Name:_____

# CSE 444, Spring 2009, Final Examination
# 11 June 2009

Rules:

- Open books and open notes.
- No laptops or other mobile devices.
- Please write clearly.
- Relax! You are here to learn.

| Question | Max | Grade |
|---|---|---|
| 1 | 25 | |
| 2 | 15 | |
| 3 | 25 | |
| 4 | 15 | |
| 5 | 10 | |
| 6 | 10 | |
| Total | 100 | |

Initials:_____

1. (**25** points) **SQL**

   Consider a database with the following schema. This database records information about chefs (including their name and their rating), and the dishes they prepare. For each dish, the database records its name and its popularity score.

```
Chef(cid, name, rating)        /* cid is a chef's unique identifier */
Dish(did, name, popularity)    /* did is a dish's unique identifier */
Prepares(cid, did)             /* cid references Chef.cid and did references Dish.did */
```

   (a) (**7** points) Write a SQL query that lists the names of the dishes prepared by only one chef. Your query should return a list of dish names.

   **Solution:**

```
SELECT   d.name
FROM     Dish d, Prepares p
WHERE    d.did=p.did
GROUP BY d.did, d.name
HAVING   COUNT(p.cid)=1
```

Initials:_____

(b) (**8** points) Write a SQL query that finds the names of the chefs with above average rating and the names of the dishes with above average popularity that they prepare. Your query should return pairs of highly-rated chef name and popular dish name.

**Solution:**

```
SELECT c.name, d.name
FROM   Chef c, Dish d, Prepares p
WHERE  p.cid = c.cid AND p.did = d.did
AND c.rating > ( SELECT AVG(c2.rating) FROM Chef c2)
AND d.popularity > ( SELECT AVG(d2.popularity) FROM Dish d2)
```

(c) (**10** points) Write a SQL query that finds, for each chef, the most popular dish that he or she prepares. Your query should output the chef's name along with the name of his/her most popular dish. If there is more than one most popular dish, your query should output them all.

**Solution:**

```
SELECT c.name, d.name
FROM   Chef c, Dish d, Prepares p
WHERE  p.cid = c.cid AND p.did = d.did
AND    d.popularity >= ALL (SELECT d2.popularity
                            FROM Dish d2, Prepares p2
                            WHERE p2.cid = p.cid AND p2.did=d2.did)
```

Initials:_____

2. (**15** points) **Transactions**

Consider a concurrency control manager by timestamps. Below are several sequences of events, including start events, where sti means that transaction Ti starts and coi means Ti commits. These sequences represent real time, and the timestamp-based scheduler will allocate timestamps to transactions in the order of their starts. In each case below, say what happens with the last request.

You have to choose between one of the following four possible answers:

(a) the request is accepted,

(b) the request is ignored,

(c) the transaction is delayed,

(d) the transaction is rolled back.

(a) st1; st2; r1(A); r2(A); w1(B); w2(B);

**Solution:**
The system will perform the following action for w2(B): accepted.

(b) st1; st2; r2(A); co2; r1(A); w1(A)

**Solution:**
The system will perform the following action for w1(A): rolled back.

(c) st1; st2; st3; r1(A); w3(A); co3; r2(B); w2(A)

**Solution:**
The system will perform the following action for w2(A): ignored.

(d) st1; st2; st3; r1(A); w1(A); r2(A);

**Solution:**
The system will perform the following action for r2(A): delayed.

(e) st1; st2; st3; r1(A); w2(A); w3(A); r2(A);

**Solution:**
The system will perform the following action for r2(A): rolled back.

Initials:_____

3. (**25** points) **Query Processing**

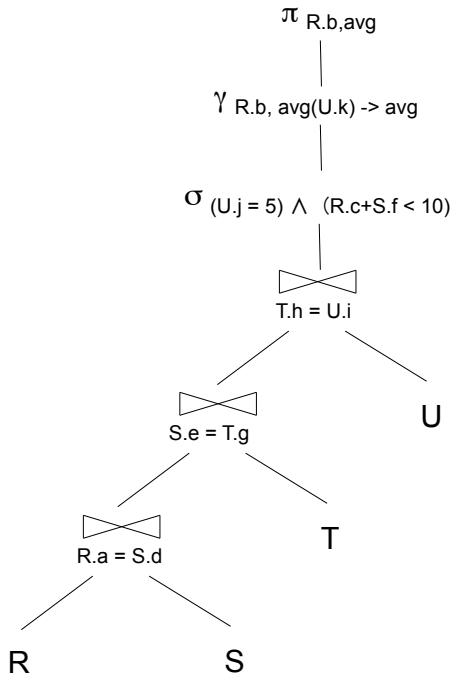
Consider four tables R(a,b,c), S(d,e,f), T(g,h), U(i,j,k).

