Name: _____

CSE 444, Winter 2011, Final Examination

17 March 2011

Rules:

- Open books and open notes.
- No laptops or other mobile devices.
- Please write clearly and explain your reasoning
- You have 1 hour 50 minutes; budget time carefully
- An extra page is provided in case you run out of space, but make sure to put a forward reference.

Question	Max	Grade
1	27	
2	20	
3	34	
4	14	
5	5	
Total	100	
Extra credit	5	

Initials: _____

1. (**27** points) **SQL and Relational Algebra**

   Consider the following schema:

   ```
   Vehicle ( VIN, model, year )
   Driver ( license, name, age )
   Insured ( license, VIN, premium )
   ```

   The key fields are underlined: `VIN` is the key for `Vehicle`, `license` is the key for `Driver`, and `VIN` and `license` together form the key for `Insured`. Also, `Insured.license` is a foreign key that references `Driver.license`, and `Insured.VIN` is a foreign key that references `Vehicle.VIN`.

   (a) (**22** points) Write the following queries in Relational Algebra and SQL. For the RA part, you can give either the relational algebra expression, or the tree representation.

   i. (**6** points) Find the VINs of vehicles that are insured for a driver between 20 and 30 years of age.

   Relational Algebra:

   $\pi_{VIN}((\pi_{license}\sigma_{age>20 \wedge age<30} Driver) \bowtie Insured)$

   or

   $\pi_{VIN}(\sigma_{age>20 \wedge age<30}(Driver \bowtie Insured))$

   SQL:

   ```
   SELECT  I.VIN
   FROM    Insured I, Driver D
   WHERE   D.age > 20 AND D.age < 30 AND D.license = I.license
   ```

Initials: _____

ii. (**8** points) Find the VINs of vehicles that are insured for some driver under 25 years of age and another driver who is over 50 years of age.

Relational Algebra:

$\pi_{VIN}((\pi_{license}\sigma_{age<25}Driver) \bowtie Insured) \bigcap \pi_{VIN}((\pi_{license}\sigma_{age>50}Driver) \bowtie Insured)$

or
$\pi_{VIN}(\sigma_{age<25}(Driver \bowtie Insured)) \bigcap \pi_{VIN}(\sigma_{age>50}(Driver \bowtie Insured))$

SQL:

```
SELECT  I1.VIN
FROM    Insured I1, Driver D1, Insured I2, Driver D2
WHERE   D1.age < 25 AND D1.license = I1.license
  AND   D2.age > 50 AND D2.license = I2.license
  AND   I1.VIN = I2.VIN
```

or

```
SELECT  I.VIN
FROM    Insured I, Driver D
WHERE   D.age < 25 AND D.license = I.license
        AND EXISTS (SELECT  D2.license
                    FROM    Driver D2, Insured I2
                    WHERE   D.age > 50 AND I2.VIN = I.VIN
                            AND D2.license = I2.license)
```

Initials: _____

iii. (**8** points) Some vehicles are operated by more than one driver, and a different premium may be charged for each driver for the same vehicle. Find pairs of license numbers such that the driver with the first license number gets charged a higher premium for the same vehicle compared to the driver with the second license number.

Relational Algebra:

$\rho(I1, Insured)$
$\rho(I2, Insured)$
$\pi_{I1.license, I2.license}(\sigma_{I1.VIN=I2.VIN \wedge I1.license \neq I2.license \wedge I1.premium > I2.premium}(I1 \times I2))$

SQL:

```
SELECT  I1.license, I2.license
FROM    Insured I1, Insured I2
WHERE   I1.VIN = I2.VIN
  AND   I1.license <> I2.license
  AND   I1.premium > I2.premium
```

Alternatively, "`I1.license <> I2.license`" can be left away because of the key `(license,VIN)` and comparison on `premium`:

```
SELECT  I1.license, I2.license
FROM    Insured I1, Insured I2
WHERE   I1.VIN = I2.VIN
  AND   I1.premium > I2.premium
```

Initials: _____

(b) (**5** points) For each one of the Relational Algebra (RA) expressions that follow, state what they compute. Simply select one of the statements below that correctly describes the output produced by each expression. **Different RA expressions may map to the same statement**, but one RA expression cannot correspond to multiple statements. For your convenience, the tree representations of the RA expressions are given on the next page.

