

Final Project Report: rDB (Relational Database)

Unmay Thagan Pawar

upawar@usc.edu

8420182018

Group No. 61

DSCI-551: Foundations of Data Management

Instructor: Wensheng Wu

December 6, 2023

Table of Contents

1. Introduction	3
2. Planned Implementation	4
3. Architecture Design	5
4. Implementation	7
- Functionalities	7
- Tech Stack	9
- Screenshots	10
5. Learning Outcomes	12
6. Conclusion	13
7. Future Scope	14
8. Appendix	15

1. Introduction

My project is dedicated to the development of a relational database, a structured repository designed to manage and correlate data seamlessly. Through this project, I aim to address the complexities associated with unstructured or loosely connected datasets by implementing a relational model that fosters logical relationships between different entities.

The key advantages of a relational database lie in its ability to organize data in a structured manner, reduce redundancy, and enable sophisticated querying. Through careful design and normalization, my project seeks to optimize data integrity, minimize data duplication, and streamline the process of extracting valuable insights. The relational model allows for the establishment of connections between entities, facilitating complex queries and supporting the seamless integration of diverse datasets.

In this project, my focus will be on creating a robust and scalable relational database that can be applied across various domains. I will delve into the intricacies of database design, exploring concepts such as entity-relationship modeling, normalization, and schema definition. By adhering to best practices in database management, my goal is to deliver a versatile and user-friendly solution that can serve as a foundation for future data-related endeavors.

The impact of a well-designed relational database extends beyond mere data storage; it is a catalyst for informed decision-making, efficient data retrieval, and improved overall system performance. Through this project, I anticipate contributing to the broader discourse on effective data management and providing a valuable resource for individuals and organizations seeking a robust solution to their data storage and retrieval needs.

2. Planned Implementation

My planned implementation for the creation of a relational database involves a meticulous approach to organizing and storing data, with a focus on simplicity, efficiency, and flexibility. The key aspects of my implementation strategy are outlined below:

- I propose to organize our databases by storing them in directories. This directory-based structure provides a clear and intuitive way to manage multiple databases, making it easier to locate and maintain the relevant data. Each directory will represent a distinct database, and within each database directory, the individual tables and associated files will be logically arranged.
- To store the data within the relational database, I have chosen the widely adopted CSV (Comma-Separated Values) file format. Each table in the database will be represented by a corresponding CSV file, where rows represent individual records, and columns represent attributes. This format is both human-readable and compatible with a multitude of data processing tools, ensuring ease of use and interoperability.
- Embracing the principles of the relational model, my database design will involve the creation of tables that establish logical relationships between different entities. Each table will represent a specific entity type (e.g., movies, actors, directors), and the relationships between tables will be defined through keys. This relational approach not only enhances data integrity but also enables efficient querying and analysis.
- Final implementation will include a query language designed to interact with the relational database. The query language will rely on keywords and follow a syntax inspired by standard SQL (Structured Query Language) conventions. This choice ensures that users, even those unfamiliar with complex database languages, can easily formulate queries using intuitive and recognizable keywords. This approach aligns with the commitment to creating a user-friendly database system.

3. Architecture Design

The flow diagram of whole project is shown below in Figure 1.

