H2 Assignment - 3

Implement an enhanced feature to H2 database in terms of query processing and optimization.

Feature :

Implementation of a new SECOND_MAX aggregate type

Overview:

- The H2 database does not currently support a aggregate type of the SECOND_MAX format. In this assignment, we have integrated a new aggregate type of the SECOND_MAX format which functions exactly like a MAX aggregate function with the only difference being that instead of returning the maximum numeric value from the column, it returns the second maximum numeric value from the column.

- The main motive behind choosing this type of aggregate function type is that most often than not, (sometimes along with the maximum value in a column) we wish to find the second maximum value as well, to maybe just visually see how much do the top 2 contenders for max values differ, with the sole aim of deriving insights from the data. And this aggregate is simple enough to understand the working of H2 and be able to implement our own aggregate

- Implementing a custom aggregate type for this would avoid any human effort or possible errors in manually calculating, figuring out or querying the second max value.

☐ A series of snapshots are attached below depicting how the custom aggregate type was integrated in the H2 database and what changes were made in the original H2 code.

Implementation of the SECOND_MAX Aggregate Type

Step 1: **Aggregate.java** (Path: src/main/org/h2/expression/aggregate/Aggregate.java) was created.

1) The aim of the file is to add the SECOND_MAX aggregate type.
2) At line 180, in the static body part of this class, we add the aggregate type SECOND_MAX.

```
177             addAggregate("JSON_OBJECTAGG", AggregateType.JSON_OBJECTAGG);
178             addAggregate("JSON_ARRAYAGG", AggregateType.JSON_ARRAYAGG);
179             // Custom compatibility
180             addAggregate("SECOND_MAX", AggregateType.SECOND_MAX);
181         }
```

3) Then in the createAggregateData function at line 416, where the new aggregate function SECOND_MAX is actually

registered as an aggregate type at line 443 in the snapshot below.

```
441         case MIN:
442         case MAX:
443         case SECOND_MAX:
444         case BIT_AND_AGG:
445         case BIT_OR_AGG:
446         case BIT_NAND_AGG:
447         case BIT_NOR_AGG:
448         case ANY:
449         case EVERY:
450             return new AggregateDataDefault(aggregateType, type);
```

4) Then in the function getValueQuick at line 507, here we define what to do when we encounter this type of aggregate function (line 514)

```
514             case SECOND_MAX: {
515                 System.out.println("INSIDE NTH MAX CASE");
516                 boolean firstFlag = aggregateType == AggregateType.SECOND_MAX;
517                 Index index = getMinMaxColumnIndex();
518                 System.out.println("QQQQQQQQ: " + index);
519                 int sortType = index.getIndexColumns()[0].sortType;
520                 if ((sortType & SortOrder.DESCENDING) != 0) {
521                     firstFlag = !firstFlag;
522                 }
523                 Cursor cursor = index.findNthMax(session, firstFlag);
524                 System.out.println("CURSOR: " + cursor);
525                 SearchRow row = cursor.getSearchRow();
526                 System.out.println("ROW: " + row);
527                 Value v;
528                 if (row == null) {
529                     v = ValueNull.INSTANCE;
530                 } else {
531                     v = row.getValue(index.getColumns()[0].getColumnId());
532                 }
533                 return v;
534             }
535             case MAX: {
```

5) Then in functions optimize and isEverything, we simply add a case for SECOND_MAX which executes what is executed by MAX aggregate type.

Step 2: **AggregateDataDefault.java** (Path: src/main/org/h2/expression/aggregate/AggregateDataDefault.java)

1) This file is where data is stored while calculating the 2nd max value as per the aggregate function is being performed.

2) This file extends AggregateData, an abstract class which declares functions for the computation of an aggregate.
3) In the beginning of the file, 2 variables are defined ‘value’ and ‘lower’ which are maximum value (current) or second maximum value (previous) respectively.
4) In the function ‘add’ at line 36, we create a case for SECOND_MAX which decides what to return when working with the SECOND_MAX aggregate type.

```
54          case SECOND_MAX:
55              if (value == null || session.compare(v, value) > 0) {
56                  System.out.println("In AggregateDataDefault: " + value + " : " + v);
57                  lower = value;
58                  value = v;
59              }
60              break;
61          case MAX:
```

5) Here we assign ‘lower’ the second maximum value and then update ‘value’ by v which contains the actual max value.
6) We also modify the function ‘getValue’ at line 114. We add a small if condition for the situation when we are working with SECOND_MAX aggregate type:-

```
113         @Override
114         Value getValue(SessionLocal session) {
115             Value v = value;
116             if (aggregateType.equals(AggregateType.SECOND_MAX)) {
117                 System.out.println("InsideNMAX CASE IN GET VALUE: ");
118                 v = lower;
119             }
120
```

Step 3: **AggregateType.java** (Path: src/main/org/h2/expression/aggregate/AggregateType.java)

1) This class is solely responsible for defining and storing the type of an aggregate function.
2) At line 236, we define SECOND_MAX.

```
233         /**
234          * The aggregate type for nth max element.
235          */
236         SECOND_MAX,
237
238     }
```

