Inclusion Dependencies Identification using Spark

Yasmine Batteieb (MTR: 3795910)
Ugur Taysi (MTR: 3806159)

January 30th, 2024

System Overview

The system, developed for the Big Data Systems course under the guidance of Prof. Dr. Thorsten Papenbrock, utilizes Apache Spark for distributed processing of CSV files to discover INDs. This process is integral to ensuring data integrity and consistency in relational databases.

1 Operational Flow

The operational flow of the program can be broken down into the following steps:

1. **Data Reading:** Utilizes `readData` to parse CSV files into Spark DataFrames.
2. **Data Transformation:** Employs `extractDataFromRows` to transform and tuple the data.
3. **Dependency Generation:** Applies `generateDependencyRows` to create tuples indicative of dependencies.
4. **Dependency Reduction:** Groups tuples by identifier and intersects dependency sets.
5. **Result Collection and Sorting:** Collects and sorts INDs alphabetically for final output.

Figure 1: Pipeline Transformation of IND Discovery

2 Implementation Details

The `discoverINDs` function is pivotal in orchestrating the overall process. It leverages several auxiliary functions to read, transform, and generate the necessary data for IND discovery.

2.1 Dependency Row Generation

This step involves creating tuples for each column along with sets of other columns that might be dependent on it. It's a preparatory step for the actual dependency analysis using the `generateDependencyRows` function.

2.2 DataFrame Combination

In cases where data is split across multiple files, this step merges them into a single DataFrame using DataFrame union operations. This is essential to ensure a holistic analysis across the entire dataset.

2.3 Inclusion Dependency Identification

1. **Grouping by Key:** Data is grouped by column names using the `groupByKey` function.

2. **Mapping Groups:** The core of the inclusion dependency discovery takes place here. By intersecting sets of column identifiers within each group using the `mapGroups` function, the actual dependencies between columns are identified.

2.4 Finalization of Results

1. **Filtering Non-empty Sets:** Post dependency analysis, sets without dependencies are filtered out using filter transformations.

2. **Collecting and Sorting Results:** The final step involves organizing the identified dependencies into a sorted list using collect and sortBy functions. This makes the data more readable and easier to interpret, marking the conclusion of the inclusion dependency discovery process.

3 Output

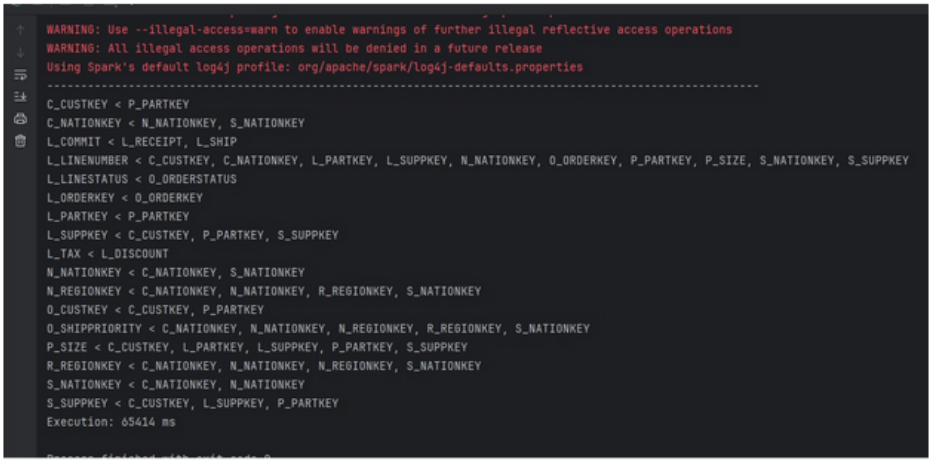

Figure 2: Output of the inclusion dependency discovery process

4 Conclusion

This report encapsulates the methodology and implementation of a Spark-based application to identify INDs within large datasets. The process illustrated herein signifies the importance of distributed computing in handling big data and the pivotal role of INDs in data integrity.