Data Systems Project Phase 1

Team

Team member name	Roll number	Email ID
Harshita Gupta	2020101078	harshita.gupta@students.iiit.ac.in
Naimeesh Narayan Tiwari	2020101074	naimeesh.tiwari@students.iiit.ac.in
Aarush Jain	2020101016	aarush.jain@students.iiit.ac.in

Blocks accessed

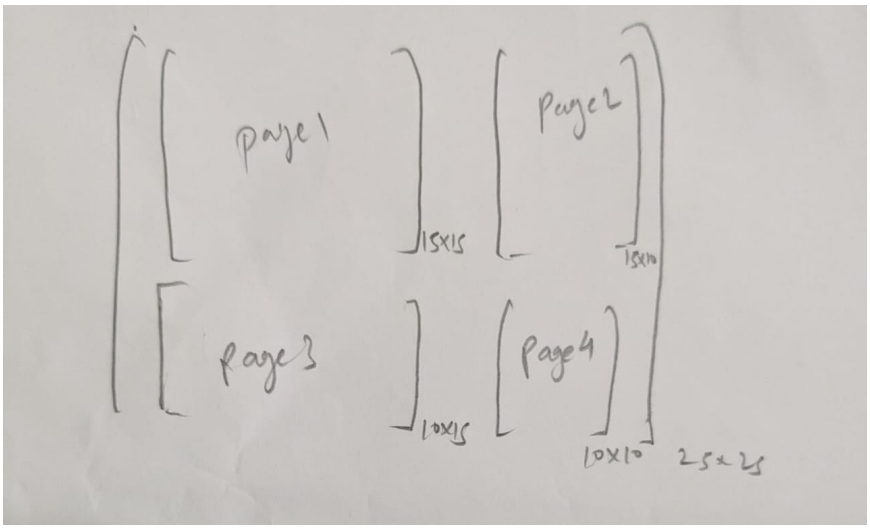
The number of blocks read is updated in **getPage** only counting the case where the block is loaded into the main memory from disk i.e when the page is inserted into the pool.

The number of blocks written is updated in **writePage.**

In the case of matrices, for all commands other than TRANSPOSE, this number is more correctly the number of writes in the pages because we append rows one by one in separate pages while loading the matrix.

Page Design

- For implementing the commands for matrices, we have saved matrices in the form of sub-matrices i.e. each sub-matrix is stored on a different page.
- The max number of rows in a submatrix is 15, so a page can contain a sub-matrix of a maximum of 15*15 dimensions.
- for example, in a 25*25 matrix:



Error Handling

- The syntactic parser checks if the command entered is correct or not.
  - the command is flagged invalid if the query size is not 4 for the following executors:
    1. RENAME
  - the command is flagged invalid if the query size is not 3 for the following executors:
    1. LOAD
    2. PRINT
    3. EXPORT
    4. TRANSPOSE
  - the command is flagged invalid if the query size is not 2 for the following executors:
    1. CHECKSYMMETRY
    2. COMPUTE
- Syntactic parser also checks if the command type is valid or not. (executor type)
- The semantic parser checks whether such a Matrix exists in the Catalogue.

LOAD MATRIX

1. Rows are picked from the CSV file (memory) one by one.
2. Each row is then divided based on which columns belong to which page (for example in the 25*25 matrix, the 15th column for row 1 belongs to Page 1 and the 16th column for row 1 belongs to Page 2.
3. Rows are appended to the pages to which they belong.

```
> LOAD MATRIX matrix
Loaded Matrix. Column Count: 1000 Row Count: 1000
Number of blocks read: 0
Number of blocks written: 67000
Number of blocks accessed: 67000
```