Statements:

i. The models of vehicles insured by a driver under 25 years old for a premium of less than \$200, and another driver who is over 50 years old, with a premium of less than \$150.

ii. The models of vehicles insured by drivers under 25 years old for a premium of less than \$200.

iii. The models of vehicles insured by a driver under 25 years old for a premium of less than \$200, or a driver who is over 50 years old, with a premium of less than \$150.

iv. The RA expression returns no tuples.

v. None of the statements i-iv correctly describes the output of the RA expression.

RA1: $\pi_{model}(\pi_{VIN}((\sigma_{age<25}Driver) \bowtie (\sigma_{premium<200}Insured)) \bowtie Vehicle)$

is described by statement: ii

RA2: $\pi_{model}(\pi_{VIN}((\sigma_{age<25}Driver) \bowtie (\sigma_{premium<200}Insured) \bowtie Vehicle))$

is described by statement: iv

RA3: $\pi_{model}(\sigma_{age<25 \wedge premium<200}((\pi_{VIN}(Driver \bowtie Insured)) \bowtie Vehicle))$

is described by statement: iv

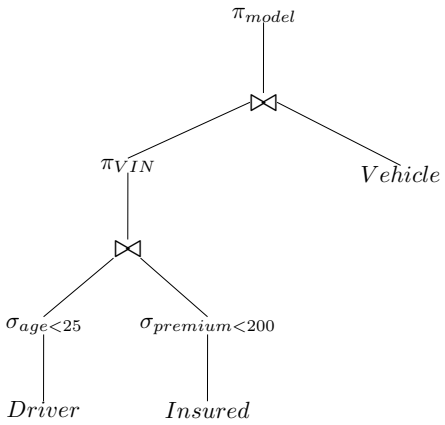
RA4: $(\pi_{model}((\sigma_{age<25}Driver) \bowtie (\sigma_{premium<200}Insured) \bowtie Vehicle)) \bigcup (\pi_{model}((\sigma_{age>50}Driver) \bowtie (\sigma_{premium<150}Insured) \bowtie Vehicle))$

is described by statement: iii

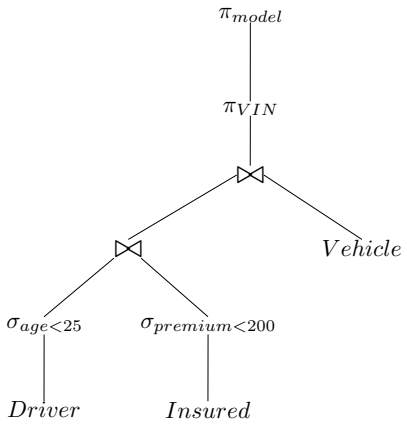
RA5: $\pi_{model}((\pi_{VIN,model}((\sigma_{age<25}Driver) \bowtie (\sigma_{premium<200}Insured) \bowtie Vehicle)) \bigcup (\pi_{VIN,model}((\sigma_{age>50}Driver) \bowtie (\sigma_{premium<150}Insured) \bowtie Vehicle))))$