Step 4: **Index.java** (Path: src/main/org/h2/index/Index.java)

1) This file is where we declare the method 'findNthMax' which is basically responsible for pointing the cursor to the desired column in the database.

```
742
743         public Cursor findNthMax(SessionLocal session, boolean firstFlag) {
744             throw DbException.getInternalError(toString());
745         }
746     }
```

Step 5: **MVDelegateIndex.java** (Path: src/main/org/h2/mvstore/db/MVDelegateIndex.java)

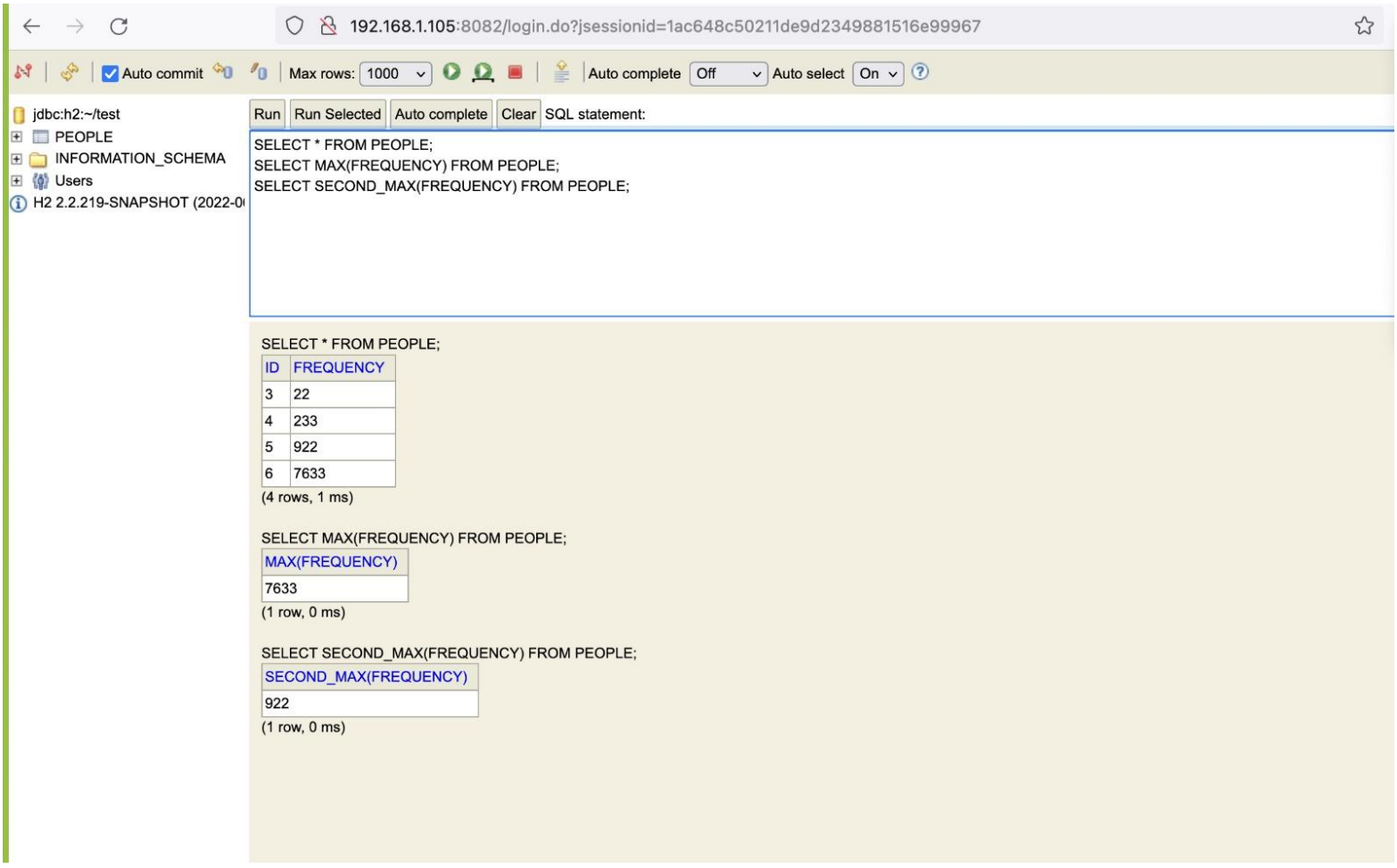
1) In this class we implement the function findNthMax.

```
95
96      @Override
97      public Cursor findNthMax(SessionLocal session, boolean first) {
98          return mainIndex.findNthMax(session, first);
99      }
100
```

FINAL TESTING

To test the working of our custom aggregate function type, we created a table ‘People’ with attributes ID and FREQUENCY. We entered some values and as output showed the maximum value and the second maximum value of the attribute FREQUENCY, for the reader to easily see and verify.

The following is the screenshot of the successful implementation of the custom aggregate type SECOND_MAX in the H2 database :



Authors

Janhavi Doshi (jd6583)
Kushal Kale (ksk7657)
Nimisha Bhagat (nb2633)
Vinay Jain (vj9898)