Άσκηση 1 (10 μονάδες) External Sorting

Answer the following questions for each of these scenarios, assuming that external merge-sort is used to sort each of the files:

- A file with 10,000 pages and 3 available buffer pages.
- A file with 20,000 pages and 5 available buffer pages.
- A file with 2,000,000 pages and 17 available buffer pages.

A. (3 points) How many runs will you produce on the first pass?
B. (3 points) How many passes will it take to sort the file completely?
C. (4 points) What is the total I/O cost for sorting the file?

Άσκηση 2 (33 points) Query Execution

A relational database system holds three relations; C (corporations), E (executives) and S (stock sales) with the following characteristics:

- Relation C (corporations):
  - Tuples are stored as fixed length, fixed format records, length 6000 bytes.
  - There are 20,000 C tuples.
  - Tuples contain key attribute C.N (corporation number), length 20 bytes; other fields and record header make up rest.
  - There is an index on attribute C.N.

- Relation E (executives):
  - Tuples are stored as fixed length, fixed format records, length 2000 bytes.
  - There are 40,000 E tuples.
  - Tuples contain attribute E.N (the corporation that the executive works for), length 20 bytes; other fields and record header make up rest.
  - Tuples also contain attribute E.I (executive identifier), length 20 bytes.
  - There is an index on attribute E.N.

- Relation S (stock sales):
  - Tuples are stored as fixed length, fixed format records, length 1000 bytes.
  - There are 120,000 S tuples.
  - Tuples contain attribute S.I (the identifier of the executive involved), length 20 bytes; other fields and record header make up rest.
  - There is an index on attribute S.I.
While the number of executives associated with each corporation varies, for evaluation purposes we may assume that each corporation has 2 executives, and each executive has 3 stock sales records associated with him or her.
The records are to be placed in a collection of 16 Kilobytes (16384 bytes) disk blocks that have been reserved to exclusively hold C, E, or S records, or combinations of those records. (That is, there are no other types of records in the blocks we are discussing in this problem.) Each block uses 50 bytes for its header; records are not spanned.)

Three disk organization strategies are being considered:
1. **Sequential**: All the corporation (C) records are placed sequentially (ordered by corporation number) in one subset of blocks. Executive (E) records are separate in another set of blocks, executives are not ordered by corporation number. Finally, stock sales (S) records are in a third set of blocks, not ordered by executive identifier.
2. **Clustered**: For each corporation (C) record, the 2 executives for that corporation (C.N = E.N) reside in the same block. Similarly, the 6 stock sale records for those corporations are in the same block. The corporation records are not sequenced in any way.
3. **Random**: The records are placed as follows, without regard to C.N, E.N, E.I, S.I values, as follows: Each block contains one random C record, 2 random E records, and 6 random S records.

We are also told there are four types of queries that constitute the vast majority of the workload:
1. **Probe**: Given a corporation number, get the corporation record.
2. **Ordered scan**: For all corporations, in increasing corporation number order, get each corporation record.
3. **Plain scan**: For all corporations, in any order, get all corporations records.
4. **Join**: For a given corporation number C.N, get the corporation record followed by all its stock sales records. (That is, get all stock sales records with S.I = E.I, for any account with E.N = C.N)

**A. (13 points)** For each of the storage strategies, compute how many total disk blocks are needed for holding relations C, E and S. Briefly explain your answers.

**B. (20 points)** For each query type and for each storage strategy, estimate the number of disk blocks that must be transferred from disk to execute the query. Briefly explain each answer.

Assume you only have one buffer page (16 KB) in memory; thus, each time you need a block it counts as one IO (unless the request is for exactly the same block you already have in the buffer). You also have enough memory to hold a single working copy of a C record, of an E record, and of an S record. So, for example, to do a C, E join, you first read in a block containing a C record, and copy the record to the working C area. Then you read in the block containing an E record into your 16 KB buffer, and join the two records. Also, ignore any IOs due to index accesses. That is, you may assume the indexes reside in memory.
**Hint:** Do not concern yourself with events that are very unlikely. For example, say you want to get the three executive records for a particular corporation, and that these records are randomly placed on a large number of disk blocks. There is some chance that the executive records you want happen to be in the same block (just by luck). However, for this problem you may assume this probability is negligible and it will take you exactly three IOs to get those three records.