is described by statement: iii

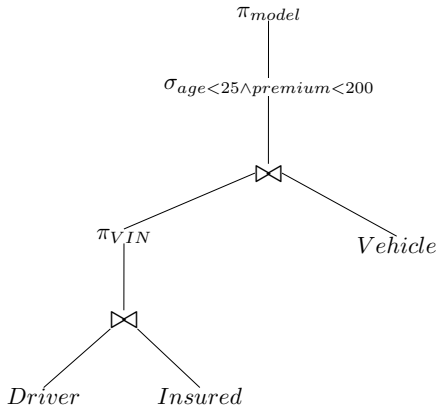
Initials: _____



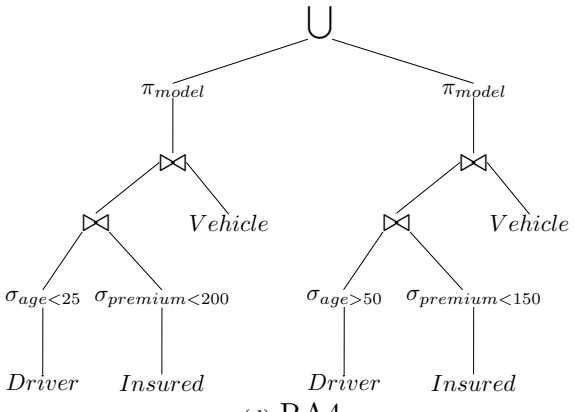
(a) RA1



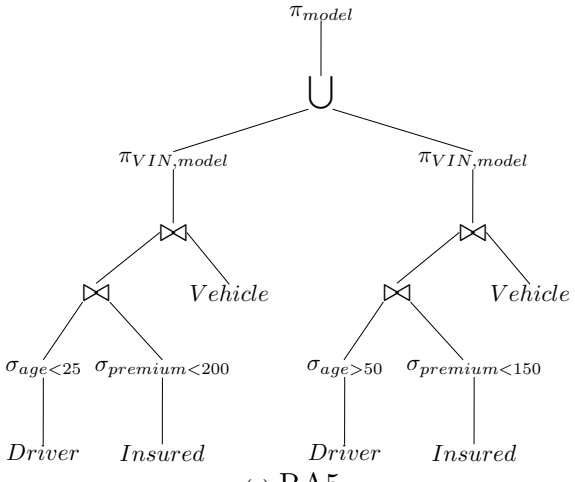
(b) RA2



(c) RA3



(d) RA4



(e) RA5

Initials: _____

2. (**20** points) **Distributed Transactions**

Assume a distributed transaction T that was submitted at site 0 (coordinator), and spawned subtransactions at sites 1, 2, and 3 (subordinates). We describe the sequences of messages that can take place during the 2 Phase Commit Protocol as follows: Let `(i, j, M)` denote that site `i` sends message `M` to site `j`. The value of `M` can be one of the following:

```
P (prepare)
C (commit)
A (abort)
Y (yes vote, ready to commit)
N (no vote, do not commit).
```

For example, the message `(0, 1, P)` denotes a prepare message send from the coordinator to the first subordinate.

We ignore ack messages.

(a) (**2** points) Describe the sequence of messages send for a successful commit during the 2 phases of the 2PC protocol. At what point exactly is the transaction considered committed?

`(0,1,P), (0,2,P), (0,3,P), (1,0,Y), (2,0,Y), (3,0,Y), (0,1,C), (0,2,C), (0,3,C)`

The Xact is considered committed when a commit record is force-written by the coordinator, just before sending commit messages to the subordinates.

(b) (**2** points) Describe the sequence of 2PC messages in case site 3 does not want to commit the transaction. Underline any messages that can be omitted as an optimization.

`(0,1,P), (0,2,P), (0,3,P), (1,0,Y), (2,0,Y), (3,0,N), (0,1,A), (0,2,A), (0,3,A)`

Initials: _____

(c) (**4** points) Suppose that the coordinator has sent all the prepare messages but has not yet received a vote from site 1. Would it be ok for the coordinator to abort the transaction at this point, and send abort messages to the subordinates? Why, or why not?

Yes. No site committed the transaction, and no commit messages have been sent.

(d) (**4** points) Suppose that the coordinator has sent all the prepare messages, received a No vote from site 1, but has not yet received the votes of sites 2 and 3. Should the coordinator wait for the two missing votes, or should it proceed to abort the transaction and why?

The coordinator should proceed to abort the transaction, as any No vote guarantees abort.

(e) (**4** points) Suppose that site 1 has received a prepare message and voted Yes, but has not received any commit or abort messages. Site 1 contacts site 2 and discovers that site 2 has received a commit message. Is it ok for site 1 to commit the transaction? Why, or why not?

Yes. If any site has received a commit message, the Xact is committed at the coordinator site.