(a) (**5** points) Consider the following SQL query:

```
SELECT   R.b, avg(U.k) as avg
FROM     R, S, T, U
WHERE    R.a = S.d
         AND S.e = T.g
         AND T.h = U.i
         AND U.j = 5
         AND (R.c + S.f) < 10
GROUP BY R.b
```

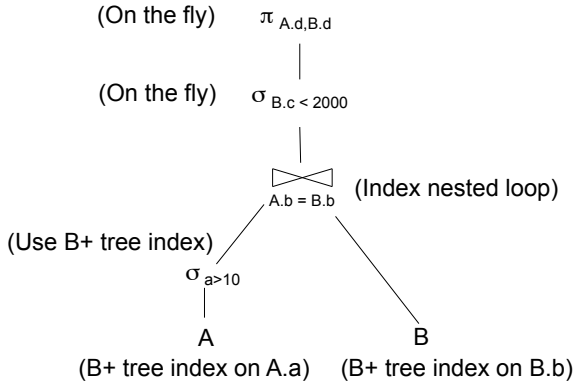
Draw a *logical* plan for the query. You may chose any plan as long as it is correct (i.e. no need to worry about efficiency).

**Solution:**
Many solutions were possible including:

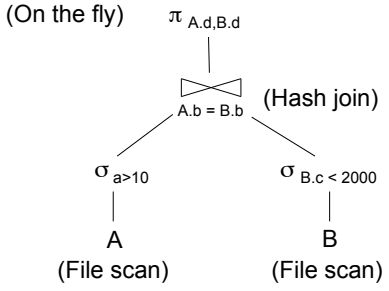


Initials:_____

(b) (**10** points) Consider the following two physical query plans. Give **two** reasons why plan B may be faster than plan A. **Explain** each reason.




Plan A




Plan B

**Solution:**
The following are three possible reasons why plan B could be faster than plan A:

- The low selectivity of the selection predicate ($a > 10$) and an unclustered index on A.a may make a file scan of A faster than using the index.
- If there are lots of matches, hash joins may be faster than index nested loops. The hash join reads its input only once (unless the input is too large). The index-nested loop may end-up reading the same pages of B multiple times.
- Pushing the selection ($B.c < 2000$) down can get rid of lots of B tuples before the join, reducing the cost of that operation.

Initials:_____

(c) (**10** points) Explain how two relations R and S can be joined together using a two-pass partitioned (Grace) hash join algorithm. In your explanation, use the following facts: R has 90 pages. S has 80 pages. There are 11 pages of memory. Please provide a detailed explanation that includes (1) goal of each step, (2) how many pages are allocated as input buffer(s), (3) how they are used, (4) how many pages are allocated as output buffer(s), (5) how they are used, etc. You can assume that hash tables have no overhead (a hash table for a relation that has 9 pages will require 9 pages of memory). You can assume a uniform data distribution.

**Solution:**
First, we split R into partitions:

- Allocate one page for the input buffer.
- Allocate 10 pages for the output buffers: one page per partition.
- Read in R one page at the time. Hash into 10 buckets. As the pages of the different buckets fill-up, write them to disk. Once we process all of R, write remaining incomplete pages to disk. At the end of this step, we have 10 partitions of R on disk. Assuming uniform data distribution, each partition comprises 9 pages.

Then, we split S into partitions the same way we split R (must use same hash function).

For each pair of partitions that match:

- Allocate one page for input buffer
- Allocate one page for the output buffer.
- Read one 9-page partition of R into memory. Create a hash table for it using a different hash function than above.
- Read corresponding S partition into memory one page at the time. Probe the hash table and output matches.

Initials:_____

4. (**15** points) **XML/XPath/XQuery**

Consider an XML instance having the following DTD:

```
<!DOCTYPE university [
<!ELEMENT university (student)* >
<!ELEMENT student (id, name, major?, course+)>
<!ELEMENT course (number, name)>
<!ELEMENT id (#PDCATA) >
<!ELEMENT name (#PDCATA) >
<!ELEMENT major (#PDCATA) >
<!ELEMENT number (#PDCATA) >
]>
```

Element `id` is a unique identifier for students and `number` is a unique identifier for courses.

