DBMS PROJECT

Instructions:

- It is an individual project.
- Project division will be according to the following formula:
Project Number = (k mod 9) + 1; where *k* = last three digits of your roll number.
- Use any object-oriented programming language.
- You need to demo the project as well as submit the code.
- Make sure you comment your code properly and proper indentation is maintained.
- Academic honesty policy applies well here, and anyone found guilty of cheating will receive an F grade.

Assumptions:

• Relations cannot fit into main memory. Hence, organize your data into pages. For this, necessary classes are provided.
• All the attributes of the relation are of integer data type.
• All the tuples are of fixed-length. The tuple size for example given below is 16 bytes.
• Each column is separated by a space and each tuple by a newline character.
For example, the input relation file will look like:

```
0001 01 153 184
0002 02 485 182
0003 03 485 641
0004 02 183 613
0005 03 184 179
0006 04 177 184
0007 02 638 686
0008 01 813 750
```

• Parameters like page size, buffer size, indexing/join attribute etc. will be run time parameters.Their values SHOULD NOT be hard coded into the program.

• The format of the file containing parameters will be: *parameter_name value*
For example,

```
page_size 1024
buffer_size 100
num_cols1
tuple_size1 12
attrb1 2
```

Necessary Parameters:

Name	**Description**
page_size	This parameter will represent the size of a page. The value corresponding to this parameter will be in bytes.
buffer_size	This will represent the maximum number of pages that can be in memory at a time. For example, if the value of this parameter is

	100, this means not more than 100 pages can be in memory at any time.
num_cols1	The value corresponding to this parameter will represent the number of columns in the first input relation, where each column is separated by a space as mentioned above.
tuple_size1	This represents size of each tuple (in bytes) of first relation.

Optional Parameters:

Name	**Required In**	**Description**
bucket_size	Linear Hashing	This parameter will represent the size of a bucket. The value corresponding to this parameter will be in bytes. This value will be same as page_size.
intial_bucket	Linear Hashing	This will give you the number of initial buckets required in Linear Hashing (should be power of 2).
attrb1	Linear Hashing, All Indexes	This will give you the hashing attribute column number(in the case of Linear and Extendible hashing) or indexing attribute column number (in the case of Primary, Secondary and Clustered indexes).
join_attrb1	All Join operations	This represents the column number of the first relation on which join has to be performed.
join_attrb2	All Join operations	This represents the column number of the second relation on which join has to be performed.
num_cols2	All Join operations	The value corresponding to this parameter will represent the number of columns in the second input relation, where each column is separated by a space as mentioned above.
tuple_size2	All Join operations	This represents size of each tuple (in bytes) of second relation.

• The parameters in the parameter file can be specified in any order.
• Columns are numbered starting from 1.
• Your program will take two/three command line arguments, the last one will be parameter file name, and the initial one/two will be input relation(s) filename(s).
For example,
./a.out relation1.txt relation2.txt parameter.txt (in case of joins)
./a.out relation1.txt parameter.txt (for the rest)
Note: Number of pages accessed => number of times the pages are being brought in main memory (i.e. buffer).

Note: You can mail all your queries and doubts to the ta who is in charge of your project only. Every project has a ta in charge.

1. Title: Multi-level Indexing on Clustered index at first level

TA in charge: Prashasti Gupta

Description : Given a relation sorted on non-prime attribute, create a first level clustered index on it .After that apply the concepts of multi-level indexing on it to bring the final index file to one block.

Input :
a) One large relation with at least 2 attributes. (Ordered on at least one)
b) Parameter file containing
    i) All the Necessary Parameters
    ii) Attrb1 (non-prime attribute on which relation is ordered)

Output:

1) Display for a relation ,
    a. No. of levels created
    b. No. of blocks used for storing index file

2) Given a query of equality or inequality on non-prime attribute,
    a. Display all the records which match
    b. No. of block access for each query.

