Data and Application Project Phase - 4

Team 50 : Systum
2022111025 - Prakhar Singhal
2022101088 - Mohak Somani
2022101039 - Chirag Dhamija
2022115007 - Samagra Bharti
Git Repo

1 Introduction

Within the Spider-Verse Mini-World, we explore the interconnected web of Spider-People from various dimensions. This database aims to create an organized repository of information and activities related to these Spider-People, enhancing communication and coordination among them as they battle different threats across the multiverse. From Peter Parker's responsibilities as a mentor to Miles Morales, Gwen Stacy's adventures as Spider-Woman, and Mary Jane Watson's journalistic pursuits, this database will serve as a central hub for Spider-People to access vital information and manage their superhero lives.

2 Initialization

The following Python script is an interactive interface with the MySQL database. Before handling any operations, database connections have been set up with the help of **'pymysql'**, **'subprocess'**, and **'os'** libraries along with global variables for cursor, connection, and shell processing. Further, it also contains functions to print the desired tables or columns as well as specific messages in case any error occurs. Upon executing the code, a set of options is presented in the terminal for potential execution. Upon selecting a particular operation, the code is directed to a "HandleChoice" function, which subsequently redirects the code to specific functions corresponding to the chosen operation.

3 Commands

The following is a brief description of every command that is executable through the python interface on the Spiderverse Database:-

3.1 Simple SQL Query

This operation allows users to input an entire SQL query for execution. The working of this operation is explained below
The function **HandleSimpleSQL** is responsible for executing any given SQL query and print the results regarding the database.

- After accepting the argument (**'query'**), the **'try'** block tries to execute the query.
- If any error is detected, it displays an appropriate message.
- If there are no errors, the function proceeds to the loop.
- Inside the loop, **'c.fetchall()'** retrieves all the rows returned by the query and prints the result set.

3.2 Insertion

This operation enables the user to insert new datapoints in the database in various tables such as

1. Insert new Spider-People from various dimensions as they are discovered.
2. Add new villains and record their abilities
3. Create mission entries, specifying objectives, participants, and outcomes.
4. Include new dimensions and their descriptions.
5. Add equipment with types, descriptions, and owners (Spider-People).
6. Insert new organizations and side characters.

The function **HandleInsert** is responsible for inserting data to different table into the database. The working of this operation is explained below:

- The function starts with displaying the names of the available tables, for data insertion, and asks the user to enter the name of the desired table.
- The user can use the word **exit** to return back.
- The function creates the initial part of the **'INSERT'** query i.e., (**'INSERT INTO table SET '**).
- The different options of the table are controlled by the **'match'** statement.
- At last, the specific query is executed using the **'execute'** method of the cursor.

3.3 Deletion

This operation enables the user to delete some datapoints from the database in various tables when some events occur such as:

1. Eliminate defeated villains and their hideouts.
2. Delete outdated mission logs.
3. Remove unused equipment entries.
4. Delete organizations and side characters that are no longer relevant.

The function **HandleDelete** is responsible for deleting records from various tables of the database. The working of this operation is explained below:

- The function starts with displaying the names of the available tables, for data insertion, and asks the user to enter the name of the desired table.
- The user can use the word **exit** to return back.
- The function creates the initial part of the **'DELETE'** query i.e., (**'DELETE FROM table WHERE'**).
- The different options of the table are controlled by the **'match'** statement.
- At last, the specific query is executed using the **'execute'** method of the cursor.

3.4 Update

This operation enables the user to update the existing datapoints in the database in variuos tables such as:

1. Update SpiderPerson attributes such as abilities and connections.
2. Modify villain attributes and threat levels based on new information.
3. Update missions logs with the latest details and outcomes.
4. Update dimensions descriptions if necessary.
5. Edit equipment entries, including ownership and descriptions.
6. Update organization details and add new members.
7. Update side character attributes and connections with Spider-People.

