DSM23 Project Phase-2

Due date: 12-10-2023

Instructions

- This is a Team Project. You will be working on this project in a team as announced before.
- All commits are to be done in the master branch of the repository which is added to your team on github classroom. You may create separate branches, but you are supposed to merge it into the master branch by the due date.
- You are supposed to make your commits to github repository you made for Phase-1.
- Late Day Form to be filled on the day of submission, if any commit is made past the due date in the master branch. You are supposed to fill the correct date of the last commit as we will be using that to download your team's repository from github. No personal queries will be entertained. All queries regarding Project Phase 2 is to be posted on the doubts document.
- Be careful with Semantic and Syntactic errors(Eg: Spacing between query words).
- **All the sections in Task-2 should be done through External Sorting which you done in Task-1.**
- Not following any of the above instructions will **attract penalty**.
- **Note:** You need to create a **report.md** which reflects your work for the respective tasks and need to add in `docs` folder.

Suggested Readings:

```
Refer 18th chapter for any Algorithm related queries.
TextBook: Fundamentals of Database Systems, 7th Edition, R Elmasri,
Shamkant B Navathe.
```

TASK-1: IMPLEMENTATION OF EXTERNAL SORTING

Extend SimpleRA to implement SORT query. To accomplish this task, you can use the sorting algorithm which is discussed in class. You may assume the given data is within **algorithmic constraints** and the data may contain **duplicates**.

```
Given a relation R(K, A1, … An)

The relation stored in file f, must be put in sorted order of a subset of
attributes back in file f.
```

The following syntax has to be implemented in the codebase by you.

```
SORT <table_name> BY <column_name1, column_name2,..., column_namek> IN
<ASC|DESC, ASC|DESC,..., ASC|DESC>
```

Here <table_name> represents the table that has to be sorted and <column_name> is the name of the column in the table that the sort order is based on. ASC or DESC are used to denote ascending or descending orders. Priority of columns to be sorted is the order of column names given in the query. For each attribute, order must be mentioned. You need to sort in based on that order only.

Note:

```
Buffer size is the number of main memory buffer bolcks you are allowed to
use to carry out the sorting operation. Assume Buffer size = 10 blocks.
```

Example:

Consider an EMPLOYEE Table:

SSN	GENDER	SALARY
1	1	30,000
2	1	50,000
3	0	25,000
4	0	30,000
5	0	20,000
6	0	40,000
7	1	20,000
8	0	25,000

If the query to sort the list of all employees based on their Gender, then the statement would be:

```
SORT EMPLOYEE BY GENDER IN ASC
```

The Resultant table in this example would be:

SSN	GENDER	SALARY
3	0	25,000
4	0	30,000

SSN	GENDER	SALARY
5	0	20,000
6	0	40,000
8	0	25,000
1	1	30,000
2	1	50,000
7	1	20,000

In the same way, if the query is to sort the list of all employees based on their Gender and SALARY(Decreasing order), then the statement would be:

```
SORT EMPLOYEE BY GENDER, SALARY IN ASC, DESC
```

Now, the resultant table looks like:

SSN	GENDER	SALARY
6	0	40,000
4	0	30,000
3	0	25,000
8	0	25,000
5	0	20,000
2	1	50,000
1	1	30,000
7	1	20,000

TASK-2: APPLICATIONS OF EXTERNAL SORTING

We can use sorting technique in various Operations such as Join, GroupBy and order by queries. So, Here you have to implement Join, GroupBy and Order by operations in the codebase with the help of External Sorting (Done in above section).

PART-A: JOIN

You have to figure out the approach of how you link up the Join operation and External sorting algorithm. This algorithm creates new table(Assignment Operation). For implementation, we assume the following syntax to be strictly followed:

```
 <new_relation_name> <- JOIN <tablename1>, <tablename2> ON <column1>
<bin_op> <column2>
```

Here, `<new_relaton_name>` refers to the name of the newly created table after Join Operation. `<tablename1>, <tablename2>` are tables which we are performing join operations.`<column1> and <column2>` are the columns of same data used to compare for Joining. `<bin_op> = "==",">",">=","<","<=",`. (Comparision Operators).