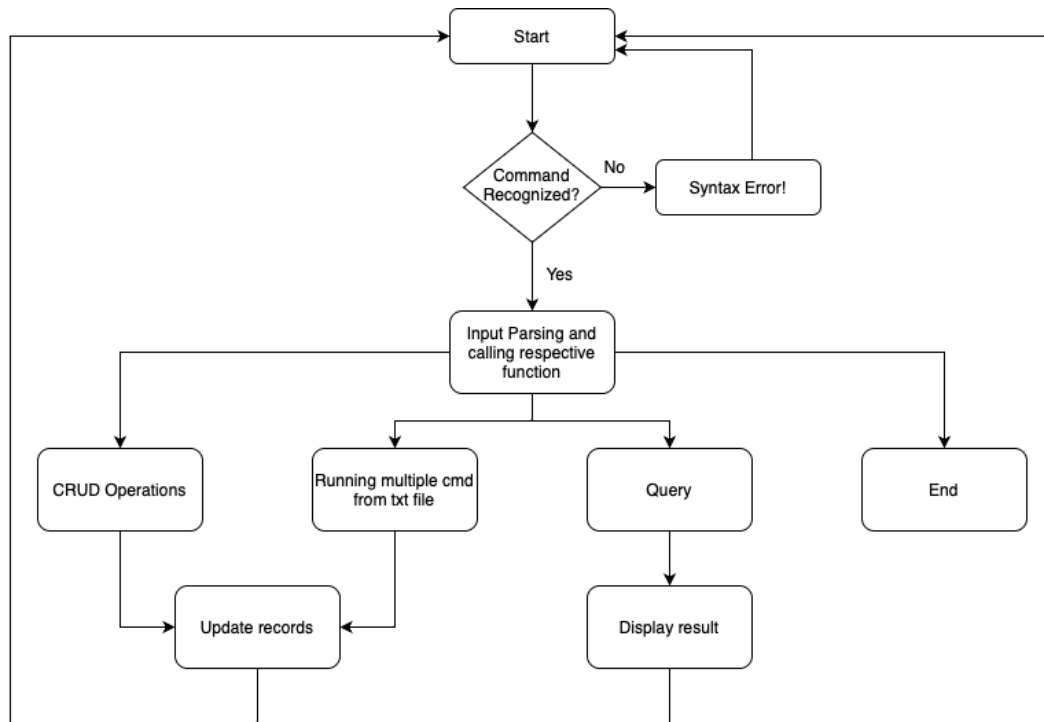


Fig. 1 Flow Diagram

- **Start:**

The process begins at start.

- **Command Recognized:**

The system checks if the entered command is recognized. If the command is not recognized then system reports “Syntax Error!” And the flow returns to start. If the command is recognized then the system proceeds to input parsing.

- **Input Parsing:**

The system parses the input to identify the type of operation requested.

- **CRUD Operation:**

If the operation is a CRUD (Create, Read, Update, Delete) operation, the system executes the corresponding function. After completing the CRUD operation, the flow returns to the start to await the next command.

- **Run Multiple Commands from Text File:**

If the operation is to run multiple commands from a text file, the system executes the commands sequentially. After updating records, the flow returns to the start.

- **Query:**

If the operation is a query, the system executes the query and retrieves the results. The results are displayed to the user. After displaying the results, the flow returns to the start.

- **Stop:**

If the operation is to stop, the system ends the process and the process concludes.

4. Implementation

• Functionalities:

The implementation of my relational database system revolves around a user-friendly interface that relies on intuitive keywords to execute various operations. Each keyword is designed to perform a specific task, ensuring a straightforward and accessible user experience. The primary focus is on the "query" keyword, which combines multiple functionalities to enhance flexibility and power in data retrieval.

The system recognizes a set of keywords such as "make", "shift", "put", "remove", "modify", "print" and others. Each keyword is associated with a specific operation, making it easier for users to convey their intentions in a natural and intuitive manner. For instance:

- "make" is used for creating new database or table.
- "shift" facilitates which database to work with.
- "put" is employed for inserting data.
- "remove" deletes records.
- "modify" allows users to update existing data.
- "print" displays information.
- "import" takes multiple commands at once from a text file and runs them sequentially.

The keywords perform actions as suggested by their names, simplifying the user's interaction with the system. Users can employ these keywords to execute CRUD operations seamlessly.

While all keywords serve distinct purposes, the "query" keyword takes center stage in our implementation. It is designed to combine multiple functionalities, allowing users to tailor their requests for data retrieval with precision and efficiency. The integration of diverse functionalities

under the "query" keyword empowers users to extract specific information from the database according to their unique requirements.

The query functionality is designed with a user-centric approach, allowing users to express complex queries in a straightforward manner. The input is segregated into three crucial parts:

- Attributes to display.
- Conditions.
- Operations.

- **Attributes to display:**

Users have the flexibility to choose the specific information they want to see in the query results. Whether it's movie titles, actor names, release dates, or any other attribute, the system accommodates diverse user preferences for data display.

- **Conditions:**

The conditions section allows users to set criteria for the data they want to retrieve. Conditions can include filtering based on specific values, ranges, or other logical expressions. This feature enhances the precision and relevance of the query results. This is an optional parameter.

- **Operations:**

Users can specify additional operations to be performed on the displayed data. This can include sorting, grouping, or aggregating data to meet specific analytical needs. The operations section adds a layer of flexibility and customization to the query functionality. This is an optional parameter.

This segmentation of input ensures that users have granular control over their queries, enabling them to extract meaningful insights from the relational database with ease. The user-friendly interface, coupled with the power of the "query" keyword, positions our implementation as a versatile and efficient tool for data exploration and analysis.

- **Example Query:**

To illustrate the power of the query functionality and its segmentation, consider the following example:

query (type, count.titles), (order= -1, first= 5), (group= type).