The function **HandleUpdate** facilitates updating records in several tables of the database. The working of the function is as follows:

- The function starts with displaying the names of the available tables, for data updation, and asks the user to enter the name of the desired table.
- The user can use the word **exit** to return back.
- The function creates the initial part of the **'UPDATE'** query i.e., (**'UPDATE table SET'**).
- The different options of the table are controlled by the **'match'** statement.
- In each case, the function prompts the user to enter data value which they want to update. Entering NULL would result in no change in the database.
- At last, the specific query is executed using the **'execute'** method of the cursor.

3.5 Projection

The Projection operation can be employed to get the below data :

1. SpiderPersons and Associated Villains and SideCharacters.
2. SpiderPersons and their Abilities.
3. Villains and their Abilities.
4. Sidecharacter and their Abilities.
5. SpiderPersons and their associated Equipments.
6. SpiderPerson and Associated Missions.

The function **HandleProjection** handles different projection operations on the database. The working of this operation is explained below :

- The function lists all available projections, and asks the user to choose desired projection.
- The '**match**' statement manages the different projection options. Each case has its separate SELECT query as per the requirement.
- The query executes and displays appropriate result.

3.6 Aggregate

The Aggregate operation can be employed to get the below data :

1. Average Threat Level of Villians
2. Total Number of Missions (Grouped by Outcome)
3. Number of SpiderPersons and Villians in each Dimension
4. Total Equipment used by each SpiderPersons

The function **HandleAggregate** performs numerous aggregate operation available according to the database. The working of this operation is explained below :

- The function lists all available aggregate, and asks the user to choose desired aggregate option.
- The '**match**' statement manages the different aggregate options. Each case has its separate SELECT query as per the requirement.
- The query executes and displays appropriate result.

3.7 Search

The Search operation enables the user to search for the below data :

1. Search for Villains of SpiderPerson(by ID)
2. Search for Mission assigned to SpiderPerson(by ID)
3. Search for Equipment used by SpiderPerson(by ID)
4. Search for ResearchNotes written by SpiderPerson(by ID)
5. Search for Members of Organization (by ID)

The function **HandleSearch** tries to execute various search operations possible on the database. The working of Search operation is explained below :

- The function lists all available aggregate, and asks the user to choose desired aggregate option.
- The '**match**' statement manages the different search options.
- The user has to enter additional information and for each case different SELECT query is constructed.
- The query executes and displays appropriate result.

3.8 Analytical

The Analytical operation can be employed to get the below data :

1. SpiderPerson Efficiency Report
2. Villian Opposition Network report
3. Equipment Usage Report

The function **HandleAnalytical** helps us to carry out several analytical operations on the database. The working of this operation is explained below :

- The function displays all available analytical choices, and asks the user to choose desired analytical option.
- The '**match**' statement manages the different aggregate options. Each case has its separate SELECT query as per the requirement.
- The query executes and displays appropriate result.

4 Assumptions

- Imaginary mask names have been included to populate the database; alternatively, their values can be set to NULL.
- When inputting a string to be inserted into the database and the string contains an apostrophe, it is recommended to replace the single apostrophe with two consecutive apostrophes at the same position during the input process. This practice helps formalize data entry and ensures accurate representation of the apostrophe in the database.

- The identifiers for certain tables, such as SpiderIdentifier and VillainIdentifier, serve as primary keys configured for auto-incrementation. If a new entry is inserted and subsequently deleted, the primary key continues to increase. For instance, if the last inserted entry had a value of 6, a newly inserted entry would be assigned the value 7. Similarly, if an entry is deleted and a new one is inserted, the subsequent entry will be assigned the next incremented value, i.e. 8.
- The update command is restricted to entity relations (tables) like SpiderPerson, Villain etc. to avoid potential update anomalies that could arise in relationship tables. This approach is adopted due to concerns about the complexity and risks associated with updating records directly in relationship tables. Consequently, the methodology involves deletion and subsequent insertion for the relationship tables to ensure data integrity and mitigate the possibility of anomalies.
- A new table named "images" has been introduced which is designed to map images to its corresponding image IDs. This table is dynamic and can be populated during runtime. The LogoId will be initialised with NULL, which can be later modified after execution of the program.
- The primary keys in the entity table typically allow users the option to either manually set a personalized primary key value or automatically generate an incremented value.