University of California, Berkeley

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

MIDTERM I - Solutions

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 – Sorting/Hashing [6 parts, 24 points total]

You have recently started a new company to help people sort their data. Currently, your servers have 4KB disk blocks, and have 800KB of memory available for sorting.
Your pricing plan is as follows:

> You charge $1 for every I/O request that gets performed during sorting.
> You do not charge to store data on disk!
> You charge only for sorting (including the cost of writing your sorted output.)

a) [3 points] Initially, your company only attracted only small users. Your first user wanted to sort a file that was 640KB in size. How much did your company earn?

$320
B = 800/4 = 200
N = 640/4 = 160
We can sort this in one pass, so the number of I/Os = 2N = $320.

b) [3 points] As news of your company's awesome sorting service spreads, you begin attracting larger and larger customers. Your next customer had a huge file of 1200 KB. How much did you earn?

$1200
B = 800/4 = 200
N = 1200/4 = 300
This requires two passes, so we need 4N I/Os = $1200.

c) [5 points] Using the optimization of tournament sort/heapsort which produces sorted runs of average size 2B in Pass 0, how much would you earn with the 1200KB file?

$300
2B = 2*(800/4) = 400
N = 1200/4 = 300
This requires one pass, so we need 2N I/Os = $600.

d) [5 points] Mr. X wants the biggest bang for his buck. What is the size in KB of the biggest file he could sort for $100,000 using the original sorting algorithm (runs of size B in Pass 0)?

100,000 KB
In one pass, we can sort 200 pages. This would cost 2N = $400. But we have more of a budget - so we can try to sort more pages using a 2nd pass
In two passes, we can sort a maximum of 200*199=39,800 pages of data. This would cost 4N = $159,200.
So we're on the right track with 2 passes, but we need to sort less data.
Specifically, 4N=$100,000, so N = 25,000 pages.
So how many KB? 4N = 100,000 KB.
Rubric: -2 points if you used log equation instead of giving the precise answer.

UC Berkeley now decides to use your service. The Chancellor of Berkeley wishes to know the distribution of the hometowns of all the students. Since he doesn't care about any form of ordering, Berkeley decides to hash the students into groups.

Each student tuple consists of (SID, name, gpa, major, hometown, address, photo). Due to the extra fields, the size of each student tuple is about 10KB. The number of students in Berkeley is 50,000. Being your alma matter, you give them a special server that has 100KB disk blocks and 101 buffer (memory) pages that can hold 100KB of data each.

e) [4 points] How many times do we have to run the **Partitioning** stage of hashing to hash the students, assuming all the partitions end up being the same length?

1 pass

50,000 students means we have 50,000*10KB of data = 500,000/100 pages = 5,000 pages of data. After one pass of partitioning, each partition will be 50 pages long (5,000 data pages / 100 buffer pages), so they'll fit nicely into the ReHash stage.

f) [4 points] What will be the size in KB of the average partition at the start of the **ReHash** stage?

5,000

Calculation shown above.

Rubric: -2 points if you gave the result in number of pages instead of KB.

Question 2 – Schema Refinement and Normalization [6 parts, 22 points total]

Consider the relation R with attributes A B C D E F
and with the functional dependencies: {$A \rightarrow BF$, $B \rightarrow F$, $CD \rightarrow E$, $DE \rightarrow F$ }

a) [4 points] Give the attribute closure of CD, also written as CD^+. (In other words, given just CD, what can we derive?)

CD -> CDEF

Rubric: 4 points for correct
- -2 points for missing E or F
- -2 points for adding another attribute
- (No points for CD – we gave full credit to solutions without CD.)

b) [4 points] There exists a single candidate key (i.e., minimal superkey) for R. What is it?

ACD

Rubric: 4 points for correct
- -2 points for missing A, C, or D.
- -2 points for adding another attribute

c) [3 points] Consider the decomposition of relation R into tables: ABF, BF, CDE, DEF. For each of the following, indicate on the answer sheet if it is True or False.

i. All relations in the decomposition are in BCNF.
False (B -> F violates)

ii. The decomposition is Lossless Join.
False (no way to join up the tables)

iii The decomposition is dependency preserving.
True (formed from the dependencies)

d) [3 points] Consider the decomposition of relation R into tables: ACE, BDF. For each of the following, indicate on the answer sheet if it is True or False.

i. All relations in the decomposition are in BCNF.
False (B -> F violates)

ii. The decomposition is Lossless Join.
False (no shared attributes, can't join!)

iii The decomposition is dependency preserving.
False (A -> BF nonexistent, for instance)

e) [4 points] Consider the decomposition of relation R into two tables: ABC, DEF. Which dependency or dependencies causes this decomposition to violate BCNF? (0 or more answers may be correct)

(A) $A \rightarrow BF$
(B) $B \rightarrow F$
(C) $CD \rightarrow E$
(D) $DE \rightarrow F$

A -> BF, which by Armstrong's Axioms implies A -> B, is violated by ABC. The others don't exist across the tables or are key constraints.

Rubric: 4 points for correct
-4 points for not having A
-2 points for every extra answer

f) [4 points] Perform a BCNF decomposition, considering the functional dependencies as shown **from left to right**. Show the final decomposed schema on the answer sheet.

AB BF CDE ACD

Rubric: 4 points for correct
-1 point for missing relation
-1 point for extra relation
only -2 points for the special case of failing to decompose 1 relation into 2 answer relations (e.g., ABF -> AB, BF)

Question 3 – SQL [5 parts, 21 points total]

You and your best friend decide to start a new social site for cute dogs! Your new service, aww-or-not.com, is a new crowd-sourced dog cuteness rating system for users to rate cute dogs and meet new dog owners! Users can signup their cute dogs, and then can start rating how cute a dog is on a scale from 1 to 10.