(f) (**4** points) Suppose that site 1 has received a prepare message and voted Yes, but has not received any commit or abort messages. Site 1 contacts all other subordinates and discovers that they have all voted Yes to the coordinators Prepare message. Is it ok for site 1 to commit the transaction? Why, or why not?

No. The coordinator may have crashed, or timed-out waiting for a yes vote and can choose to unilaterally abort the Xact.

Initials: _____

3. (**34** points) **Query Optimization**

Consider the following schema:

```
Guitars ( gid, brand, price )
Players ( pid, name, age )
LastPlayed ( gid, pid, date )
```

Here, `LastPlayed.gid` is a foreign key that references `Guitars.gid`, and `LastPlayed.pid` is a foreign key that references `Players.pid`. Consider the following query:

```
SELECT   P.name
FROM     Guitars G, Played P, LastPlayed L
WHERE    G.gid = L.gid AND P.pid = L.pid
  AND    P.age <= 25 AND G.brand = 'Gibson'
  AND    G.price >= 3000;
```

Further assume that the data is evenly distributed, and that the following statistics hold:

`Guitars.gid` has 1,000 distinct values
`Guitars.brand` has 15 distinct values
`Guitars.price` ranges from 1,000 to 4,999
`Players.pid` has 15,000 distinct values
`Players.age` ranges from 11 to 85

(a) (**6** points) Compute the selectivity for each individual term in the where clause, then provide a final selectivity estimate for what fraction of the total tuples will appear in the output. Calculate each of the 6 values below:

`G.gid = L.gid` : $1/1,000$

`P.pid = L.pid` : $1/15,000$

`P.age <= 25` : $15/75 = 1/5$

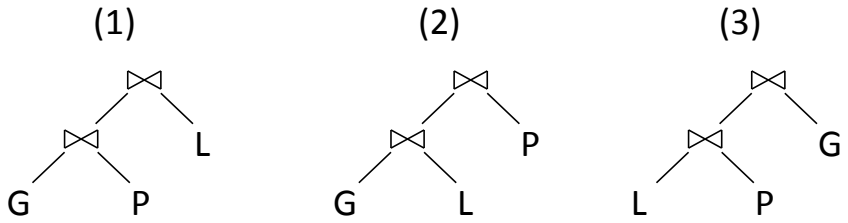
`G.brand = 'Gibson'` : $1/15$

`G.price >= 3000` : $2,000/4,000 = 1/2$

TOTAL : $1/(1,000 * 15,000 * 5 * 15 * 2) = 1/(2,25 * 10^9)$

Initials: _____

(b) (**2** points) In the following picture you see three possible linear plans (they are all left-deep). Which of the shown query plans would not be considered by any reasonable query optimizer and why? Circle the plan(s) that would not be considered and explain your reasoning below.



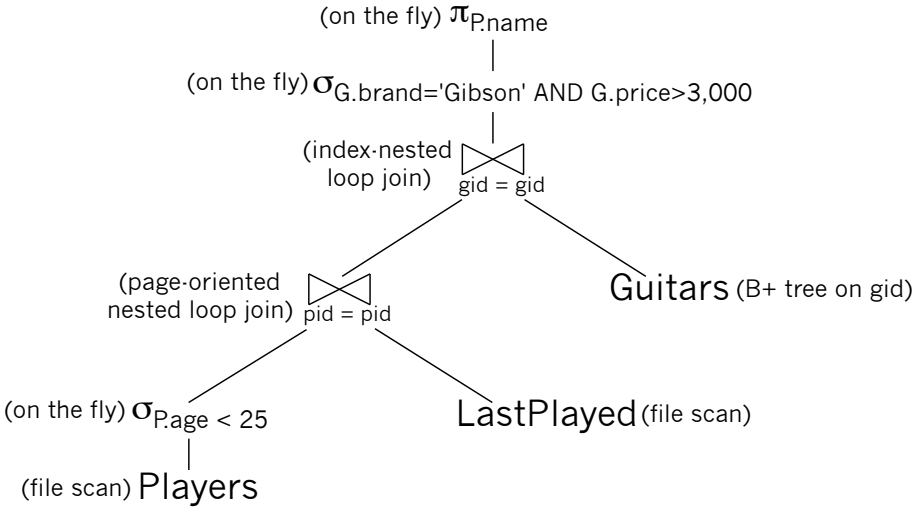
The first plan would not be considered. Since G and P share no join predicate, the first plan would require a cross product.