Here, attributes to display are type and count of titles associated with them which is part of aggregation. Conditions provided to the system are order the data in descending order and display the first 5 results only. And finally to achieve this we need to perform grouping of the data by the attribute type, so it is provided in the operations section.

- **Tech Stack:**

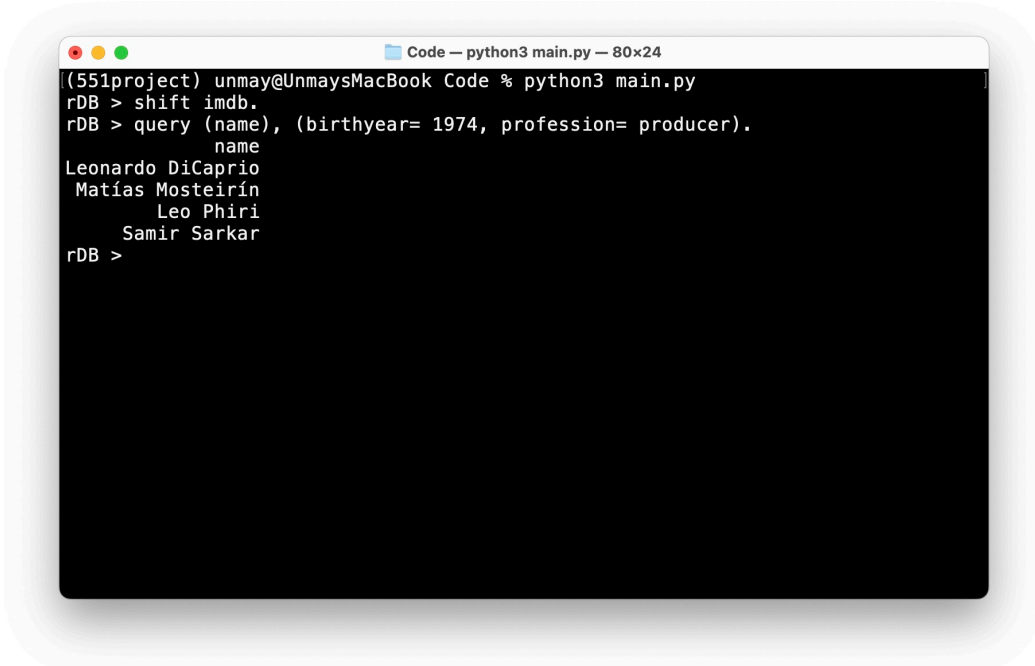
- **Programming Language:** Python.
- **Required Libraries:** pandas.

The entire technology stack for the relational database implementation is exclusively built on Python, emphasizing simplicity, versatility, and ease of use. Python serves as the primary programming language, providing a cohesive and efficient environment for developing the system. While the relational database functionalities are implemented natively in Python, Pandas is utilized primarily for the purpose of displaying query results. This streamlined tech stack underscores the commitment to creating a user-friendly, Python-centric solution for relational database management.

- **Screenshots:**

Here are few examples to show implementation.

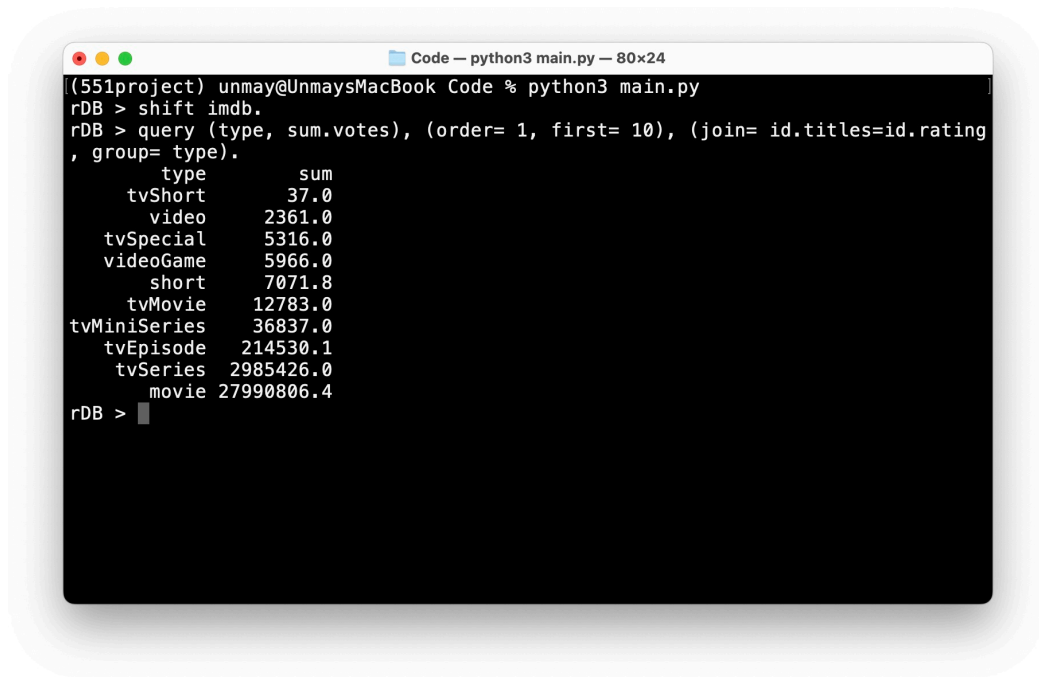
Figure 2 shows results of the simple query executed to find out names of the producers who were born in the year 1974.

