Modern Database Systems
Assignment 1

Due by **19 February 2016**

Note 1 Use mycourses.aalto.fi to submit your solutions. The name of the file you submit should have the following format.

modernDB1 – [your-student-id] – [your-full-name].pdf

Moreover, make sure that the cover (first page) of your pdf contains the following pieces of information: (i) first name, (ii) surname, (iii) student id, and (iv) email.

Note 2 This assignment is personal and no solutions should be shared. If you have discussed the tasks with someone else, please mention their name in your submission.

Note 3 The grading scheme is as follows.

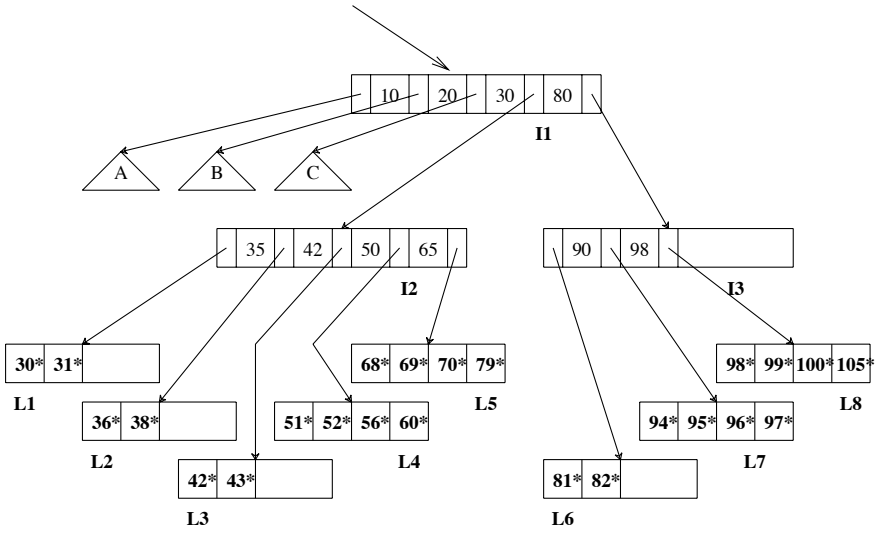
	Part A	Part B	Total
Max Points	15	15	30

Points of individual questions are mentioned next to the question numbers.

PART A: Pen and Paper

Question 1 (3 points)

Consider the B+ tree index shown in the figure below, which uses alternative type (1) for data entries. Each intermediate node can hold up to five pointers and four key values. Each leaf can hold up to four records, and leaf nodes are doubly linked as usual, although these links are not shown in the figure.

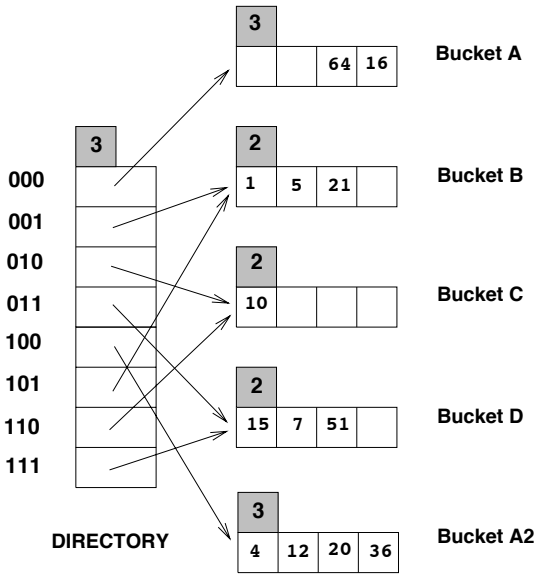


Answer the following questions.

1. Name all the tree nodes that must be fetched to answer the following query: "Get all records with search key greater than 38."
2. Show the B+ tree that would result from inserting a record with search key 109 into the tree.
3. Note that subtrees A, B, and C are not fully specified. Nonetheless, what can you infer about the contents and the shape of these trees?

Question 2 (1 points)

Consider the Extendible Hashing index shown in the figure below. Show the index after inserting an entry with hash value 68.



Question 3 (3 points)

Consider a relation R(a, b, c, d) containing 1 million records, where each page of the relation holds 10 records. R is organized as a heap file with unclustered indexes, and the records in R are randomly ordered. Assume that attribute *a* is a candidate key for R, with values lying in the range 0 to 999,999.

Consider the query: "find all R tuples such that a > 50 and a < 100".

Estimate the cost (in terms of disk I/O pages) of evaluating the query for each of the following approaches:

- Scanning through the whole heap file for R.
- Using a B+ tree index (alternative 2) on attribute R.a.
- Using a hash index (alternative 2) on attribute R.a.

Question 4 (4 points)

Suppose the external sorting algorithm is employed on file with 2,000,000 pages and 17 available buffer pages. Answer the following questions.

1. How many runs will the algorithm produce in the first pass?

2. How many passes will it take to sort the file completely?

3. What is the total I/O cost of sorting the file?

4. How many buffer pages do you need to sort the file completely in just two passes?

Question 5 (4 points)

Consider the join $R \bowtie_{R.a=S.b} S$, given the following information about the relations to be joined. The cost metric is the number of page I/Os unless otherwise noted, and the cost of writing out the result should be uniformly ignored.