Initials: _____

(c) (**16** points) Next assume there is an unclustered B+ tree index on `Guitars.gid` which is kept in main memory, and consider the following statistics for each of the 3 relations:

`Guitars`: 40 bytes/tuple, 100 tuples/page, size: 10 pages
`Players`: 80 bytes/tuple, 50 tuples/page, size: 300 pages
`LastPlayed`: 25 bytes/tuple, 150 tuples/page, size: 90,000 pages

Consider the query plan below. Calculate the number of I/Os, step-by-step. Make your calculations explicit so you can get partial credit. Remember that for a page-oriented nested loop, only one single page of the build relation is kept in main memory while iterating over the probe relation. Assume the main memory has clearly space for more than 2 pages.



- Scan of Players creates 300 I/Os.
- Selection `Page<25` leaves tuples that fit into $300 * 1/5 = 60$ pages. No additional cost.
- The page-oriented nested loop join iterates 60 times over the probe relation (one page at a time). Cost: $60 * 90,000 = 5.4 * 10^6$ I/Os.
- The number of tuples after the nested loop join on the foreign key is the number of tuples in `LastPlayed` $(90,000 * 150)$ times the selection 1/5 on `Players`, hence $90,000 * 30 = 2.7 * 10^6$.
- For each such tuple, the index-nested loop join retrieves exactly one page due to the foreign key join. Cost: $2.7 * 10^6$ I/Os.
- Total cost: $300 + (5.4 + 2.7) * 10^6 \approx 8.1 * 10^6$ I/Os.

Initials: _____

(d) (**10** points) Consider the plan from part (c). For each of the following 5 changes to it, answer "Y" (yes) or "N" (no) depending on whether it could have led to a further reduction in the number of I/Os. Shortly explain your reasoning. Consider each change individually!

1. Pushing down the selection on `G.brand` and `G.price` below the join:

N: This would not reduce the number of I/Os. Each index look-up of the index-nested loop would retrieve one tuple from `Guitars`, which would then be either selected and joined or not. Since the results of the join are pipelined, there is no difference in I/Os.

2. Creating a temporary file to store the results of the selection on `P.age`:

N: This would increase the number of I/Os as we would materialize intermediate results instead of pipelining them.

3. Have the index on `Guitars.gid` be clustered:

N: Each index look-up retrieves only one single tuple (due to the foreign key join). Hence, we need to retrieve exactly one page either way for each lookup.

4. Projecting out `Players.age` and `LastPlayed.date` before the join.

Y: By projecting out attributes that are not used later, we can fit more tuples into the single block that is used for the build relation in the page-oriented nested loop join. Hence, fewer iterations over `LastPlayed`.

5. Changing the first join to block-oriented nested loop join:

Y: Block-oriented nested loop join may allows to fit more tuples of the build relation into main memory and iterate fewer times over `LastPlayed`.

Initials: _____

4. (**14** points) **Parallel Databases**

Consider the join $R(x, y) \bowtie S(y, z)$ between two relations, where R and S occupy 1000 and 300 blocks, respectively.

(a) (**4** points) What is the minimum cost in disk I/Os that is needed to join the relations using a block-nested loop join on a machine with 101 blocks of main memory?

We can use 100 blocks of memory to buffer the smaller relation S. We thus read S in 3 chunks of size 100, and iterate 3 times over the larger relation R. The total cost is $B(S) + 3 * B(R) = 3300$ I/Os.

(b) (**6** points) Now consider you want to execute the same query on a cluster of 4 machines, with each machine being identical to the one from part (a). Calculate the time needed in I/Os for the best parallel execution plan you can think of and describe this plan. Assume that tuples for both relations are equally distributed among the 4 machines in a round-robin fashion. Further assume that communication cost between the machines are negligible.