PRINT MATRIX

1. To find the page number, use the submatrix index.
2. Go to that page and get the desired row.
3. Repeat this step for all pages containing parts of that row.
4. If the matrix is larger than 20 rows and 20 columns, only display the first 20 rows and 20 columns.

```
> PRINT MATRIX matrix
1, 1001, 2001, 3001, 4001, 5001, 6001, 7001, 8001, 9001, 10001, 11001, 12001, 13001, 14001, 15001, 16001, 17001, 18001, 19001
2, 1002, 2002, 3002, 4002, 5002, 6002, 7002, 8002, 9002, 10002, 11002, 12002, 13002, 14002, 15002, 16002, 17002, 18002, 19002
3, 1003, 2003, 3003, 4003, 5003, 6003, 7003, 8003, 9003, 10003, 11003, 12003, 13003, 14003, 15003, 16003, 17003, 18003, 19003
4, 1004, 2004, 3004, 4004, 5004, 6004, 7004, 8004, 9004, 10004, 11004, 12004, 13004, 14004, 15004, 16004, 17004, 18004, 19004
5, 1005, 2005, 3005, 4005, 5005, 6005, 7005, 8005, 9005, 10005, 11005, 12005, 13005, 14005, 15005, 16005, 17005, 18005, 19005
6, 1006, 2006, 3006, 4006, 5006, 6006, 7006, 8006, 9006, 10006, 11006, 12006, 13006, 14006, 15006, 16006, 17006, 18006, 19006
7, 1007, 2007, 3007, 4007, 5007, 6007, 7007, 8007, 9007, 10007, 11007, 12007, 13007, 14007, 15007, 16007, 17007, 18007, 19007
8, 1008, 2008, 3008, 4008, 5008, 6008, 7008, 8008, 9008, 10008, 11008, 12008, 13008, 14008, 15008, 16008, 17008, 18008, 19008
9, 1009, 2009, 3009, 4009, 5009, 6009, 7009, 8009, 9009, 10009, 11009, 12009, 13009, 14009, 15009, 16009, 17009, 18009, 19009
10, 1010, 2010, 3010, 4010, 5010, 6010, 7010, 8010, 9010, 10010, 11010, 12010, 13010, 14010, 15010, 16010, 17010, 18010, 19010
11, 1011, 2011, 3011, 4011, 5011, 6011, 7011, 8011, 9011, 10011, 11011, 12011, 13011, 14011, 15011, 16011, 17011, 18011, 19011
12, 1012, 2012, 3012, 4012, 5012, 6012, 7012, 8012, 9012, 10012, 11012, 12012, 13012, 14012, 15012, 16012, 17012, 18012, 19012
13, 1013, 2013, 3013, 4013, 5013, 6013, 7013, 8013, 9013, 10013, 11013, 12013, 13013, 14013, 15013, 16013, 17013, 18013, 19013
14, 1014, 2014, 3014, 4014, 5014, 6014, 7014, 8014, 9014, 10014, 11014, 12014, 13014, 14014, 15014, 16014, 17014, 18014, 19014
15, 1015, 2015, 3015, 4015, 5015, 6015, 7015, 8015, 9015, 10015, 11015, 12015, 13015, 14015, 15015, 16015, 17015, 18015, 19015
16, 1016, 2016, 3016, 4016, 5016, 6016, 7016, 8016, 9016, 10016, 11016, 12016, 13016, 14016, 15016, 16016, 17016, 18016, 19016
17, 1017, 2017, 3017, 4017, 5017, 6017, 7017, 8017, 9017, 10017, 11017, 12017, 13017, 14017, 15017, 16017, 17017, 18017, 19017
18, 1018, 2018, 3018, 4018, 5018, 6018, 7018, 8018, 9018, 10018, 11018, 12018, 13018, 14018, 15018, 16018, 17018, 18018, 19018
19, 1019, 2019, 3019, 4019, 5019, 6019, 7019, 8019, 9019, 10019, 11019, 12019, 13019, 14019, 15019, 16019, 17019, 18019, 19019
20, 1020, 2020, 3020, 4020, 5020, 6020, 7020, 8020, 9020, 10020, 11020, 12020, 13020, 14020, 15020, 16020, 17020, 18020, 19020
Number of blocks read: 4489
Number of blocks written: 0
Number of blocks accessed: 4489
```

EXPORT MATRIX

Similar to PRINT, this time we write to a file.

```
> EXPORT MATRIX matrix
Number of blocks read: 4489
Number of blocks written: 0
Number of blocks accessed: 4489
```

RENAME MATRIX

We first rename the matrix in the catalogue. Then we rename the pages corresponding to the old matrix in the temp folder as well as in the buffer manager.

