

# Phase 0

Data Systems

Deadline: 11:55PM 26th August 2022

Concerned TAs: Kartik Garg, Ayush Goyal

August 17, 2022

## 1 Part 1 - Writing Queries

The first part of this phase is to get you familiar with the query language implemented. Please refer to the overview.md or overview.html document present in the docs folder of your repositories to get the list of implemented commands.

A basic version of the employee database with various tables has been provided to you in the data folder. You can find a detailed description of the dataset at this . Remember your queries will be evaluated on a superset of this database.

**Q1:** List all employees with salaries  $\geq 30000$   
output: Q1.csv - Q1(Ssn, Salary)

**Q2:** List all pairs of employees and their supervisor both of whom have the same birth date  
output: Q2.csv - Q2(Ssn, Super\_ssn, Bdate)

**Q3:** List all projects which have both male and female employees working on them  
output: Q3.csv - Q3(Pno)

**Q4:** List all projects (along with the employee and his supervisor) on which both an employee and his/her supervisor work  
output: Q4.csv - Q4(Ssn, Super\_ssn, Pno)

## 2 Part-2 : Extending SimpleRA to Include Matrices

The Objective of the second part of the project is to get you familiar with the boilerplate code structure. In this part, you have to extend the boilerplate which currently only handles tables, to handle matrices as well.

Square  $n \times n$ , ( $n \leq 10^5$ ) matrices will be provided as csv files in the data folder. For instance, Given A

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

To load A into the system, a file called A.csv should be present in the data folder with the following contents

```
1 1,2
2 3,4
```

In matrices, there are no column names, so unlike tables, the first row of the csv file will not contain the column names, it contains the first row of the matrix. As a part of this phase, you have to implement the following commands

## 2.1 LOAD MATRIX <matrix\_name>

Like the LOAD command for tables, the LOAD MATRIX command should load the contents of the csv and store it as blocks in the temp directory

Therefore, to load matrix A into your system, you should be able to run

```
1 > LOAD MATRIX A
```

You are free to experiment with the page layout. It is important to note that you have to appropriately change the cursor and the buffer pool to handle matrices.

IMPORTANT: In case of tables, we implicitly assume a block is at least as big as a table row, but this isn't the case for matrices. A whole row or column of the matrix need not fit into a block. The maximum allowed size of a block is 8KB. The BLOCK\_SIZE parameter in the boilerplate is in KB.

## 2.2 PRINT MATRIX <matrix\_name>

Like the PRINT command for tables, you must implement a PRINT MATRIX command that prints the matrix on the terminal. If the matrix is big(i.e.  $n > 20$ ), print only the first 20 rows.

For example, the following output is expected

```
1 > PRINT MATRIX A
2 1 2
3 3 4
```

## 2.3 CROSS\_TRANSPOSE <matrix1\_name> <matrix2\_name>

You have to implement a new Non-Assignment Operation called CROSS\_TRANSPOSE. This Operation is an update operation, so the initial matrices has to be updated.

You are expected to use IN PLACE updates, not doing so will result in either getting a zero (for this command) or a heavy penalty.

- 1 IN PLACE updates means not using any additional disk blocks. Although, you are allowed to use a limited amount of main memory (say 2 blocks).
- 2 During Evaluation we will be monitoring both the disk blocks (the temp directory) and the main memory usage

### 2.3.1 DESCRIPTION

Given two matrices A and B, CROSS\_TRANSPOSE is a kind of operation such that after execution A becomes  $B^T$  and B becomes  $A^T$ . That is the blocks corresponding to the original matrix A now store the transpose of the matrix B and similarly the blocks corresponding to the original matrix B now store the transpose of the matrix A

### 2.3.2 CONSTRAINTS

- Both the matrices are supposed to be square matrices of same size, lets say  $n * n$

### 2.3.3 EXAMPLE

```
1 > LOAD MATRIX A
2 > PRINT MATRIX A
3 1 2
4 3 4
5 > LOAD MATRIX B
6 > PRINT MATRIX B
7 5 6
8 7 8
9 > CROSS_TRANSPOSE A B
10 > PRINT MATRIX A
11 5 7
12 6 8
13 > PRINT MATRIX B
14 1 3
15 2 4
```

### 2.4 EXPORT MATRIX <matrix\_name>

Just like for tables, the EXPORT command should write the contents of the matrix named <matrix\_name> into a file called <matrix\_name>.csv in the data directory

Therefore, if matrix A and matrix B has been operated upon using the CROSS\_TRANSPOSE operation by running the CROSS\_TRANSPOSE A B command and then we subsequently call EXPORT MATRIX A and EXPORT MATRIX B

```
1 > LOAD MATRIX A
2 > PRINT MATRIX A
3 1 2
4 3 4
5 > LOAD MATRIX B
6 > PRINT MATRIX B
7 5 6
8 7 8
9 > CROSS_TRANSPOSE A B
10 > EXPORT MATRIX A
11 > EXPORT MATRIX B
```

Then the contents of the A.csv file in the data folder should be updated and look like

```
1 5 7
2 6 8
```

And similarly the contents of B.csv file in the data folder should be updated and look like

```
1 1 3
2 2 4
```

NOTE: For Every Command perform appropriate syntactic and semantic checks

## 3 Part 3 : Extending Part 2 to Sparse Matrices

In this part, you are only supposed to write a report. NO CODE WRITING is involved in this part. Suggest a scheme following which you can incorporate sparse matrices in your DBMS. You have to describe each one of the below mentioned things

- Hsuya being a CS geek does not want to waste a useful resource as Memory and Since sparse matrices are in question so he knows that a majority part of these matrices contain only zeroes so in order to save space/memory he want to store these matrices in a compressed format. But being a geek he always remains busy with his other courses and he wants your help with it. So suggest a technique which can help Hsuya to solve this issue. Also explain the page layout which needs to be followed corresponding to this technique.
- Once you have made a decision about the compression technique and the page layout to be followed, another Geek Hsnayirp comes and all of a sudden wants to take transpose of a sparse matrix but since he didn't work on coming up with a compression technique and a page design by his own he is struggling with it and needs your help. So explain the strategy you would follow to take the transpose of a sparse matrix considering the page design and compression technique you used in the previous part.

BONUS: If you implement this in your codebase then that will be counted towards your bonus

## 4 Submission Instructions

All your code should be pushed to your GitHub repository. At the deadline, your codebase will be automatically downloaded.

For Part 1, all your queries must be written in a file called queries.txt in the data directory of your repository. The contents of the queries.txt file should follow this format

```

1 LOAD EMPLOYEE
2 ...
3
4 <Q1 query>
5 ...
6
7 <Q4 query>
8 ...
9
10 EXPORT Q1
11 ...
12 EXPORT Q4
13 QUIT
```

- Each query can consist of multiple statements
- Every query should export a table(Q1-Q4). If the result of a query is an empty set, please still include the EXPORT command (it will print an error statement if the table doesn't exist, which is okay)
- The table you generate can have duplicate rows. The order of rows in the resultant tables does not matter
- Please follow the provided output format.

In addition to the code, you also have to submit a report titled matrix.md that should be present in the docs directory of your repository. The contents of the report should contain the following information

- Page layout to store matrices (corresponding to PART 2)
- Describe how you are implementing the in place CROSS\_TRANSPOSE operation (corresponding to PART 2)

- PART 3 Discussion

For any doubts please use the moodle forum. If you need to interact with a TA drop an email to any of us and we will find and respond with a appropriate time slot.

**This course is intolerant to plagiarism. Any plagiarism will lead to an F in the course**

NOTE: We will be checking plagiarism with senior batches too