We begin by hashing each tuple of S on each machine into 4 buckets by using a hash function that depends only on the join attributes Y. These 4 buckets represent the 4 machines, and tuples are shipped to the processor corresponding to their bucket. The total number of disk I/Os needed to read R is 300, or 75 per machines.

At each machine, these 75 pages can be stored in the 101 available blocks of memory.

Then we repeat the same hashing and shipping for all tuples of R with cost 1000, or 250 per machine. This operation can be pipelined with the join and the total number of I/Os is $[B(S) + B(R)]/4 = 250 + 75 = 325$.

(c) (**4** points) Your classmate looks at what you have done and is completely fascinated: "Wow, you are so awesome! By using 4 instead of 1 machine, you have created a speed-up of more than factor 10. How on earth did you do that?" Be honest and explain the trick.

By using 4 instead of 1 machines, we also have 4 times the main memory available. For our example, this allows us to fit the whole build relation into the available memory. We thus only need to scan the larger relation only once instead of 3 times. We thus only need $B(S) + B(R) = 1300$ instead of 3300 I/Os. This difference explains the additional difference in time.

Initials: _____

5. (**5** points + **5** extra credit) **Sorting**

A big company in Seattle called `amazort.com` decides to offer "sorting as a service." Users can upload a table of numbers, and the company will sort it for them and provide the sorted result for download. Unfortunately, after launching their advertising, the executives at `amazort.com` realize that they have no idea how to deal with large data. They hire you as a recent cse444 alumni to help them figure this out. The future of the company's reputation now relies on your shoulders.

(a) (**5** points) The new service sorts one file at a time. You promise the executives that the I/O count for sorting the file (including reading it from disk and writing the result to disk) will be 4 times that of simply reading it. To guarantee this, you suggest introducing a limit on the size of file uploads to some number `maxsize` bytes per file. The `amazort.com` servers use 64 kilobytes disk blocks, and have 512 megabytes of memory available for sorting. What value would you recommend for `maxsize` in kilobytes?

- I/O cost of 4 times reading the file suggests a two-pass external merge-sort.
- A two-pass external sort can be used as long as $B \leq M(M-1) \approx M^2$.
- $M = 512\text{MB}/64\text{kB} = 2^{19}\text{kB}/2^6\text{kB} = 2^{13}$ blocks.
- Hence, maximal number of blocks is $B_{max} = 2^{26}$ blocks.
- Hence, `maxsize` $= 2^{26} * 2^6\text{kB} = 2^{32}\text{kB} = 4\text{TB}$.

(b) (**5** extra credit) The chief executive of `amazort.com` is unhappy about the idea of size limits. "We are `amazort.com`, we have no limits." He asks you how many disk I/Os it would take to sort a file that is arbitrary x bytes big on a single server. At first, you are puzzled: the instructors of cse444 never taught you how to sort a file that is too big to allow for a two-pass algorithm, and you almost lose faith in your ability to save the future of `amazort.com`. After you calm down, you do what you are good at and why they hired you: you think. Can there be something like a 3-pass / 4-pass / n-pass sort algorithm that deals with increasingly large data? Yes, there is! You figure out that the second step in the two-pass algorithm simply generalizes, and with that, the formula for the maximal page size for a n-pass algorithm. Given file size x bytes, and M main memory blocks (M is sufficiently large to approximate $M-1 \approx M$), how many I/Os are needed?

- The generalization from the two-pass sort-merge to the n-pass sort merge algorithm is: $B \leq M(M-1)^{n-1} \approx M^n$. Each step after the initial sorting of files of size M blocks, one can sort-merge $M-1$ files into one file of size $M(M-1)$. This can be repeated recursively.
- Hence, the maximal file size in bytes that can be sorted with a n-pass algorithm is $2^{16} * M^n$.
- Using the logarithm, we get the number n that is necessary to sort a file of size x as $n = \log_M(x/(2^{16}))$.
- An n-pass sort algorithm reads the file n times and writes it again n times. Hence $2n$ disk I/Os are needed for an n-pass sort algorithm.
- Hence, the necessary number of I/Os for size x is $2\log_M(x/(2^{16}))$.

Initials: _____

EXTRA PAGE