DIGITAL CONTENT RETRIEVAL - MOD. B

Search Facility for Local File System

Yasamin Hosseinzadeh Sani

Department of Computer Engineering - Data Science

University of Pavia, Italy

Contact: yasamin.hosseinzadehsa01@universitadipavia.it

June 15, 2024

Abstract

This project focuses on developing a robust search facility for a subtree within a local file system. The key objectives include creating a system that searches for a specified string within file names and their contents, ensuring the subtree meets specific depth and content requirements, and providing detailed search results including the file path, type, and occurrence count of the search term. The implementation leverages a MySQL database for indexing and storing file metadata, facilitating efficient search operations.

1 Introduction

The purpose of this project is to design and implement a search engine that operates on a specified subtree of a local file system, adhering to predefined structural and content requirements. The subtree must have a depth of at least six levels, with a specific subtree named ”DCRB” containing at least 50 files distributed across at least four levels. The search engine is designed to identify matches based on file names and content, providing detailed results including file paths, types, and occurrence counts of the search term.

2 System Architecture

The system architecture involves the interaction between a local file system, a Python-based search application, and a MySQL database. The following components are utilized:

- **Hardware Setup**: The development was conducted on a local machine.
- **Software Components**: The main components include Visual Studio Code (VS Code) for development, MySQL Workbench for database management, and MySQL server for data storage.
- **Interaction**: The Python application connects to the MySQL database using MySQL connectors. The database is managed through MySQL Workbench, and command-line operations are performed via the command prompt.

2.1 Commands Used for MySQL Connection

To connect the Python application to the MySQL database, the following command is used in the command prompt:

```
mysql -u root -p
```

This command prompts for the MySQL root user password and allows access to the MySQL shell where database operations can be performed.

3 Implementation Details

The project implementation involves several steps:

- **Programming Languages and Frameworks**: The primary language used for the development is Python. MySQL is used for the database, and MySQL Workbench is used for database management.
- **Development Methodologies**: The project followed an iterative development approach, allowing for continuous testing and integration of new features.
- **Tools and Libraries**: Key libraries used include `mysql-connector-python` for database connectivity and `os` and `sys` modules for file system operations.

4 Main Functions

4.1 check_subtree_depth(root_dir, min_depth)

This function traverses the given directory to verify if it meets the minimum depth requirement. It iterates through the directory structure, counting the levels and returning `True` if the depth condition is satisfied.

4.2 verify_dcrb_subtree(root_dir)

This function ensures the presence of the "DCRB" subtree within the specified root directory and verifies that it meets the required depth of at least four levels. It returns the path to the "DCRB" subtree if the conditions are met.

4.3 verify_dcrb_contents(dcrb_path)

This function checks the content requirements of the "DCRB" subtree, confirming that it contains at least 50 files spread across at least four levels. It returns `True` if both conditions are satisfied.

4.4 connect_to_database()

This function establishes a connection to the MySQL database where file metadata and search results are stored. It handles connection errors and prints a success message upon a successful connection.

4.5 create_files_table(connection)

This function creates the necessary tables in the database to store file metadata and search results. It ensures the presence of indexes on key columns to optimize search performance.

4.6 insert_files_into_table(directory, connection)

This function populates the database with metadata from the files in the specified directory. It reads each file's content if applicable, and stores the file name, path, type, size, and content in the database.

4.7 count_occurrences(file_path, search_term)

This function reads the content of a given file and counts the occurrences of the specified search term. It handles potential file reading errors gracefully.