Example:

Consider a Students Table:

ID	Age
2	21
3	23
1	18
5	26
4	17

Consider a Courses Table:

courseID	RollNo
3	2
6	3
5	4
2	5

courseID	RollNo
1	1
7	4
2	1

If the query to be run is to list all the students along with courses they are enrolled in, then the statement would be

```
Result <- JOIN Students, Courses ON ID == RollNo
```

The resultant table looks like:

ID	Age	courseID	RollNo
1	18	1	1
1	18	2	1
2	21	3	2
3	23	6	3
4	17	5	4
4	17	7	4
5	26	2	5

Note:

```
  The Result table must contain all the columns from both the tables. For
the sake of this assignment, you can assume that both tables will never
contain columns with matching names. The order of the columns must be
columns from the first table followed by columns from the second table.
```

PART-B :ORDER BY

Here, you need to implement ORDER BY Operation through External Sorting done in TASK-1. You have to order your table based on the given column.

The syntax of the ORDER BY statement is as follows:

```
<new_table> <- ORDER BY <attribute> ASC|DESC ON <table_name>
```

<attribute> means the column on which table should be sorted. It has to be done in mentioned order. <table_name> is the table on which the query is performed.

Example:

Consider a table of Movies:

Movie_ID	Title	Year_Released
1	Jersey	2019
2	KGF	2018
3	Pathaan	2023
4	RRR	2022
5	Dangal	2016

If the query is to order the table based on the released year scored by student, then the query looks like

```
Result <- ORDER BY Year_Released ASC ON Movies
```

The resultant table should looks like:

Movie_ID	Title	Year_Released
4	Dangal	2016
2	KGF	2018
1	Jersey	2019
4	RRR	2022
5	Pathaan	2023

PART-C GROUP BY:

In this section, you need to implement GROUP BY Aggregates Operation with a HAVING clause. This is done through External sorting which has done in TASK-1. Here Aggregate functions can be max,min,count,sum,avg etc.

The syntax of the GROUP BY statement is as follows:

```
  <new_table> <- GROUP BY <grouping_attribute> FROM <table_name> HAVING
<aggregate(attribute)> <bin_op> <attribute_value> RETURN
<aggregate_func(attribute)>
```

Here, <new_table> is the final table which is created and stores the result of group by operation. <grouping_attribute> is the column in the table based on which table has to be sorted. <attribute> column which is used to test(Compare) with column values. <bin_op> is the comparison operator mentioned above. Aggregate functions are MIN,MAX,SUM,COUNT,AVG.

Example:

Consider an Employee table:

EmployeeID	Department_ID	Salary
1	1	80,000
2	2	60,000
3	2	20,000
4	3	60,000
5	1	25,000
6	4	25,000

If the query is to find, the Department_ID and the maximum Salary in that Department where the average Salary of that Department > 50,000, then the query is:

```
T1 <- GROUP BY Department_ID FROM EMPLOYEE HAVING
AVG(Salary) > 50,000 RETURN MAX(Salary)
```

The resultant table is:

Department_ID	MAXSalary
1	80,000
3	60,000

If the query is to find, the Department_ID and Sum of salary in that Department where the sum of Salaries in that Department > 70,000, then the query is :

```
T2 <- GROUP BY Department_ID FROM EMPLOYEE HAVING
SUM(Salary) > 70,000 RETURN SUM(Salary)
```

The resultant table is:

Department_ID	SUMSalary
1	1,05,000
2	80,000

Note: The output column name will be the aggregate operator concatenated with column name it's operating on

```
MAX|MIN|SUM|AVG(X) -> MAX|MIN|SUM|AVGX
```

Report:

Submit a Report.md in the docs folder including the following details:

- Detailed Report for each command you have implemented. Detailed Explanation of External Sorting algorithm.
- Elaborate explanation about how you performed JOIN, GROUP BY HAVING, ORDER BY using external sorting algorithm.
- All the assumptions should be mentioned in this report under the title `Assumptions`.
- Your learning and take away from this Phase, title `Learnings`
- Contribution of each member in the Project and how you have coordinated as a Team.