Example:
Input:

]$./a.out relationfile parameter file
=50

]$./a.out relationfile parameter file
>=50

Output:
Levels: 4
Blocks: 64

1 50 23
2 50 67
3 50 1
…

Block access: 7

---

2. Title-External Memory set operation for union and intersection sql queries

TA in charge : Raghunath Reddy

Description -The set operation needs to be performed using external memory as the result would not fit in the given buffer size. The attributes in the queries are union and intersection compatible. You should not perform the union or intersection by sorting the intermediate results.

Input format:
select a,b,c from relations where condition1
union or intersection operator
select d,e,f from relations where condition2

output:
The tuples satisfying the above query

---

3. Title:- Left Outer Join Using Linear Hashing

TA in charge : Arnav Pyasi
Description: Given two relations, hash its joining attributes using linear hashing technique and store it in a separate file. Left Outer join them on given join attributes using Hash join based upon the hashing.

Input Format:
a) Two large unordered relations
b) Parameter file containing
i) all the Necessary Parameters
ii) initial_bucket
iii) join_attrb1, join_attrb2

Output Format :
a) Result of the left outer join operation
b) Number of pages accessed while performing left outer join.
c)Total number of buckets required for hashing for relation1 and relation2 respectively.

---

4. Title : Left outer join using Index Nested Loop Join .
TA in charge : Shivam Agarwal

Description : Index Nested Loop join is done when an index exist on one of the relations. First Create a index of your choice on inner relation . Then carry out Left outer join using

Index Nested Loop Join .
Input:

a) Two Large Unordered Relations
b) Parameter file containing
    i) Join_attr1 and Join_attr2
    ii) All other necessary parameters i.e. Tuple Size , Buffer Size , Page Size
Output:
a) Result of Left Outer Join
b) Number of pages accessed in Left Outer Join
c) Type of index maintained and number of pages required to store the index.

---

5. Title: "Right Outer Join" using "Sort-Merge Join with External Sorting"
TA in charge : Anshul Bansal

Input:
a) Two large unordered relations
b) Parameter file containing
i) all the Necessary Parameters
ii) join_attrb1, join_attrb2

Output:
a) Result of the Right Outer join operation
b) Number of pages accessed while sorting each of the unordered relations
c) Total number of pages accessed while merging to get Right Outer join result

Description:
Given two unordered relations, first sort each of the relations using external sorting technique. You can implement your favorite sort algorithm to perform external sort. Right Outer Join these sorted relations on given join attributes using Sort-Merge join.

Your program should accept following kinds of inputs:
Shell$] ./a.out relationfile1 relationfile2 parameterfile

---

6. Title:"Full Outer Join" using "Nested Loop Join"

TA in charge : Akshat Surana
Input:
a) Two large unordered relations
b) Parameter file containing
  i) all the Necessary Parameters

ii) join_attrb1, join_attrb2

Output:
a) Result of the Full Outer join operation
b) Number of pages accessed for each of the relations.

Description:
Given two relations, perform Full Outer Join on them using Nested Loop Join.

Your program should accept following kinds of inputs:
Shell$] ./a.out relationfile1 relationfile2 parameterfile

---

7. Title: Aggregate functions using clustered index on non key attribute.
TA in charge: Shraddha Agrawal
Input:
a) One large unordered relation
b) Parameter file containing
    i) All the Necessary Parameters
    ii) Attrb1

Output:
a) Given some aggregate function queries based on the clustered indexing, print the corresponding output. Such aggregate function involves queries such as MAX, MIN, COUNT DISTINCT, and COUNT ALL.
b) Given some aggregate function queries NOT based on the clustered indexing, print the corresponding output. Such aggregate function involves queries such as SUM, and AVG.

Description:
Given an ordered relation (relation is sorted on a non-key attribute) , create a clustered index on this. Based upon this indexing and the dataset, output the given query results, all related to above mentioned aggregate functions.