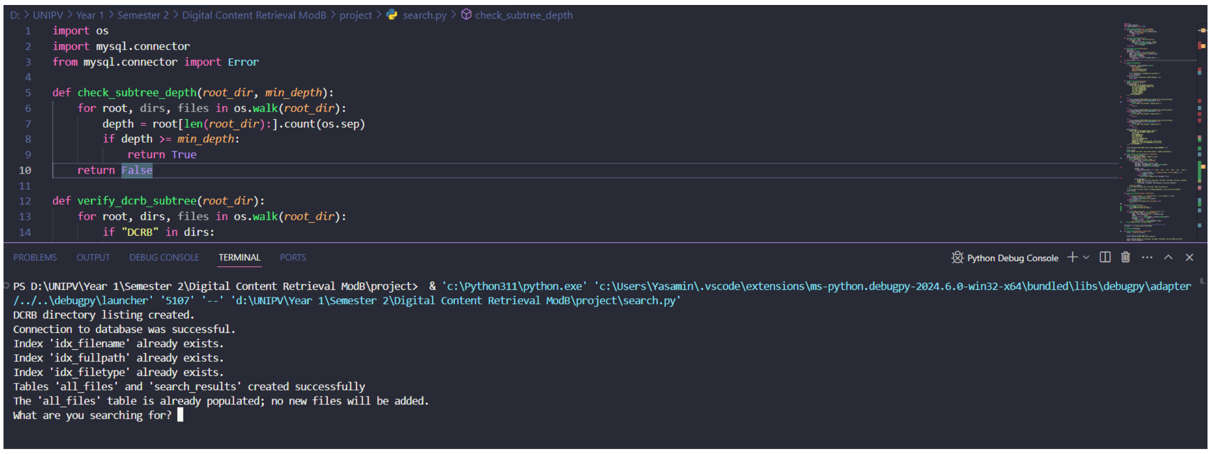
4.8 list_directory(root_dir)

This function generates a listing of the directory structure starting from the root directory and writes it to a file named `DCRB_listing.txt`. It helps in verifying the structure and contents of the "DCRB" subtree.

4.9 search_files(connection, search_term)

This function performs the actual search operation. It queries the database for files matching the search term in their names or contents, counts occurrences, and stores the results in the `search_results` table. It prints matching results and handles directories separately.

4.10 Screenshot of VS Code Asking for Search Term


D: > UNIPV > Year 1 > Semester 2 > Digital Content Retrieval ModB > project > search.py > check_subtree_depth

```python
import os
import mysql.connector
from mysql.connector import Error

def check_subtree_depth(root_dir, min_depth):
    for root, dirs, files in os.walk(root_dir):
        depth = root[len(root_dir):].count(os.sep)
        if depth >= min_depth:
            return True
    return False

def verify_dcrb_subtree(root_dir):
    for root, dirs, files in os.walk(root_dir):
        if "DCRB" in dirs:
```

```
PS D:\UNIPV\Year 1\Semester 2\Digital Content Retrieval ModB\project>  & 'c:\Python311\python.exe' 'c:\Users\Yasamin\.vscode\extensions\ms-python.debugpy-2024.6.0-win32-x64\bundled\libs\debugpy\adapter
/../..\debugpy\launcher' '5107' '--' 'd:\UNIPV\Year 1\Semester 2\Digital Content Retrieval ModB\project\search.py'
DCRB directory listing created.
Connection to database was successful.
Index 'idx_filename' already exists.
Index 'idx_fullpath' already exists.
Index 'idx_filetype' already exists.
Tables 'all_files' and 'search_results' created successfully
The 'all_files' table is already populated; no new files will be added.
What are you searching for? █
```

Figure 1: VS Code Search Prompt

5 Database Schema

The database schema includes two main tables: `all_files` and `search_results`. The `all_files` table stores metadata about the files, including file paths, types, sizes, and contents. The `search_results` table stores the results of search queries, including file paths, types, occurrence counts, and sizes.

6 Search Algorithms

The search engine uses a combination of SQL queries and string matching algorithms to search file names and contents. The process involves first running the code to store all files and the contents of readable files in the database (referred to as ”DCRB”). When a search is performed, it queries this database instead of searching through the exact locations of the files. This approach enhances search efficiency and speed.