A screenshot of a terminal window titled "Code — python3 main.py — 80x24". The terminal shows a Python shell session where a variable 'imdb' is assigned to 'shift imdb.'. Then, a query is executed: 'query (name), (birthyear= 1974, profession= producer)'. The output lists four names: Leonardo DiCaprio, Matías Mosteirín, Leo Phiri, and Samir Sarkar. The prompt 'rDB >' is visible at the end of the output.

```
(551project) unmay@UnmaysMacBook Code % python3 main.py
rDB > shift imdb.
rDB > query (name), (birthyear= 1974, profession= producer).
      name
Leonardo DiCaprio
Matías Mosteirín
      Leo Phiri
      Samir Sarkar
rDB >
```

Figure 2

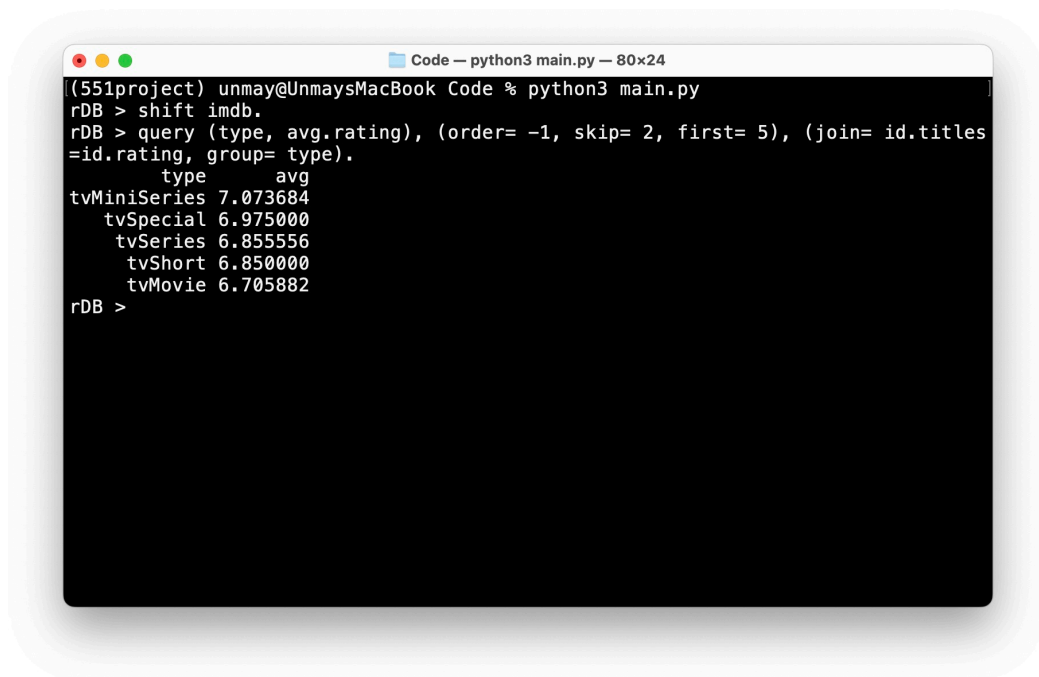
Figure 3 displays execution of query which involves grouping operation performed on the attribute type.



```
Code — python3 main.py — 80x24
((551project) unmays@UnmaysMacBook Code % python3 main.py
rDB > shift imdb.
rDB > query (type, sum.votes), (order= 1, first= 10), (join= id.titles=id.rating
, group= type).
  type      sum
tvShort      37.0
video    2361.0
tvSpecial   5316.0
videoGame  5966.0
short      7071.8
tvMovie   12783.0
tvMiniSeries 36837.0
tvEpisode 214530.1
tvSeries  2985426.0
movie  27990806.4
rDB >
```

