University of California, Berkeley

CS 186 Intro to Database Systems, Fall 2012, Prof. Michael J. Franklin

MIDTERM II - Questions

This is a **closed book** examination – but you are allowed one 8.5" x 11" sheet of notes (double sided). You should answer as many questions as possible. Partial credit will be given where appropriate. There are 100 points in all. You should read **all** of the questions before starting the exam, as some of the questions are substantially more time-consuming than others.

Write all of your answers on the **SEPARATE ANSWER SHEET**. We will be grading only the answer sheets. You must put your CS 186 Class account on the answer sheet (Question 0).

GOOD LUCK!!!!

Question 1 – Buffer Management [4 parts, 19 points total]

a) [10 points] Consider a database system with:

Three buffer frames (A, B, C), initially empty

A file of five disk pages (1, 2, 3, 4, 5).

A sequence of requests is made to the buffer manager as described in the Request column (below). At certain times a Pin request is immediately followed by an Unpin request (represented as Pin/Unpin), but other times Pin and Unpin requests happen separately.

Starting at time T4, fill in the table **ON THE ANSWER SHEET** showing the buffer contents after the completion of each operation using the CLOCK page replacement policy. Assume that at time T0 we initialize the clock hand to point to Buffer Frame A. To avoid checking the same frame twice in a row we always advance the clock hand after replacing a page.

For each page indicate the page number and pin count (PC)

If a page is unpinned, list the the reference bit (true or false i.e. max ref count = 1)

If the entry in a column **does not change** from the previous time slot, **leave it blank**

(2 points for each row; 1 point given if part of the change in the row was correct)

Time	Request	Buffer Frame A	Buffer Frame B	Buffer Frame C
T1	Pin 5	5 PC = 1	Empty	Empty
T2	Pin/Unpin 2		2 PC = 0; Ref = True	
T3	Pin/Unpin 3			3 PC = 0; Ref = True
T4	Pin 2		2 PC = 1	
T5	Pin/Unpin 4			4 PC = 0; Ref = T
T6	Unpin 5	5 PC=0; Ref = T		

T7	Pin 1	1 PC = 1		4 PC=0 Ref = F
T8	Pin 3			3 PC = 1

Question 1 – Buffer Management (Continued)

For parts b-d, Consider a buffer pool of 3 frames, and a heap file of 10 pages.
(3 points for each question; all or nothing)

b) [3 points] Assume we scan the heap file **twice** from start to finish. Starting with an empty buffer pool, using an **MRU** replacement strategy, how many of the 20 page requests will be page hits (i.e., found in the buffer pool)?

Answer: 3

c) [3 points] Now, consider the same scenario but instead using an **LRU** replacement strategy. Starting with an empty buffer pool, how would the **buffer hit rate** of LRU compare to that obtained with MRU (pick one):

A) LRU is higher B) MRU is higher C) They have same hit rate

Answer: B

d) [3 points] A friend from a university known as "The Farm" suggests that from his experience plowing fields (on the Farm), that to scan the file twice, you should first scan forward (from page 1 to page 10) but then in the second scan, go backwards (from page 10 to page 1). With an initially empty buffer pool and using an **LRU** policy, how does the **buffer hit rate** of this "plowing" strategy compare to simply scanning the file twice in a forward direction?

A) "Plowing" is higher B) "Plowing" is lower C) They have same hit rate

Answer: A

Question 2 – Join Operators [3 parts, 20 points total]
When considering the costs of the various join methods, we considered only the number of IOs. A more accurate estimation would make a distinction between *sequential* IOs and *random* IOs, since random IOs tend to take much longer than sequential ones. For this problem, use the following tables:

Relation A: 200 pages and 5 tuples per page = 1,000 tuples
Relation B: 500 pages and 12 tuples per page = 6,000 tuples

Assume that we only have 1 disk and that each table is stored in its own contiguous file but that the files are located in different places on the disk. Also, assume that we do **not** have to write the resultant tuples back to disk and that we don't cache any pages in our buffer pool.

a) [6 points] Consider the join of A and B using the naïve (record at a time) nested loops algorithm with A as the outer.