(a) (**1** points) Is the following XML document valid given the above DTD (ignoring headers)? Please answer true or false.

```
<university>
    <student>
        <id>12345</id>
        <name>Bob</name>
        <course>
            <number>444</number>
            <name>db</name>
        </course>
        <course>
            <number>451</number>
            <name>os</name>
        </course>
    </student>
    <student>
        <id>67890</id>
        <name>Lidia</name>
        <major>CSE</major>
        <course>
            <number>444</number>
            <name>db</name>
        </course>
    </student>
</university>
```

**Solution:**
TRUE

Initials:_____

(b) (**1** points) Is the following XML document valid given the above DTD (ignoring headers)? Please answer true or false.

```
<university>
</university>
```

**Solution:**
TRUE

(c) (**1** points) Is the following XML document valid given the above DTD (ignoring headers)? Please answer true or false.

```
<university>
    <student>
        <id>12345</id>
        <name>Bob</name>
        <major>Math</major>
    </student>
    <student>
        <id>67890</id>
        <name>Lidia</name>
        <major>CSE</major>
        <course>
            <number>444</number>
            <name>db</name>
        </course>
    </student>
</university>
```

**Solution:**
FALSE

Initials:_____

(d) (**5** points) Write an XPath expression that computes the name of all courses taken by at least one Math major. Do NOT worry about duplicates. Your answer should only return values such as:

```
444
451
444
...
```

Note that you have to write an XPath expression, not an XQuery expression.

**Solution:**

```
/university/student[major='Math']/course/name/text()
```

Initials:_____

(e) (**7** points) Write an XQuery expression that reformats a valid XML document as per the above DTD into one that matches the following DTD:

```
<!DOCTYPE university [
<!ELEMENT university (course)* >
<!ELEMENT course (number, student*)>
<!ELEMENT student (id)>
<!ELEMENT id (#PDCATA) >
<!ELEMENT number (#PDCATA) >
]>
```

Courses should occur uniquely. They should include all registered students.

**Solution:**

```
<university> {
   for $d in document("university.xml")/university
   for $c in distinct-values($d/student/course/number/text())
   return <course>
            <number> { $c } </number>
            { for $s in $d/student[course/number=$c]/id
              return <student> {$s} </student>
            }
          </course>
}
</university>
```

Initials:_____

5. (**10** points) **Parallel Query Processing**

Olivia is a famous astronomer at the University of Washington. As part of her research, Olivia runs very large simulations of the universe. These simulations produce massive datasets that Olivia analyzes by executing complex queries. Olivia just ran a simulation that produced 100 GB of data. Her query took 1,000 minutes to run on this dataset using a single machine.

What would it mean for a parallel DBMS (or a system like Pig/MapReduce) to *speedup* Olivia's analysis task? How about to *scaleup* her task? What would it mean for the speedup and scaleup to be linear? Make sure your answer discusses **both** speedup and scaleup.

**Solution:**
A parallel DBMS could speedup or scaleup Olivia's analysis task by partitioning the data and query operators across many machines, such that each machine processes a subset of the data yielding either a shorter overall query latency for a given input data size (speedup) or the same query latency for a larger input data size (scaleup) compared to a single machine.

With a linear speedup, the latency would improve proportionally to the number of machines. If it takes 1,000 minutes with 1 machine, it should take 100 minutes with 10 machines, 10 minutes with 100 machines, and 1 minute with 1,000 machines.

With a linear scaleup, if it takes 1,000 minutes to process 100 GB on one machine, it should take the same 1,000 minutes to process 1TB using 10 machines, 10 TB using 100 machines, etc.

Initials:_____

6. (**10** points) **Databases as a Service**

Frank and Betty own a small Internet based company that sells collector pens over the web. Recently, they decided to get rid of all their infrastructure and move to using Amazon Web Services. As part of this transformation, they plan to get rid of their DBMS and run their application on top of Amazon SimpleDB.

(a) (**5** points) List three potential benefits of this move. **Explain** each benefit.

**Solution:**

Many answers were possible for this question including:

i. Frank and Betty will no longer need to administer their DBMSs themselves.

ii. Frank and Betty will not have to worry about equipment upgrades or purchases. SimpleDB will provide them elastic scalability and a "pay-as-you-go" pricing model.

iii. Amazon will worry about operations problems such maintaining high-availability.

(b) (**5** points) List three potential challenges that they will face. **Explain** each challenge.

**Solution:**

Again, many answers were possible including:

i. SimpleDB does not have all the features of a DBMS: there are no transactions, the query model is limited, etc.

ii. Frank and Betty will need to rewrite their applications to use SimpleDB.

iii. They may worry about data privacy now that the data will be stored on Amazon's servers.