7 Testing and Validation

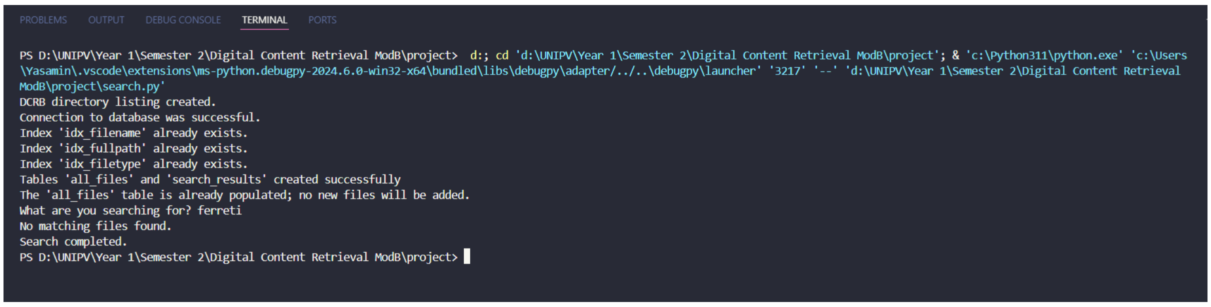
The system was thoroughly tested to ensure its robustness and accuracy. Test cases included:

- Searching for non-existing strings.
- Searching for strings that matched file names but not contents.
- Searching for strings that matched both file names and contents.

The test results confirmed that the system correctly identified and reported search matches, with accurate occurrence counts and file paths.

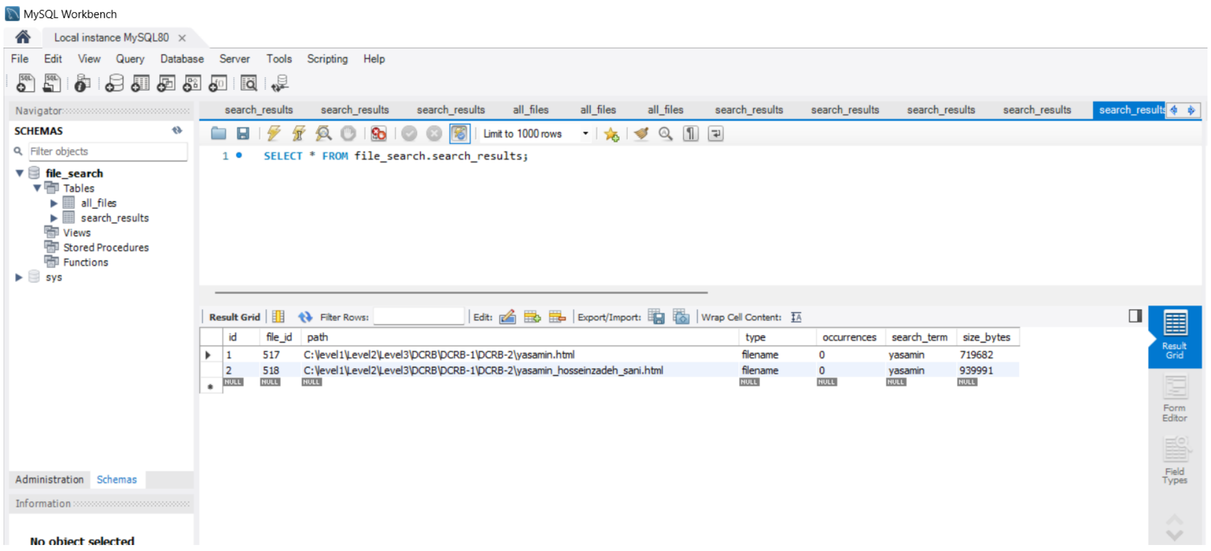
8 Results and Discussion

The project includes detailed search functionalities demonstrated through various scenarios. The following screenshots showcase the search results for different test cases:



```
PS D:\UNIPV\Year 1\Semester 2\Digital Content Retrieval ModB\project>  d:; cd 'd:\UNIPV\Year 1\Semester 2\Digital Content Retrieval ModB\project'; & 'c:\Python311\python.exe' 'c:\Users
\Yasamin\.vscode\extensions\ms-python.debugpy-2024.6.0-win32-x64\bundled\libs\debugpy\adapter/../..\debugpy\launcher' '3217' '--' 'd:\UNIPV\Year 1\Semester 2\Digital Content Retrieval
ModB\project\search.py'
DCRB directory listing created.
Connection to database was successful.
Index 'idx_filename' already exists.
Index 'idx_fullpath' already exists.
Index 'idx_filetype' already exists.
Tables 'all_files' and 'search_results' created successfully
The 'all_files' table is already populated; no new files will be added.
What are you searching for? ferreti
No matching files found.
Search completed.
PS D:\UNIPV\Year 1\Semester 2\Digital Content Retrieval ModB\project> 
```