What is the total number of IOs this join will require?

[2pt] The formula for NNLJ is [M] + |M|[N] = 200 + 1000*500 = 500,200.

Of the total number of IOs, how many are **sequential** IOs?

[4pt] Each outer access will be random, as we'll be coming from the end of the inner. Similarly, the first access of each inner pass will be random, since we came from the outer. Thus, we will have 200 random IOs for the outer and 1000 random IOs for the inner. This leaves 499,000 IOs as sequential.

Of the total number of IOs, how many are **random** IOs?

1200 IOs as calculated above.

The rubric for each of these sequential/random IO questions went like this:

- +1 for sum of sequential and random being total
- +3 for sequential *or* random IOs being right
    - +2 if you forgot to take the first IO of each pass as random, but otherwise right

b) [6 points] Consider the join of A and B using the **page-oriented** nested loops algorithm with A as the outer.

What is the total number of IOs this join will require?

The formula for PNLJ is [M] + [M][N] = 200 + 200*500 = 100,200.

Of the total number of IOs, how many are **sequential** IOs?

Following from 1a, we will have 200 random IOs for the outer and another 200 random IOs for the inner, leaving us with 99,800 sequential IOs.

Of the total number of IOs, how many are **random** IOs?

400 IOs as calculated above.

c) [8 points] Now let's consider **index nested loops** join with A as the outer. We have an **Alternative 2** index on Relation B with 3 levels, including the leaves and root. Assume that the join column is a primary key for B, so every tuple of A will match at most 1 tuple of B.

How many IOs does it take to perform a single equality look up on the inner relation with this index?

3 IOs to get to the leaf level of the index. Since it's alternative 2, we need to perform 1 more IO to get 4 IOs total.

What is the total number of IOs this join will require?

The formula for INLJ is [M] + |M|(LookupCost) = 200 + 1000*4 = 4,200.

Of the total number of IOs, how many are **sequential** IOs?

Every IO will be random! Many people thought we'd have 200 or so sequential IOs, but note that for every outer page, we're coming from having read some inner page, so these will be random as well. Thus, 0 sequential IOs.

Of the total number of IOs, how many are **random** IOs?

4,200 IOs as calculated above.

For this problem, we took off 2 points if you said there were 200 sequential IOs.

Question 3 – Advanced SQL [4 parts, 15 points total]

Your new social site for cute dogs from Midterm 1, aww-or-not.com, where users can signup their cute dogs, and then can start rating how cute a dog is on a scale from 1 to 10, has been a big success so you are now going to add some analytics queries to your site.

Recall that aww-or-not.com has the following database tables.

```
/* Table of users. */
CREATE TABLE Users (
   user_id INTEGER NOT NULL,
   username TEXT NOT NULL,
    email VARCHAR(90)NOT NULL,
  PRIMARY KEY (user_id),
  UNIQUE KEY (email)
);
```

```
/* Dogs. Each has a single owner. */
CREATE TABLE Dogs (
    dog_id INTEGER NOT NULL,
    owner INTEGER NOT NULL,
    color TEXT NOT NULL,
    name TEXT NOT NULL,
    breed TEXT,
    age INTEGER,
  PRIMARY KEY (dog_id),
  FOREIGN KEY (owner)
        REFERENCES Users(user_id)
);
```

```
/* Table of user ratings of
cuteness for dogs.
num_awwws is an integer from 1
to 10. */
CREATE TABLE Awwws (
    voter INTEGER NOT NULL,
    dog INTEGER NOT NULL,
    num_awwws INTEGER NOT
    NULL,
  PRIMARY KEY (voter, dog),
  FOREIGN KEY (voter)
  REFERENCES Users(user_id),
  FOREIGN KEY (dog) REFERENCES
  Dogs (dog_id)
)
```

a) [5 points] You want to show a list of the dogs and their average cuteness scores, but only for dogs who have received at least 10 votes. For each such dog, show the name of the dog, and the average score. The structure of the query is below but put your answer **ON THE SEPARATE ANSWER SHEET**:

