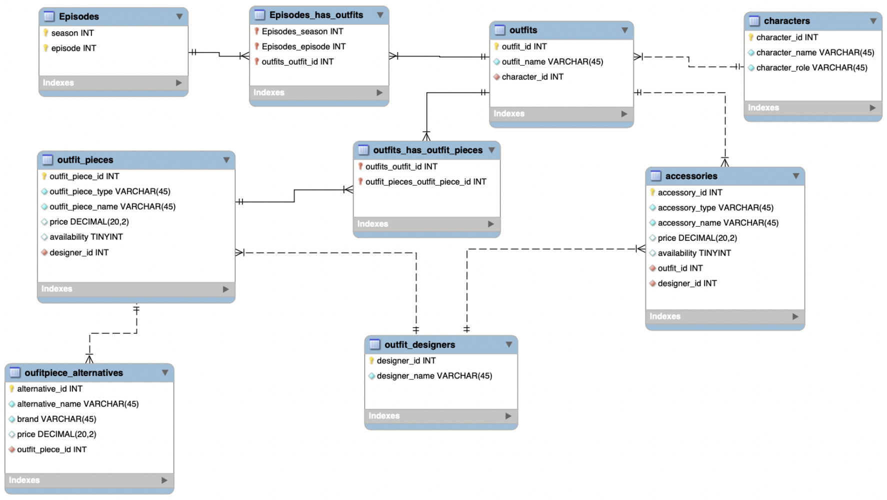
Q8: View of all outfit pieces and their prices, sorted by their popularity in terms of the number of characters who wear them. This query uses a subquery to calculate the popularity of each outfit piece, and then joins that result to the outfit_piece_combinations and designer_pieces tables to retrieve the relevant information

3. Changes from Original Design

3.1 Iteration 1

Using the feedback received from the professors, we added an accessories table to provide clarity and remove repetition. An additional episodes table was added as well. Two intermediary tables were added to represent many-to-many relationships; many outfit pieces can be used in many outfits, and many outfits can appear in many episodes. These are dependent relationships. Overall table structures and columns were reassessed and normalized. Normalization was done using the top-down approach using the ERD built and submitted with the project proposal.

Normalized Draft ERD:



3.2 Iteration 2

The second round of feedback offered plenty of changes to the ERD. Since the ERD above only has 9 tables, suggestions were made to increase the table count. A types table was added as a many to many relationship. This replaces the outfit pieces and accessories table organization that was previously sketched out. By doing so, there is more clarity and less clutter in a single table. A brand table was added to keep track of brands separately from designer pieces as well, since it overlapped with the brand_alternatives table. Another character roles table was also added to provide more context for characters in the show. A currency column was added to improve clarity of the prices and the memory allocated to the price attribute was reduced from 20 digits to 9 to avoid unnecessary space allocation and improve performance.

4. Issues Experienced During Development

4.1 ERD Normalization

A. **Issue**

One of the major challenges for creating ERD was the redundancy of the data. Outfit names were present in several tables, causing inefficiencies in updating and calling data.

B. **Solutions considered**

The obvious solution would be to normalize the tables. However, to do so, the table attributes had to be defined first.

C. **Solutions chosen**

To avoid the loss of data, the table outfit_pieces was rearranged. Various iterations were made to adjust the attributes that were deemed necessary and remove the unnecessary. The table was finally normalized once there was a consensus on what attributes to include. This eliminates the duplicate data in different tables.

4.2 Data Importation

A. **Issue**

After completing the ERD design, the team chose to use the forward engineer function in MySQL to create the schema. After successfully doing so, the Table Import Wizard was used to import data from csv files representing each table. However, there were many errors associated with this process. Certain special symbols are not supported in UTF-8 format, causing the import to stop immediately with the error "'ascii' codec can't decode byte".

B. **Solutions considered**

There were not many solutions considered with this, since the error logs were clear on what the issue was. Diagnosing the location and nature of the issue took a significant amount of time. The time spent on fixing the issues was largely from searching errors and their specific causes.

C. **Solutions chosen**

After removing these symbols and replacing them with adequate counterparts, the file could be imported. However, upon checking the logs, there were still errors such as "Incorrect value" or "Data too long" or "Out of range value". The errors lie in trying to insert mismatching data types, null values, or values too large for the allocated space. The data from the csv files had to be checked again for incorrect types, and the column attributes were readjusted to accommodate larger values.

5. Lessons Learned

The largest roadblock was successfully importing the data from the excel sheets. Going through the cells individually to check if there were symbols that would not import properly into MySQL took a significant amount of time, and even after fixing that, there was more troubleshooting to be done with errors. In this case, the data was entered manually so quality control could have been done earlier. However, even with automated data entry, data cleansing has to be performed on the data set. Due to the time consuming nature of manual data entry, it is worth considering writing a script to parse and collect the data from Instagram rather for future work.

The main lesson learned from this is understanding the importance of data uniformity. In the future, more attention should be paid to what type of data is being entered and the memory allocated to it. More time should be allocated to assessing the raw data and the different ways they can be organized in a database system.

The team also discovered that different versions of MySQL can cause different errors to occur. While one member would receive errors in running a query, another would not. Due to operating system limitations and version mismatch, there may be some inconsistencies in query execution. However, this was mitigated by testing queries on multiple computers to ensure successful execution.

6. Potential Future Work

6. 1 Extensions within the current technology and design

Future alterations to the database would include more specific outfit piece descriptions to aid in a user's search. A possible addition may include adding in designer locations, or future outfits if the show is renewed for another season. A future improvement could include organizing so that pieces are searchable by year or collection, if applicable. Another possibility is to develop a website with outfit pictures, so that it is easier to choose a certain outfit and its pieces, and compare prices.

6.2 Feasibility of alternative implementations

MySQL is inefficient in handling very large data, so if the show is to continue for more seasons, it may be more feasible to store the data on a different platform. This is when using a platform with lower cost and space requirements like Oracle would be more efficient. Another weakness of MySQL is its weak security; however, this does not pose a problem to the database at hand since the data stored is not of sensitive nature. With how fast and how scalable MySQL is, it makes sense to continue using it as the main paradigm and tool. NoSQL would not be necessary, as the data a user would pull from the database would not need to be accessed too quickly. In addition, due to the nature of the audience, using a method with no standardized language and difficulties with complex queries would likely lead to user frustration.

7. References

1. *Star, D. (2020, October 2). Watch Emily in Paris: Netflix official site. Watch Emily in Paris | Netflix Official Site. Retrieved March 7, 2023, from https://www.netflix.com/title/81037371*
2. *Emily in Paris outfits (@emilyinparisoutfits) - instagram*. (n.d.). Retrieved March 8, 2023, from https://www.instagram.com/emilyinparisoutfits/
3. *Wendy Drago | Emily in Paris outfits (@themrsdrago) - instagram*. (n.d.). Retrieved March 8, 2023, from https://www.instagram.com/themrsdrago/