COSC-4411(A) Midterm #2

Sur / Last Name:
Given / First Name:
Student ID:

- **Instructor:** Parke Godfrey
- **Exam Duration:** 75 minutes
- **Term:** Fall 2004

Answer the following questions to the best of your knowledge. Be precise and be careful. The exam is open-book and open-notes. Write any assumptions you need to make along with your answers, whenever necessary.

There are four major questions. Points for each question and sub-question are as indicated. In total, the exam is out of 50 points.

If you need additional space for an answer, just indicate clearly where you are continuing.

Regrade Policy

- Regrading should only be requested in writing. Write what you would like to be reconsidered. Note, however, that an exam accepted for regrading will be reviewed and regraded in entirety (all questions).

Grading Box	
1.	/10
2.	/10
3.	/15
4.	/15
Total	/50

1. (10 points) **Join Algorithms.** *A pass too far.* [analysis]

   Dr. Bas has noted many queries involving joins that look as follows: $\mathsf{R} \bowtie_{\mathsf{J}} \mathsf{S}$ where $V_{\mathsf{R.J}} \ll V_{\mathsf{S.J}}$; that is, the number of distinct values of $\mathsf{R.J}$ is much less than the number of distinct values of $\mathsf{S.J}$. Furthermore, the set of distinct values of $\mathsf{R.J}$ is reasonably small, and could fit in main memory, say, within the space of 100 buffer frames. (This is *not* to say that R will fit in main memory. Assume that it does not.)

   a. (7 points) Design a variant of the 2-pass hash join algorithm that would be much more efficient than the usual hash join under these circumstances.

---

b. (3 points) Given $V_{\mathbf{R}.\mathsf{J}} = 10,000$, $V_{\mathbf{S}.\mathsf{J}} = 80,000$, $\mathbf{R}$ is 1,000 pages, $\mathbf{S}$ is 20,000 pages, and you have a quota of 150 buffer frames, how much does your hash join cost to evaluate $\mathbf{R} \bowtie_{\mathsf{J}} \mathbf{S}$?

2. (10 points) **General.** *Jan, ken, pon.* [multiple choice]

   For each of the following, choose *one* best answer.

---

a. (1 point) If the results of an operator in a query plan are *materialized*, what does this imply?

A. That it uses blocked I/O.
B. That no other operator can come after it in the query plan. (In other words, it must be the last operator in the tree.)
C. That it works on "blocks" of records at a time.
D. That it is a "blocking" operator that cannot be pipelined to the next operator. (So it must finish before the next operator can start.)
E. That it must be a join operator.

---

b. (1 point) Which of the following is true?

A. Anywhere a 2-pass hash join can be used, a 2-pass sort-merge join could be used instead, at the same cost.
B. A block-nested-loops join is never better than a hash join or a sort-merge join. It is part of the repertoire simply for the cases when hash join and sort-merge join cannot be applied.
C. Select conditions can only be evaluated by use of an index.
D. Index-nested-loops join will rarely be the best choice unless there are other selective—returning few results—conditions in the query.
E. Hash joins and sort-merge joins cannot be used together in the same query plan.

---

c. (1 point) Using replacement sort instead of quicksort for pass zero of the external sort algorithm has the advantage that

A. it is faster than quicksort (even though they are the same $\mathcal{O}$-wise).
B. it allows for block reads whereas quicksort does not.
C. it produces runs twice as long, on average, as the use of quicksort does.
D. it reduces the number of I/O's for pass zero (compared with using quicksort).
E. it reduces the number of I/O's of each subsequent (merge) pass (compared with using quicksort in pass zero).

---

d. (1 point) Using replacement sort instead of quicksort for pass zero of the external sort algorithm has the following disadvantages, *except* for

A. it is slower than quicksort.
B. it does not permit block reads to the extent that using quicksort does.
C. it is difficult to implement efficiently data-structure-wise compared with quicksort.
D. its performance can diminish if the input is already sorted.
E. it can increase the number of I/O's for pass zero (compared with using quicksort).

---

e. (1 point) Consider table **R** with attributes A and B, table **S** with attributes A and B. All joins below are natural joins; that is, the join columns are those columns named the same between the two tables.

I. $\sigma_{A \geq 5}(\mathbf{R} \bowtie \mathbf{S})$
II. $\sigma_{A \geq 5}(\mathbf{R}) \bowtie \mathbf{S}$
III. $\sigma_{A \geq 5}(\mathbf{R}) \bowtie \sigma_{A \geq 5}(\mathbf{S})$

Which of the above relational algebra expressions necessarily evaluate to the same result?

A. All three must evaluate to the same result.
B. They each may evaluate to a different result.
C. **I** & **II**
D. **I** & **III**
E. **II** & **III**
F. Not enough information is provided to determine this.

---

f. (1 point) Which of the following would not be considered a benefit of indexing?

