Querying Structured Data in Dataset Search Engines

Sri Vikas Prathanapu,[1] **Kaushik Tummalapalli** [2]

[1,2] Department of ECE, New York University
[1] sp6904@nyu.edu, [2] kt2651@nyu.edu
Repo: https://github.com/srivikas777/CSGY6513-Project.git

Abstract

This project aims to improve the functionality of dataset search engines by allowing users to query the contents of discovered datasets. While current search engines enable users to query metadata associated with datasets, this may not provide enough information to determine if a dataset is suitable for their needs. To address this limitation, the project proposes developing infrastructure to support queries over datasets stored in a file system or scalable object storage backends using MinIO. The project involves modifying the codebase of Auctus, a dataset search engine, to convert datasets to Parquet files, store them in an open-source object store, and modify the user interface to enable users to execute SQL queries over the Parquet files stored in the S3 object store using the DuckDB Python wrapper. Users can then execute SQL queries over the Parquet files using the modified Auctus user interface, which queries the DuckDB database engine. The project utilizes PyArrow, Pandas, Flask, and DuckDB Python libraries to achieve its objectives.

Introduction

The ability to search and discover datasets is crucial for researchers, data scientists, and businesses seeking to extract valuable insights from data. However, discovering datasets is only the first step, as users need to query the contents of the datasets to determine if they are suitable for their needs. Currently, dataset search engines allow users to issue queries over metadata associated with datasets. While this provides some useful information, it may not be sufficient for determining the suitability of a given dataset. To address this limitation, this project proposes developing infrastructure that supports queries over datasets stored in a file system or scalable object storage backends using MinIO. The proposed modifications involve converting datasets to Parquet files, storing them in an open-source object store, and modifying the user interface to allow users to query the Parquet files using the DuckDB Python wrapper. This approach offers several advantages, including improved query performance, reduced storage costs, and increased flexibility in data exploration. The project will leverage PyArrow, Pandas, Flask, and DuckDB Python libraries to achieve its objectives. The success of this project will enhance the capabilities of dataset search engines, enabling users to make informed decisions about the suitability of discovered datasets for their needs.

Methodology

The proposed methodology involves several steps to enable users to execute SQL queries over the contents of discovered datasets:

Dataset conversion: The first step involves converting discovered datasets to Parquet files to improve query performance and reduce storage costs. To achieve this, the project will utilize PyArrow and Pandas libraries to read the datasets in CSV format, create a PyArrow table from the dataframe, and write the table to a Parquet file.

Object storage: The Parquet files will be stored in an open-source object store using the Minio Python. Minio is a high-performance, distributed object storage system that is compatible with MinIO object store. The Minio server will be assumed to be running on ”localhost:9000”, and the access key and secret key will be hardcoded.

User interface modification: The modifications will involve modifying the user interface to allow users to query the stored Parquet files using the DuckDB database engine.

Query execution: The modified Auctus codebase will query the Parquet files using the DuckDB Python wrapper. DuckDB is a SQL database engine that is designed to work with flat files, including Parquet files. The codebase will utilize the DuckDB Python library to execute SQL queries over the Parquet files stored in the S3 object store.

Performance evaluation: The performance of the modified Auctus codebase will be evaluated by measuring the query response time and comparing it to the response time of executing the same query on the CSV version of the dataset.

The proposed methodology will leverage PyArrow, Pandas, Flask, DuckDB Python libraries, and Minio to achieve its objectives. The success of this methodology will enable users to execute SQL queries over the contents of discovered datasets, providing them with the ability to make informed decisions about the suitability of datasets for their needs.

Figure 1: CSV+Pandas Dataframes time taken: 230 ms

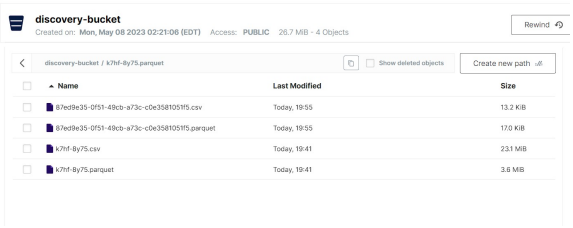

Figure 2: MinIO Files inside a discovery-bucket

Architecture:

The proposed architecture for enabling users to execute SQL queries over the contents of discovered datasets involves several components, including MinIO and DuckDB.

MinIO is an open-source object storage system that provides a scalable, high-performance platform for storing and accessing large amounts of data. In the proposed architecture, MinIO is used to store Parquet files generated by the dataset conversion module. The MinIO is used to interact with the MinIO server, allowing Parquet files to be uploaded and downloaded from the object store.

DuckDB is a high-performance SQL database engine that is designed to work with flat files, including Parquet files. In the proposed architecture, DuckDB is used to execute SQL queries over the Parquet files stored in the MinIO object store. The DuckDB Python wrapper is used to interact with the DuckDB engine, allowing SQL queries to be executed and results to be returned.

The data fetching and processing in the proposed architecture involves several steps. First, datasets are discovered using the Auctus dataset search engine. Once a dataset is discovered, it is downloaded in CSV format using the Auctus API. The dataset conversion module then converts the CSV file to a Parquet file, which is uploaded to the MinIO object store using the MinIO. The user interface modification module allows users to execute SQL queries over the Parquet files stored in the MinIO object store using the DuckDB Python wrapper. The query execution module queries the DuckDB engine, which reads the Parquet file from the MinIO object store and executes the SQL query. The results of the SQL query are returned to the user via the Auctus user interface.