**Άσκηση 3 (28 points) Query Optimization**

Consider the following schema

- **Sailors(sid, sname, rating, age)**
- **Reserves(sid, did, day)**
- **Boats(bid, bname, size)**

Reserves.sid is a foreign key to Sailors and Reserves.bid is a foreign key to Boats.bid.

We are given the following information about the database:

- Reserves contains 10,000 records with 40 records per page.
- Sailors contains 1000 records with 20 records per page.
- Boats contains 100 records with 10 records per page.
- There are 50 values for `reserves.bid`.
- There are 10 values for Sailors.rating (1…10)
- There are 10 values for Boat.size
- There are 500 values for Reserves.day

Consider the following query

```sql
SELECT S.sid, S.sname, B.bname
FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid AND R.bid = B.bid AND
B.size > 5 AND R.day = 'July 4, 2003';
```

A. (4 points) Assuming uniform distribution of values and column independence, estimate the number of tuples returned by this query.

B. (4 points) Consider the following query

```sql
SELECT S.sid, S.sname, B.bname
FROM Sailors S, Reserves R, Boats B
```

Draw all possible left-deep query plans for this query.
C. **(20 points)** For the first join in each query plan (the one at the bottom of the tree, what join algorithm would work best? Assume that you have 50 pages of memory. There are no indexes, so indexed nested loops is not an option.

**Ασκηση 4 (10 points) Physical Schema Design**

Name three indexes on R (including at least one hash index) that match the predicate in the following SQL query and briefly explain why each index matches.

```sql
SELECT * FROM R
WHERE (B = "b" OR A <> "a") AND NOT (C <= 20 OR A <> "a")
```

**Ασκηση 5 (10 points) Database Statistics**

Consider the following schema.

<table>
<thead>
<tr>
<th>Table</th>
<th>Keys</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auctions (aid, minprice, description, seller, end_date)</td>
<td></td>
</tr>
<tr>
<td>Members (mid, nickname, name, since)</td>
<td></td>
</tr>
<tr>
<td>Bids (aid, buyerid, amount)</td>
<td></td>
</tr>
</tbody>
</table>

Assume there is an unclustered B-tree index on the key of each table. In answering questions, use the summary statistics functions that we learned about in class: NPages(), NTuples(), Low(), High(), NKeys(), IHeight(), INPages().

A. **(3 points)** Consider the query

```sql
SELECT 'found it!' FROM Members WHERE mid = 98765;
```

Given the information above, write a formula for the optimizer's lowest estimated cost for this query.

B. **(3 points)** Consider the query

```sql
SELECT * FROM Bids, Members
WHERE bids.buyerid = members.mid AND members.since < '2001';
```

Write the formula the optimizer would use for the selectivity of the entire WHERE clause.
C. (4 points) Consider the following query

```
SELECT R.*
FROM R, S, T
WHERE R.a = S.b
AND S.b = T.c
```

The following plans are generated during an intermediate pass of the System R optimizer algorithm. For each plan, write down the ordering column(s) of its output if any, and whether the plan would get pruned (P) or kept (K) at the end of the pass. If there is no clear ordering on the output, write “none”.

<table>
<thead>
<tr>
<th>PLAN</th>
<th>Cost in I/Os</th>
<th>Ordering Columns of Output</th>
<th>Prune or Keep</th>
</tr>
</thead>
<tbody>
<tr>
<td>IndexNestedLoops (</td>
<td>2010</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FileScan(R),</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IndexOnlyScan(Btree on S.b))</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SortMergeJoin(</td>
<td>3010</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FileScan(R),</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FileScan(S))</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BlockNestedLoops(</td>
<td>1010</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FileScan(R),</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FileScan(S))</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IndexNestedLoops(</td>
<td>30000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IndexOnlyScan(Btree on (R.d, R.a)),</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IndexScan(Btree on S.b))</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Άσκηση 6 (16 points) Logical Query Optimization

Draw the query tree structures for the following:

1. \( \pi_A((R \in \text{Join}_{B=C} S) \text{ Join}_{D=E} T) \)
2. \( \pi_A(\pi_E(T) \text{ Join}_{E=D} \pi_{ACD}(S) \text{ Join}_{C=B} R) \)

- Write down the sequence of steps needed to transform Query #1 to Query #2.
- List the attributes that each of the schemas R, S, and T must have and the attributes that each (or some) of these schemas must not have in order for the above transformation to be correct.