A. To improve performance during data sorting.
B. To ensure uniqueness of key values.
C. To avoid reading the records when processing queries that retrieve only indexed columns.
D. To improves performance during large sequential table scans.
E. To help the query optimizer in cost estimation.

---

g. (1 point) Which following heuristic tends to result often in a less expensive query plan?

A. Preferring filescans to using an unclustered index.
B. Making the larger table of the two in a join the outer table.
C. Making the smaller table of the two in a join the outer table.
D. Evaluating all selects on-the-fly.
E. Performing selects at the bottom of the tree before other operations.

---

h. (1 point) If a query involves `group by`, then

A. the query plan cannot employ index nested loops joins.
B. it will be much less expensive than the same query but without the `group by`, because fewer records are returned.
C. the data will need to be partitioned or sorted.
D. the tables involved will have to be scanned repeatedly.
E. index-only plans are not possible.

---

i. (1 point) Consider the two variants of the same query: "`select distinct ...`" and "`select all ...`".

A. The *distinct* version is guaranteed to run faster.
B. The *all* version is guaranteed to run faster.
C. The two versions are guaranteed to run in the same time, since the query plan in either case must sort.
D. The *distinct* version is guaranteed to run faster if there is a join involved; otherwise, the *all* version will run faster.
E. None of the above.

---

j. (1 point) Unclustered tree indexes are not useful

A. to replace sorting, when the records need sorting by the index's search key.
B. for index-only plans.
C. for use in index-intersection.
D. in index-nested loop joins for foreign-key joins with the parent table as the inner.
E. as access paths for selective `select` conditions.

3. (15 points) **Reduction factors, etc.** *Reduce today!* [short answer & exercise]

a. (3 points) Consider the relation **employee**(id, name, phone, salary, gender). On which attribute would it be least useful to build an index? Briefly, why?

b. (3 points) Consider tables $\mathbf{R}(\mathsf{A}, \mathsf{B})$, $\mathbf{S}(\mathsf{B}, \mathsf{C})$, and $\mathbf{T}(\mathsf{C}, \mathsf{D})$. What would be a reasonable estimate for the result size (number of records) of the expression

$$\sigma_{\mathsf{A}=10}((\mathbf{R} \bowtie_{\mathsf{B}} \mathbf{S}) \bowtie_{\mathsf{C}} \mathbf{T})$$

if the only available statistics are the numbers of records in the table—$N_{\mathbf{R}}$, $N_{\mathbf{S}}$, and $N_{\mathbf{T}}$—and the distinct value counts for the attributes—$V_{\mathbf{R}.\mathsf{A}}$, $V_{\mathbf{R}.\mathsf{B}}$, $V_{\mathbf{S}.\mathsf{B}}$, $V_{\mathbf{S}.\mathsf{C}}$, $V_{\mathbf{T}.\mathsf{C}}$ and $V_{\mathbf{T}.\mathsf{D}}$.

c. (4 points) Name two distinct reasons why complex queries with deep query trees are difficult to optimize.

d. (5 points) File **F** is 400 pages and 5 records fit per page. You have a buffer-pool quota (B) of 6 frames.

Calculate the I/O cost of sorting **F** using the basic external sort routine. (Assume that pass 0 produces runs of size B, so 6 in this case.)

4. (15 points) **Query Evaluation.** *What* do *polysci students take?!* [exercise / analysis]

   Consider tables

   - **student**(id, name, major) with 100,000 records on 2,000 pages
   - **enrol**(id, course#, section, term, grade) with 4,000,000 records on 40,000 pages

   There is a foreign key from **enrol** onto **student**.

   Available indexes:

   - hash index on **student** on id (linear hash)
   - clustered tree index on **enrol** on id, course#, section, term (index pages are 3 deep)
   - unclustered tree index on **enrol** on course#, id (index pages are 3 deep)

   Statistics:

   - the number of values of **student**.major: 100
   - values of **enrol**.course#: 1000, ..., 4999 (so 4000 values)

   Consider the query

```
select name, S.id, course#, section, term, grade
    from student S, enrol E
    where S.id = E.id
      and course# is between 4000 and 4999
      and major = 'political science';
```

   You have an allocation of 25 buffer frames.

   a. (3 points) Estimate the cardinality of the query.

b. (7 points) Devise a good query plan for the query. Show the query tree, *fully* annotated with the chosen algorithms and access paths.

Estimate the cost of your plan. For full credit, you should have a plan that costs less than 10,000 I/O's.

c. (5 points) Consider the query

```
select name, S.id, course#, section, term, grade
    from student S, enrol E
    where S.id = E.id
      and course# is between 4000 and 4999;
```

This is the same as before, but with the "`major = 'political science'`" condition dropped.

Show a good query plan (annotated tree) for this query. (By good, I mean better than picking just any naïve approach.) What is its cost?

(Scratch space.)

(Scratch space.)

(Scratch space.)

RELAX. TURN IN YOUR EXAM. GO HOME.