Akka ddm Project

Muhammed Kauser

Steve Oscar

Anjana Saji

Column Fetching from Dataset

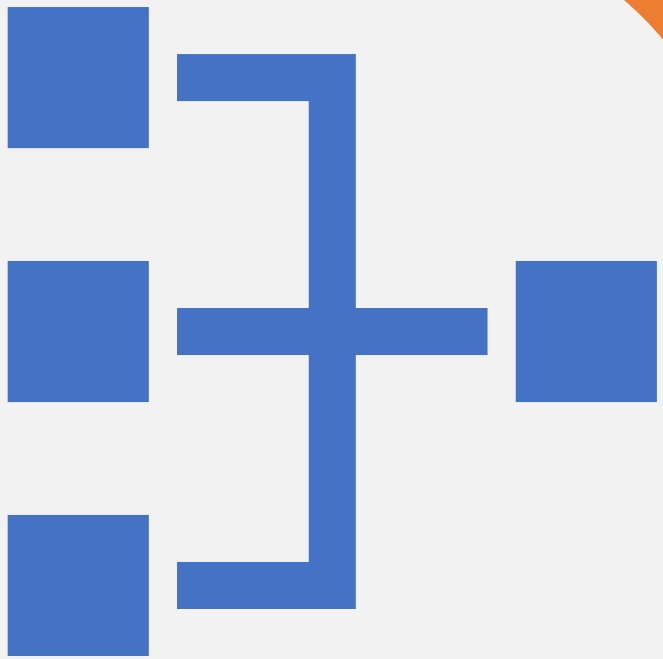
- Dynamic fetching of columns based on task requirements.
- Actors collaborate in a distributed system.
- Handles ReceptionistListingMessage, TaskMessage, and ColumnReceiver.
- Dynamically fetches required columns.

```
1 usage
Column(int id, String type, String columnName, String nameOfDataset){
    this.id = id;
    this.type = type;
    this.columnName = columnName;
    this.nameOfDataset = nameOfDataset;
    columnValues = new HashSet<>();
}
```

Act or State

State Variables:

- largeMessageProxy: Actor reference for large message handling.
- columnOfStrings and columnOfNumbers: HashMaps for storing columns.
- taskMessage: Stores the current task details.
- key1, key2, key3, key4: Task keys.
- compositeKeyPool: Pool for managing composite keys.



• Task Creation:

- Tasks are created for each combination of two columns (key pairs), considering both string and number columns.
- Tasks are represented by the DependencyWorker.TaskMessage class, which includes information about the columns involved and whether they are string or number columns.

```
7 usages  Steve *
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
public static class TaskMessage implements Message, LargeMessageProxy.LargeMessage {
    private static final long serialVersionUID = -4667745204456518160L;
    ActorRef<LargeMessageProxy.Message> dependencyMinerLargeMessageProxy;
    int taskId;
    // for the first column
    String key1;
    String key2;
    // for the second column
    String key3;
    String key4;s


    boolean isStringColumn;

}
```

Task Creation Code

```
private void creatingTaskLists() {
    //This is for columnOfNumbers
    for (CompositeKey key1 : columnOfNumbers.keySet()) {
        for (CompositeKey key2 : columnOfNumbers.keySet()) {
            if (!key1.equals(key2)) {
                taskDone.add(false);
                listOfTasks.add(
                        new DependencyWorker.TaskMessage(
                                dependencyMinerLargeMessageProxy: null,  taskId: -1,
                                key1.getSubKey1(),
                                key1.getSubKey2(),
                                key2.getSubKey1(),
                                key2.getSubKey2(),
                                isStringColumn: false));
            }

        }
    }
    //This is for columnOfStrings
    for (CompositeKey key1 : columnOfStrings.keySet()) {
        for (CompositeKey key2 : columnOfStrings.keySet()) {
            if (!key1.equals(key2)) {
                taskDone.add(false);
                listOfTasks.add(
                        new DependencyWorker.TaskMessage(
                                dependencyMinerLargeMessageProxy: null,  taskId: -1,
                                key1.getSubKey1(),
```

Task Execution and IND Check:

Workers execute tasks by checking for inclusion dependencies between the specified columns.

The findingIND() method is called after columns are received for a task.