Figure 3

Figure 4 goes on to show combination of multiple operations and various conditions applied on the records.



```
Code — python3 main.py — 80x24
((551project) unmays@UnmaysMacBook Code % python3 main.py
rDB > shift imdb.
rDB > query (type, avg.rating), (order= -1, skip= 2, first= 5), (join= id.titles
=id.rating, group= type).
  type      avg
tvMiniSeries 7.073684
tvSpecial  6.975000
tvSeries   6.855556
tvShort    6.850000
tvMovie    6.705882
rDB >
```

Figure 4

5. Learning Outcomes

Through this project, I gained hands-on experience in designing and implementing a relational database system. This encompassed creating tables, establishing relationships, and ensuring data integrity, contributing to a solid understanding of fundamental database principles.

The exclusive reliance on Python as the primary programming language enhanced my proficiency in Python development. Leveraging Python for both backend database operations and user interaction reinforced the coding skills and familiarity with Python libraries.

Emphasizing intuitive keywords and a user-friendly query interface honed my ability to design systems with end-users in mind. The segmentation of the query functionality and the incorporation of interactive features aimed at making the system accessible to users with varying levels of expertise.

Implementing a flexible query system presented challenges in parsing user inputs effectively. Ensuring the accurate segregation of queries into display elements, conditions, and operations required careful consideration to handle a variety of user inputs.

Designing a robust error-handling system and providing meaningful feedback to users posed challenges. Striking a balance between informative error messages and maintaining a user-friendly interface required iterative improvements.

Implementing interactive query building features demanded thoughtful design to guide users effectively. Balancing simplicity with the provision of advanced querying capabilities posed challenges in creating a user-friendly yet powerful system.

Despite these challenges, the learning outcomes reflect a successful journey in developing a relational database system using Python. The project not only enhanced technical skills but also emphasized the importance of user-centric design and problem-solving in the development process.

6. Conclusion

In the culmination of relational database project, I have successfully crafted a Python-based system that marries the principles of simplicity, functionality, and user accessibility. By relying exclusively on Python and leveraging the Pandas library for display purposes, I have created a streamlined tech stack that embodies versatility and ease of use. The implementation, driven by intuitive keywords, empowers users to interact with the relational database effortlessly, performing operations such as creation, modification, and querying with natural language commands.

The focal point of my project, the "query" keyword, emerges as a powerful tool, combining multiple functionalities to enhance the precision and flexibility of data retrieval. The segmentation of the query input into attributes to display, conditions, and operations underscores the commitment to providing users with a granular and intuitive querying experience.

Throughout this endeavor, I navigated challenges related to query parsing complexity, error handling. These challenges, while formidable, served as valuable learning opportunities, refining my problem-solving skills and reinforcing the importance of a user-centric design approach.

In essence, this project is not merely a culmination of code and functionalities; it represents a journey of learning, exploration, and growth. The relational database system I have created is a testament to the potential synergy between technical prowess and user-centric design. As I conclude this project, I look forward to its continued evolution, anticipating its impact on users' experiences in exploring and interacting with relational data in an intuitive and efficient manner.

7. Future Scope

The completion of my relational database project marks the foundation for a promising future, and there are several avenues for expansion and improvement. The following outlines potential areas of focus for future development:

- Expand the capabilities of the query language to support more advanced functionalities, such as nested queries, additional aggregate functions, and further options for data manipulation. This will cater to the needs of users with more complex analytical requirements.
- Invest in performance optimization to ensure efficient query execution, particularly as the database grows. Techniques like indexing, query optimization, and caching mechanisms can be explored to enhance system responsiveness.
- Implement robust security measures to protect the integrity and confidentiality of the data. This includes user authentication, authorization mechanisms, and encryption protocols to secure data both during storage and transmission.
- Incorporate mechanisms for managing concurrent access to the database to prevent data inconsistencies. Implementing strategies such as locking mechanisms and transaction control will enhance the system's suitability for multi-user environments.
- Engage in an iterative feedback loop with users to identify pain points, gather suggestions, and implement usability improvements. Regular updates based on user feedback will contribute to the ongoing refinement of the system.

8. Appendix

- **Code Link**