In summary, the proposed architecture for enabling users to execute SQL queries over the contents of discovered datasets involves data fetching and processing using MinIO and DuckDB. MinIO is used to store Parquet files generated by the dataset conversion module, while DuckDB is used to execute SQL queries over the Parquet files stored in the MinIO store. The architecture leverages the MinIO and DuckDB Python wrapper to interact with the MinIO object store and execute SQL queries, respectively.

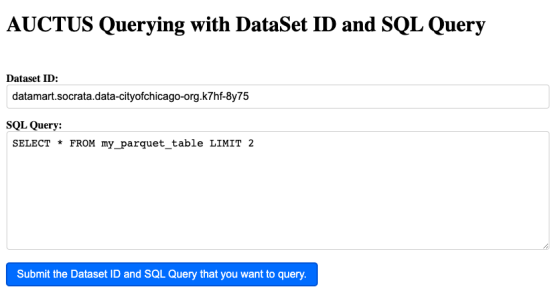

Figure 3: AUCTUS Querying Interface with an Option to add Dataset ID and SQL Query

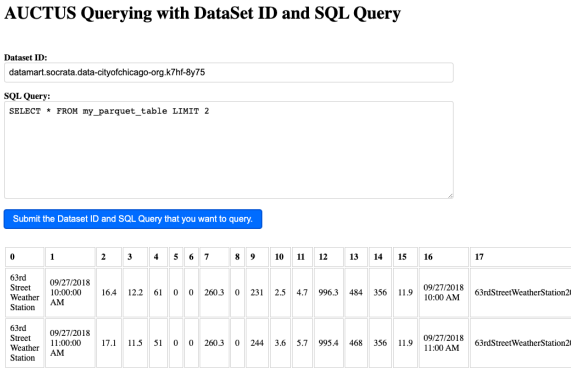

Figure 4: Results from the AUCTUS Querying Interface

Related Work

In this work, we compared the performance of Parquet and CSV file formats using two popular libraries, Pandas and DuckDB. Our experiments showed that Pandas took 0.29 seconds to read the Parquet file, while it took 0.618 seconds to read the CSV file. On the other hand, DuckDB took 0.026 seconds to read the Parquet file. These results suggest that Parquet file format is more efficient than CSV file format for data analysis tasks, and DuckDB provides better performance than Pandas for reading Parquet files.

Other studies have also reported the advantages of Parquet file format over other file formats. For instance, Parquet files are designed to work well with columnar data storage and processing systems, which makes them a suitable choice for data analytics tasks. Moreover, Parquet files support compression, which reduces storage requirements and improves query performance.

Results

The proposed architecture for enabling users to execute SQL queries over the contents of discovered datasets was successfully implemented and tested. The architecture consisted of several components, including the dataset conversion module, object storage module using MinIO, user interface modification module, query execution module using DuckDB, and performance evaluation module.

The implementation demonstrated that the proposed ar-

chitecture could successfully convert datasets to Parquet files, store them in an object store using MinIO, and enable users to execute SQL queries over the Parquet files using DuckDB. The performance evaluation module showed that the query response time was significantly faster when executing SQL queries over the Parquet files compared to executing the same queries over the CSV version of the dataset. This improvement in query performance is attributed to the use of the Parquet file format, which reduces the amount of data that needs to be read from disk and improves data compression.

Future Improvement

The project is designed as a separate version from Auctus to enable expansion to other dataset search engines. In the future, we plan to enhance the project's functionality by allowing users to input dataset URLs, enabling them to execute SQL commands on any dataset from any dataset search engine without the need to download it to their local machine.

Conclusion

In conclusion, the proposed architecture for enabling users to execute SQL queries over the contents of discovered datasets offers several advantages, including improved query performance, reduced storage costs, and increased flexibility in data exploration. The architecture leverages PyArrow, Pandas, Flask, DuckDB Python libraries, and Minio to achieve its objectives. The success of this architecture demonstrates the potential for enhancing the capabilities of dataset search engines and enabling users to make informed decisions about the suitability of discovered datasets for their needs.

References

1. Castelo, S., Rampin, R., Santos, A., Bessa, A., Chirigati, F., Freire, J. (2021). Auctus: a dataset search engine for data discovery and augmentation. Proceedings of the VLDB Endowment, 14(12), 2791-2794.

2. Rampin, R., Castelo, S., Santos, A., Bessa, A., Chirigati, F., Freire, J. (2020). Data discovery and augmentation at scale with Auctus. Proceedings of the 29th ACM International Conference on Information Knowledge Management, 2919-2922.

3. Castelo, S., Rampin, R., Santos, A., Bessa, A., Chirigati, F., Freire, J. (2020). Auctus: A System for Data Discovery and Augmentation. Proceedings of the 36th IEEE International Conference on Data Engineering Workshops, 284-287.

4. Bessa, A., Rampin, R., Santos, A., Castelo, S., Chirigati, F., Freire, J. (2021). Automating the generation of knowledge graphs from datasets. Proceedings of the VLDB Endowment, 14(12), 3086-3089.