Relation R contains 10,000 tuples and has 10 tuples per page. Relation S contains 2000 tuples and also has 10 tuples per page. Attribute b of relation S is the primary key for S. Both relations are stored as simple heap files.

Neither relation has any indexes built on it. 52 buffer pages are available.

1. What is the cost of joining R and S using a page-oriented simple nested loops join?

2. What is the cost of joining R and S using a block nested loops join?

3. What is the cost of joining R and S using a sort-merge join?

4. What is the cost of joining R and S using a hash join?

PART B: Programming

Setup

You are provided with:

- a VirtualBox[1] instance running Ubuntu, with PostgreSQL[2] and pgAdmin[3] installed,
- four (4) datasets, inside the aforementioned VirtualBox instance.

Note that the computers at the Paniikki Lab have VirtualBox already setup. Please use them for this assignment.

a. VirtualBox Instance

Once you login at a terminal, you can obtain the VirtualBox instance in the following manner from the command line.

```
> cp /scratch/mathiom1/moderndb-ubuntu.ova /scratch/your-username/
```

When you use VirtualBox to mount the instance (choosing *"Import Appliance"* from the menu), set the

```
hard disk controller (SATA) / virtual box image
```

attribute to point to **your scratch directory**.

```
hard disk controller (SATA) /virtual box image: /scratch/your-username
```

IMPORTANT!
When you use the Panikki Lab, do NOT store any
of the provided files under your home folder.
Instead, use the space under the /scratch/your-username directory.

If you prefer to open the VirtualBox instance on a different computer, please use the Paniikki Lab to access the dataset as described earlier.

The VirtualBox instance comes with Ubuntu and you can login with the following credentials.

username: **student**
password: **moderndb**

PostgreSQL is already installed for that account. You can access it as **student** (username/role) with password **moderndb**.

b. Datasets

You are provided with 4 datasets that contain data from IMDB (imdb.com). They are all stored in the following directory of the VirtualBox instance.

/home/student/moderndb/assignment1

[1] https://www.virtualbox.org/
[2] http://www.postgresql.org/docs/9.3/static/
[3] http://pgadmin.org/

There is one `small', one `medium', one `large', and one 'xlarge' dataset in that directory, each with its own .psql file. Once a dataset is loaded, the database should contain two tables: "actors" and "actresses", both with 4 columns, *id, name, movie, year*.

Below we describe how to load a dataset from the psql console.

To access the PostgreSQL console and connect to database `asg1db`, type the following on a terminal.

```
> psql asg1db -U student
```

To **load a dataset** into the database, type the following in the psql console.

```
asg1db=# \i path/to/dataset.psql
```

Note that loading one dataset will destroy any previous "actors" and "actresses" tables that existed in the database.

Question 6 (3 points)

Use the **'large'** dataset for this question. Write the query and the result for the following queries:

1. How many male actors were in the 2009 blockbuster movie 'Avatar'?
2. What is the year of the first appearance of actress Cate Blanchett? Notice that the name format is 'Blanchett,Cate'.
3. Find the total number of male actors who played in a movie with a title that contains the string 'Batman'.

Question 7 (12 points)

We are interested in counting the pairs of men and women (one man and one woman) that played together in multiple movies. In particular, we want to find the number of pairs that played together in at least 3 movies.

1. Write the appropriate query to find the number of pairs that played together in at least 3 movies.
2. Do *not* use any indexing structure for this question.
   For each dataset, provide a diagram for the execution plan used by PostgreSQL to execute your query and describe briefly the related access paths.
3. For the same setting as the question above (i.e., no index is built):
   Execute the query and report the time it takes to execute the query for each dataset.
4. You are asked to build either one (1) or (2) two indexes, on either or both of the relations (*actors* and *actresses*). The goal is to speed-up the execution time of your query on the '**large**' dataset. The index(es) are allowed to be clustered or unclustered.
   What index would you build? Justify your answer.
5. Build the index(es) of your choice, as specified in your answer above, for each dataset (*not just the large one*). For each dataset,
   a. determine whether PostgreSQL uses the index(es) to evaluate the query,
   b. provide the execution plan used by PostgreSQL to execute your query,

c. and describe briefly the related access paths.

6. For the same setting as the question above (i.e., the index(es) is (are) built): Execute the query and report the time it takes to execute the query for each dataset. How do the running times compare with those of q. (7.3)? Provide a brief explanation for differences in running times you observe.

Notes and Tips for Question (7):

- You do not have to report running times for queries that take longer than one (1) hour. If your query takes longer than that for a dataset, simply report "longer than 1hour" as its running time.
- You are encouraged to use pgAdmin to write your SQL queries, explore their execution plans, and execute them. However, you can also use the PostgreSQL command line (psql) if you wish to.
- Use the ANALYZE directive to instruct the dbms to collect statistics after you import a dataset or build/drop an index.

- Useful tips if you use the psql command line to write and execute your queries:
  - Type the command "\timing on" so that the execution time for each statement is reported at the end of its execution.
  - Type the command "\di" to list the indexes in the database.
  - You can use the EXPLAIN directive to obtain the execution plan for a query.

- Useful tips if you use the pgAdmin to write and execute your queries:
  - When you open pgAdmin, double-click to connect to the server listed as asg1 (localhost:5432).
  - Use the SQL editor (press button with label "SQL") to write queries, execute queries ("execute query" button in the editor window), or retrieve the execution plan for a query ("explain query" button).