Your program should accept following kinds of inputs:
Shell$] ./a.out relationfile1 parameterfile
MIN //implies min from indexed attribute

Shell$] ./a.out relationfile1 parameterfile
COUNT //implies select count of total tuples in the relationfile1

Shell$] ./a.out relationfile1 parameterfile
COUNT DISTINCT //implies select count of distinct tuples based on the indexing.

Shell$] ./a.out relationfile1 parameterfile

SUM //implies total sum of the indexed attribute. However, indexing doesn't help here; you need to access each tuple of the dataset.

---

8. Title: Primary Indexing using B+ Trees

TA in charge : Gaurav Kharkwal

Description: Given a relation sorted on a primary key attribute, create a B+ tree using that attribute as the search key.

Input:

a) One large relation with at least 2 attributes (ordered on the primary key).
   - All attributes are of integer data type.
   - Each column represents one attribute.
   - Tuple size should be fixed; the example below has tuple-size 16 bytes.
   - For example, the input relation file may look like:

```
0001 01 153 184
0002 02 485 182
0003 03 485 641
0004 02 183 613
0005 03 184 179
0006 04 177 184
0007 02 638 686
0008 01 813 750
```

b) Parameter-file containing:
   - All the Necessary Parameters
     - Parameters are of type:
       - prim_key – the column number containing the primary key
       - num_cols – the number of columns
       - tuple_size – the size of the tuples
   - The format of the file containing parameters will be: *parameter_name value*
     - For example:

```
prim_key 1
num_cols 4
tuple_size 16
```

Output:

3) Display for a relation:
   a. No. of levels created
   b. No. of blocks used for storing index file

4) Given a query of equality or inequality on the primary key,
   a. Display all the records which match

Example:
Input is of the form:
$ ./a.out relation_file parameter_file
=50

$ ./a.out relation_file parameter_file
>=50 and <100 # implies, select * from relation_file where
prim_key >=50 and prim_key <100;

$ ./a.out relation_file parameter_file
=50 or =100

Output should be of the form:
Levels: 4
Blocks: 64

1 50 23
2 50 67
3 50 1
…
(just an example)

---

9. Project Title: Correlated nested queries with primary index on the inner query attributes.

TA in charge : Lini Thomas
Description:

The format of the query to be processed is:

SELECT <list of attributes>
FROM Relation1
WHERE Attr1 = (SELECT Attr2 FROM Relation2 WHERE <condition>)

where Attr1 is an attribute of Relation1 and Attr2 is the primary key of relation "Relation2". <list of attributes> is a set of attributes that belong to Relation1 and <condition> is a simple condition on one of the attribute of Relation2. Assume that all the attributes of both Relation1 and Relation2 are integer valued attributes.

You need to process the above query in the following manner:

i. Build primary index for Relation2 on attribute Attr2.
ii. For every tuple from Relation1, extract the value of Attr1 and using the primary index

of Relation2, get the tuple of Relation2 that has same value as Attr1 and check whether it satisfies <condition>.
iii. For each tuple of Relation1 that satisfies the condition in (ii), extract the attributes in <list of attributes> and print.

Input: The input for the project contains:
a) the description of the two relations of the query whose format is:

Name Of Relation1
Number of Attributes of Relation1
List of attributes of Relation1 separated by space
Name Of Relation2
Number of Attributes of Relation2
List of attributes of Relation2 separated by space

b) The tuples of the two relations Relation1 and Relation2 which each tuple in one line and columns separated by space.

c) the input query

Sample Input:

Please find in the attachment sample inputs:
a) desc.txt: the description of two relations R1 and R2
b) r1.txt and r2.txt: the tuples of two relations R1 and R2
c) query.txt: Sample query

Notes: The schemas of the relations are: R1(A int, B int, C int, D int) and R2(P int, Q int, R int, S int) where A and P are the primary keys of R1 and R2 respectively.
Example Query: SELECT A,B from R1 where D = (SELECT P from R2 where S > 2000).

The sample output is in the file: "output.txt" that contains the output of the query in "query.txt".