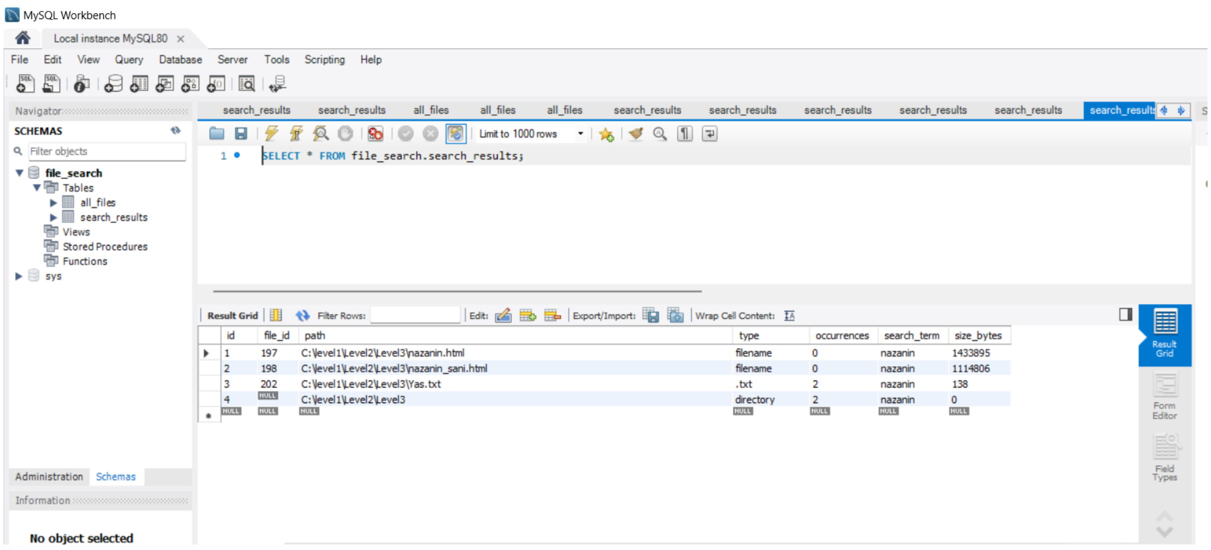
Figure 2: Search for a Non-existing String



```sql
SELECT * FROM file_search.search_results;
```

id	file_id	path	type	occurrences	search_term	size_bytes
1	517	C:\level1\Level2\Level3\DCRB\DCRB-1\DCRB-2\yasamin.html	filename	0	yasamin	719682
2	518	C:\level1\Level2\Level3\DCRB\DCRB-1\DCRB-2\yasamin_hosseinzadeh_sani.html	filename	0	yasamin	939991
NULL	NULL	NULL	NULL	NULL	NULL	NULL