SELECT _________________
FROM _________________
WHERE _________________
GROUP BY_________________
HAVING________________

```
SELECT name, AVG(num_awwws)
FROM Dogs D, Awwws A
WHERE D.dog_id = A.dog
GROUP BY dog_id, name
HAVING COUNT(num_awwws) >= 10
```

1pt for each line.

Question 3 – Advanced SQL (Continued)

b) [4 points] On the answer sheet (NOT HERE) write the letters for ALL the following queries that are guaranteed to return no more than 10 rows (one or more may be correct)

```
A) SELECT dog FROM Awwws LIMIT 10;

B) SELECT voter FROM Awwws
   WHERE dog IN (SELECT dog_id FROM Dogs LIMIT 10);

C) SELECT DISTINCT(dog) FROM Awwws WHERE voter <= 10;

D) SELECT num_awwws FROM Awwws GROUP BY num_awwws;
```

Answer: A D (1pt for each option)

B: selects all the voters for 10 dogs, which could be more than 10

C: There is no constraint on voter, and voters can rate multiple dogs.

c) [3 points] You wanted a query to return the name of the user who has voted for the most dogs. The following query looks like it could do it, but it has a bug. On the answer sheet, **briefly** explain why the query below could return a wrong answer based on the schema above.

```
SELECT username
FROM Users U, Awwws A
WHERE U.user_id = A.voter
GROUP BY username
ORDER BY COUNT(username) DESC
LIMIT 1
```

Answer: Could return a wrong answer if multiple users have the same username. This would aggregate users with the same name together, which is a bug. (-1pt for a bug fix, but no explanation)

d) [3 points] On the answer sheet – state **in a single sentence** what the following query returns.

```
SELECT *
FROM Users U
WHERE U.user_id IN (SELECT voter FROM Awwws A
                    GROUP BY voter
                    ORDER BY COUNT(voter) DESC
                    LIMIT 1)
```

Answer: The user tuple of the user who has voted for the most dogs. (ties broken arbitrarily)

Question 4 – Query Optimization [5 parts, 22 points total]

Consider the following catalog info and statistics for the aww-or-not.com database:

Users, 200K tuples, 2K pages
Dogs, 500K tuples, 5K pages; age values in range [1, 20]
Awwws, 5000K tuples, 50K pages; 100k distinct dogs; num_awwws values in range [1, 10]

Consider the following query:

```
SELECT dog_id,
FROM Dogs, Awwws
WHERE dog_id = dog AND age < 5 AND num_awwws > 9;
```

a) [4 points] Estimate the result size (i.e., number of tuples) for this query.

4pts: for 100k, or 125k

2pts: for any answer that is off by a small factor, e.g., 500k, and for any answer that lists out the correct formula: (NTuple(Dogs) * NTuples(Awwws)) * RFs.

b) [4 points] For part (a) we had to assume uniformly distributed values. Given the following histograms for the value distributions of Dogs. age and Awwws.num_awwws, respectively, what would be the estimated result size now? (You can still make the uniform assumption on dog ids)
The histogram of **Dogs.age** (e.g., 9% of Dog tuples have age values in the range [11-12]):

value	1-2	3-4	5-6	7-8	9-10	11-12	13-14	15-16	17-18	19-20
count	20%	20%	15%	10%	10%	9%	7%	7%	1%	1%

The histogram of **Awwws.num_awwws**:

value	1	2	3	4	5	6	7	8	9	10
count	1%	3%	5%	6%	8%	20%	30%	15%	10%	2%

4pts: for 40k

2pts: for any answer that is ⅖ of your answers from a), e.g., you got 500k for a) and 200k for this one, which means you know how to use the histogram.

c) [6 points] Estimate the cost (in number of disk I/Os) of each step of the following plan for the given query. **Don't use the above histograms for your estimations in this part!**

i) OUTER: HeapScan on Dogs to select Dogs.age < 5; don't write the output.
2pts: 5k
ii) INNER HeapScan on Awwws to select num_awwws > 9; write the output to a temp file.