The IND check involves comparing values in two columns and determining if one column is included in the other.

```
3 usages  Steve                                                                    ⚠5 ⚠6 ^
private void findingIND() {

    boolean result;
    Column column1;
    Column column2;
    if (taskMessage.isStringColumn()) {
        column1 = columnOfStrings.get(getCompositeKey(taskMessage.getKey1(), taskMessage.getKey2()));
        column2 = columnOfStrings.get(getCompositeKey(taskMessage.getKey3(), taskMessage.getKey4()));
    } else {
        column1 = columnOfNumbers.get(getCompositeKey(taskMessage.getKey1(), taskMessage.getKey2()));
        column2 = columnOfNumbers.get(getCompositeKey(taskMessage.getKey3(), taskMessage.getKey4()));
    }
    this.getContext().getLog().info("Looking for IND between {} and {}", column1, column2);
    result = column1.getColumnValues().containsAll(column2.getColumnValues());
    this.getContext().getLog().info("{}", result);

    LargeMessageProxy.LargeMessage resultMessage = new DependencyMiner.CompletionMessage(
            this.getContext().getSelf(),
            taskMessage.getTaskId(),
            result,
            column1,
            column2);

    this.largeMessageProxy.tell(new LargeMessageProxy.SendMessage(resultMessage, taskMessage.getDependencyMinerLargeMessageProxy()));

}
```

```
LargeMessageProxy.LargeMessage resultMessage = new DependencyMiner.CompletionMessage(
        this.getContext().getSelf(),
        taskMessage.getTaskId(),
        result,
        column1,
        column2);

this.largeMessageProxy.tell(new LargeMessageProxy.SendMessage(resultMessage, taskMessage.getDependencyMinerLargeMessageProxy()));
```

Upon completing an IND check, a CompletionMessage is sent back to the DependencyMiner with the result.

If an IND is found, relevant information is logged, and the result is sent to the ResultCollector.

Terminal Local

```
[22:23:54.848 INFO ]          akka://ddm/user/worker/dependencyWorker_0| New Task 129
[22:23:54.849 INFO ]          akka://ddm/user/worker/dependencyWorker_0| Looking for IND between de.ddm.actors.profiling.Column@24014f55 and de.ddm.actors.pro
filing.Column@71d4a7f9
[22:23:54.851 INFO ]          akka://ddm/user/worker/dependencyWorker_0| true
[22:23:54.854 INFO ]             akka://ddm/user/master/dependencyMiner| Received CompletionMessage from worker Actor[akka://ddm/user/worker/dependencyWorker_
0#-766591885] for task 129!
[22:23:54.854 INFO ]             akka://ddm/user/master/dependencyMiner| Received IND from worker Actor[akka://ddm/user/worker/dependencyWorker_0#-766591885]
for task 129!
[22:23:54.855 INFO ]             akka://ddm/user/master/dependencyMiner| IND is L_SHIP and tpch_lineitem.csv from tpch_lineitem.csv to tpch_lineitem.csv!
[22:23:54.855 INFO ]             akka://ddm/user/master/dependencyMiner| Giving task 130 to worker Actor[akka://ddm/user/worker/dependencyWorker_0#-766591885]
[22:23:54.855 INFO ] ://ddm/user/master/dependencyMiner/resultCollector| Received 1 INDs!
[22:23:54.856 INFO ]          akka://ddm/user/worker/dependencyWorker_0| The keys are tpch_lineitem.csv : L_SHIP and tpch_customer.csv : C_NATIONKEY
[22:23:54.856 INFO ]          akka://ddm/user/worker/dependencyWorker_0| New Task 130
[22:23:54.857 INFO ]          akka://ddm/user/worker/dependencyWorker_0| Looking for IND between de.ddm.actors.profiling.Column@24014f55 and de.ddm.actors.pro
filing.Column@11916e4d
[22:23:54.857 INFO ]          akka://ddm/user/worker/dependencyWorker_0| false
[22:23:54.858 INFO ]             akka://ddm/user/master/dependencyMiner| Received CompletionMessage from worker Actor[akka://ddm/user/worker/dependencyWorker_
0#-766591885] for task 130!
[22:23:54.859 INFO ]             akka://ddm/user/master/dependencyMiner| Giving task 131 to worker Actor[akka://ddm/user/worker/dependencyWorker_0#-766591885]
[22:23:54.860 INFO ]          akka://ddm/user/worker/dependencyWorker_0| The keys are tpch_lineitem.csv : L_SHIP and tpch_lineitem.csv : L_PARTKEY
[22:23:54.860 INFO ]          akka://ddm/user/worker/dependencyWorker_0| New Task 131
[22:23:54.861 INFO ]          akka://ddm/user/worker/dependencyWorker_0| Looking for IND between de.ddm.actors.profiling.Column@24014f55 and de.ddm.actors.pro
filing.Column@3ec36b19
[22:23:54.861 INFO ]          akka://ddm/user/worker/dependencyWorker_0| false
[22:23:54.956 INFO ]             akka://ddm/user/master/dependencyMiner| Received CompletionMessage from worker Actor[akka://ddm/user/worker/dependencyWorker_
0#-766591885] for task 131!
```

Conclusion

1.Actor-Based Dependency Discovery:

- The class is an Akka actor responsible for mining inclusion dependencies (IND) between columns in different datasets.

2.Task Distribution and Parallel Processing:

- Distributes tasks to DependencyWorker actors for efficient parallel processing of inclusion dependency discovery.

3.Dynamic Configuration and Result Collection:

- Utilizes dynamic configurations from SystemConfigurationSingleton and InputConfigurationSingleton.
- Communicates discovered inclusion dependencies to the ResultCollector actor for further processing.