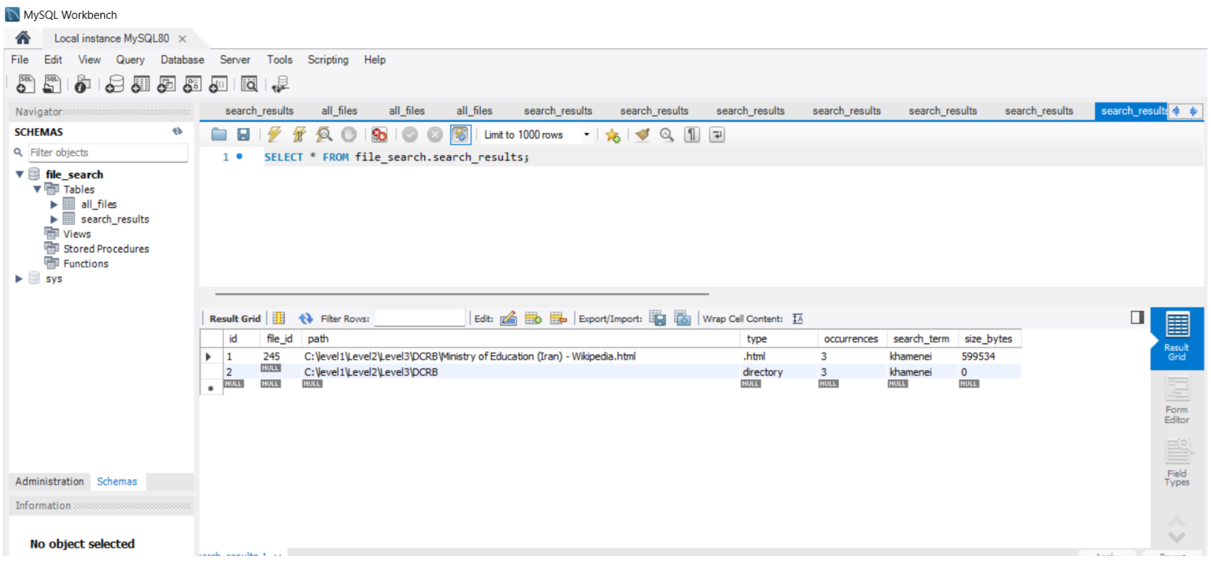
Figure 3: Search for a String Matching at Least Two File Names



```sql
SELECT * FROM file_search.search_results;
```

id	file_id	path	type	occurrences	search_term	size_bytes
1	197	C:\level1\Level2\Level3\nazanin.html	filename	0	nazanin	1433895
2	198	C:\level1\Level2\Level3\nazanin_sani.html	filename	0	nazanin	1114806
3	202	C:\level1\Level2\Level3\Yas.txt	.txt	2	nazanin	138
4	NULL	C:\level1\Level2\Level3	directory	2	nazanin	0
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figure 4: Search for a String Matching at Least One File Name and Found in Contents



```
SELECT * FROM file_search.search_results;
```

id	file_id	path	type	occurrences	search_term	size_bytes
1	245	C:\level1\Level2\Level3\DCRB\Ministry of Education (Iran) - Wikipedia.html	.html	3	khamenei	599534
2	NULL	C:\level1\Level2\Level3\DCRB	directory	3	khamenei	0
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figure 5: Search for a String Not Matching File Names but Found in Contents

9 Challenges and Solutions

During the development, several challenges were encountered:

- **Handling Large Directories**: Managing and searching through large directories required optimizing the search algorithms and database queries.
  - **Solution**: Implemented indexing and efficient string matching algorithms.
- **Database Connectivity Issues**: Ensuring a stable connection between the Python application and MySQL database.
  - **Solution**: Used robust error handling and retry mechanisms.

10 Future Work

Future improvements and potential extensions include:

- **User Interface**: Developing a graphical user interface for easier interaction with the search engine.
- **Advanced Search Features**: Adding support for regular expressions and more complex search queries.
- **Performance Optimization**: Further optimizing the search algorithms for better performance with very large datasets.

11 Conclusion

The search facility developed in this project meets the specified requirements, providing a comprehensive solution for searching within a defined subtree of a local file system. By leveraging a MySQL database for efficient indexing and storage of file metadata, the system ensures quick and accurate search results. The implementation includes thorough checks for directory structure and content, ensuring compliance with the project specifications. Overall, this project demonstrates the effective use of database management and file handling techniques to build a reliable search engine.

12 References

- MySQL Documentation: `https://dev.mysql.com/doc/`
- Python MySQL Connector Documentation: `https://dev.mysql.com/doc/connector-python/en/`

13 Appendix

13.1 Directory Listing

A comprehensive listing of the ”DCRB” directory and its subdirectories is generated and saved in a file named `DCRB_listing.txt`. This file provides a detailed view of the directory structure and contents, ensuring compliance with the project requirements.

13.2 SQL File Explanation

The SQL script included in this project is used to create and populate the database required for the search functionality. The script creates a database named `file_search` and two tables: `all_files` and `search_results`. The `all_files` table stores metadata about each file, including its name, path, type, size, and content. The `search_results` table stores the results of search queries, including the file ID, path, type, occurrence count of the search term, and the search term itself.