2pts: 55k

1pt: 50k (which means you only miss the cost of writing the result out)

iii) JOIN: Page-oriented nested-loop join, of (i) and (ii)

2pts: for 5M or 3.75M, but we give full credit to 5M + X and 3.75M + X, where X is a small additional cost in thousands for counting in the cost of i) or ii).

d) [4 points] Consider the plan for part (c) above, but with the INNER being an **on-the-fly selection** rather than writing to a temp file. What is the **total cost** of the plan in this case?

4 pts: for 50M+5k, or 37.5M+5k; any answer with 50M + X or 37.5M + X gets full credit, where X is in thousands.

2 pts: for any answer that is off by a small factor, or any answer that is 10 times your answers in c iii).

e) [4 points] Consider the 3-way join of Users, Dogs and Awws by their primary key/foreign keys. Give a **join order** that a System R-style optimizer **would not** evaluate in pass 3?

4pts: The answer should be None; every join order might be considered. Due to the ambiguity of the question, you get full credit as long as you write down a join order.

Question 5 – Indexes [6 parts, 23 points total]

It's Election 2012! You have been tasked with designing and maintaining the DBMS used to register voters for today's elections (don't forget to vote!). Consider a table in your system that has the following schema:

Voters(voter_id, age, name, address)

Consider a B+ Tree index constructed on the **age** field. Given the current state of the B+ answer the following questions: **NOTE**: Each part of the question starts with the tree you are given! So the answer from part a. does not affect part b and so on.

Note that the B+Tree is order d=2, which means that the minimum number of keys in a non-root node is 2, and the maximum is 4.

a) [2 points] Which **leaf** nodes need to be examined to answer the range query to retrieve all data entries between 18 and 38 (both inclusive)?

Node 2, Node 3, Node 4 -- 2 pts if all three nodes are listed, 1 pt for any two of them

b) [5 points] Show the keys in the root node after inserting a data entry with key "47" into the tree. How many levels does the resulting tree have?

Root node contains 20 32 42 46 - 3 pts // 1 point off if any key is missing
Number of levels 2 - 2 pts

c) [5 points] Starting with the original tree: Show the keys in the root node after inserting a data entry with key "26" into the tree. How many levels does the resulting tree have?

Root node contains 32 - 3 pts
Number of levels 3 - 2 pts

d) [5 points] Starting with the original tree: Show the keys in the root node after deleting the data entry with key "42" from the tree. Remember that you need to preserve the "order d=2" constraint on the tree. How many levels does the resulting tree have?

Root node contains 20 32 39 46 - 3 pts // 2 points for getting 39, 1 pt for the other entries
Number of levels 2 - 2 pts

e) [3 points] Write a short SQL query on Voters that can be answered efficiently with a **clustered** B+Tree index on age, but that **could not** take advantage of an **unclustered** B+Tree index on age.

SELECT * from Voters where Voters.age > 18 and Voters.age < 40;
Any range query gets 3 points
Query selecting only 1 key (e.g age = 10) gets 0 points as performance depends on cardinality

f) [3 points] Write a short SQL query on Voters that can be answered efficiently with a B+Tree on age, but that **could not** take advantage of an Extensible Hash index on age.

SELECT * from Voters where Voters.age > 18 and Voters.age < 40;
Any range query gets 3 points
Any query which has ORDER BY age also gets 3 points
GROUP BY age gets 0 points as hash index can be used for efficient aggregation as well