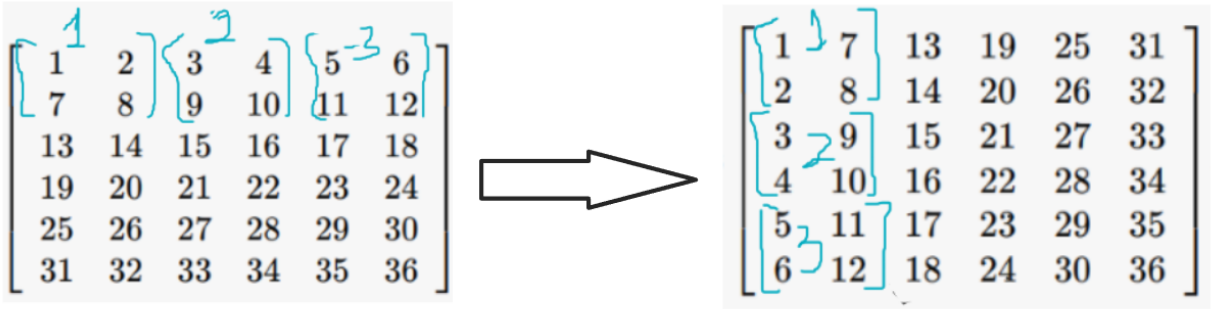
```
> RENAME MATRIX matrix AX
Number of blocks read: 0
Number of blocks written: 0
Number of blocks accessed: 0
```

TRANSPOSE MATRIX

The Logic here goes like this:

When you flip a big matrix, each smaller section within it also flips. These smaller sections switch places with their matching sections on the opposite side of the diagonal.

To make this happen, we access and flip each small section one by one using page numbers and the buffer pool, and then put them back on the other side of the diagonal.



```
> TRANSPOSE MATRIX AX
Number of blocks read: 4489
Number of blocks written: 4489
Number of blocks accessed: 8978
```

CHECKSYMMETRY

We iterate through the submatrices (stored in separate pages). The submatrix at (i, j) will be stored in page : **i * this->totalSubmatricesAcrossRow + j**

For submatrices along the diagonal (i == j) we check if each of them is symmetric.

For submatrices in the upper triangle (i>j) we check if it is equal to the transpose of the corresponding submatrix in the lower triangle.

If all these conditions are satisfied, we return **true**, else **false**.

```
> CHECKSYMMETRY AX
FALSE
Number of blocks read: 133
Number of blocks written: 0
Number of blocks accessed: 133
```

COMPUTE

We iterate through the submatrices in a similar way. For every submatrix, we subtract the transpose of the corresponding submatrix on the opposite side of the diagonal from it. We write this resultant matrix into a new page corresponding to the new matrix A_RESULT.

```
> COMPUTE AX
Number of blocks read: 8911
Number of blocks written: 4489
Number of blocks accessed: 13400
```

Learnings

- Designing Pages
- Efficiently Managing Pages and Memory Access
- Matrix Transposition Using Sub-matrices
- Problem-Solving Skills

Contributions

Team member name	Contributions
Harshita Gupta	EXPORT, CHECKSYMMETRY, COMPUTE, Error handling
Naimeesh Narayan Tiwari	PRINT, TRANSPOSE, COMPUTE, Testing
Aarush Jain	blocks accessed, LOAD, RENAME, Report, Testing

Key Takeaways

- We utilized Github and VS Code Liveshare for collaboration.
- Our initial steps involved discussing and formulating the foundational mathematics necessary for page indexing, access, and other operations.
- Subsequently, we focused on translating these concepts into practical implementation.