You are in charge of implementing the prototype, so you start off with creating some 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
);
```

```
/* 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,
  FOREIGN KEY (dog) REFERENCES Dogs
)
```

a) [4 points] To show how diverse your service is, you want to display all the unique colors of dogs that are signed up on your website. On the answer sheet list ALL the queries that are guaranteed to return all the dog colors with no duplicates (One or more may be correct)

(A) SELECT DISTINCT color FROM Awwws, Dogs WHERE dog = dog_id;
(B) SELECT color FROM Dogs;
(C) SELECT DISTINCT color FROM Dogs;
(D) SELECT DISTINCT color FROM Users, Dogs, Awwws
WHERE user_id = owner AND user_id = voter AND dog = dog_id;
(E) SELECT color FROM Awwws, Dogs;

Rubric: -4 points for no C
-2 points for 1 extra answer
-1 points for every extra answer beyond the first

Question 3 – SQL (continued)

b) [4 points] You also want to view which dogs have received a cuteness rating of 10 (maximum cuteness). Which query will return all **dog names** and **owner names** of all the dogs that have received at least one maximum cuteness rating of 10? On the answer sheet list the query of your choice (one answer is correct).

(A) SELECT name AS dog_name, owner FROM Awwws, Dogs
WHERE dog = dog_id AND num_awwws = 10;

(B) SELECT name AS dog_name, username AS owner_name FROM Awwws, Dogs, Users
WHERE owner = user_id AND dog = dog_id AND num_awwws > 9;

(C) SELECT name AS dog_name, username FROM Awwws, Dogs, Users
WHERE owner = user_id AND num_awwws = 10;

(D) SELECT name, num_awwws FROM Awwws, Dogs
WHERE dog = dog_id;

(E) SELECT name, username FROM Users, Awwws, Dogs
WHERE dog = dog_id AND num_awwws = 10;

Rubric: -4 points for incorrect

c) [4 points] You love bulldogs, so you want all bulldog owners to be more engaged with your site. You want to find all the bulldog owners who have not rated any dogs, so you can notify them to visit the site. Which queries will return the **emails** of the bulldog owners who have not rated any cute dogs? On the answer sheet (NOT HERE) mark the letters for ALL the queries that are guaranteed to return the emails (Zero, one or more may be correct)

(A) SELECT email FROM Dogs, Awwws, Users
WHERE breed = "bulldog" AND owner = user_id AND user_id != voter;

(B) SELECT email FROM Dogs, Awwws, Users
WHERE breed = "bulldog" AND owner = user_id AND dog_id != dog;

(C) SELECT email FROM Users
EXCEPT
SELECT email FROM Users, Dogs, Awwws
WHERE breed = "bulldog" AND owner = user_id AND user_id = voter

(D) SELECT email FROM Users, Dogs
WHERE breed = "bulldog" AND owner = user_id
EXCEPT
SELECT email FROM Awwws, Users
WHERE user_id = voter

(E) SELECT email FROM Users, Dogs, Awwws
WHERE breed = "bulldog" AND user_id = voter

Rubric: -4 points for no D
-2 points for 1 extra answer
-1 points for every extra answer beyond the first

Question 3 – SQL (continued)

d) [4 points] Consider the following query:

```
SELECT user_id, dog, num_awwws FROM Users, Dogs, Awwws
WHERE user_id = owner AND dog = dog_id
EXCEPT ALL
SELECT user_id, dog, num_awwws FROM Awwws, Users, Dogs
WHERE num_awwws <= 5 AND  user_id = owner AND dog = dog_id
```

Which ONE of the following could be a valid result set for this query?

(A)	(B)	(C)	(D)	(E)
(1, 1, 6)	(1, 1, 6)	(1, 1, 5)	(1, 1, 1)	(1, 2, 5)
(1, 1, 7)	(2, 1, 7)	(1, 2, 5)	(2, 3, NULL)	(1, 3, 6)
(1, 1, 8)	(3, 4, 8)	(1, 2, 5)	(3, 4, 10)	(1, 4, 7)
(1, 1, 9)	(4, 10, 9)	(1, 2, 5)	(4, 10, 5)	(1, 5, 8)

Rubric: 4 points for A
1 point for B (not correct, but close)
0 points otherwise

e) [5 points] Say that the Users table has U rows, the Dogs table has D rows and the Awwws table has A rows, and U > 1, D > 1 and A > 1. How many rows would the following query have in its result set? (Note: all the foreign key constraints are enforced by the database)

```
SELECT * FROM Users, Dogs WHERE user_id = owner;
```

On the answer sheet (NOT HERE), choose ONE of the following:

(A) U + D   (B) U * D   (C) More than U * D   (D) D   (E) U

Rubric: -5 points for incorrect

Question 4 – Relational Algebra [2 parts, 10 points total]

a) [5 points] Consider two relations R(A, B) and S(B, C). Which one of the following relational algebra expressions is not equivalent to the others? Note that π eliminates duplicates.

(A) $\pi_{R.A,\ R.B}\ (R \bowtie S)$

(B) $R \bowtie \pi_{S.B}\ (S)$

(C) $R \cap (\pi_A\ (R) \times \pi_B\ (S))$

(D) None, they are all equivalent.

b) [5 points] Friend Recommendation. Using the schema from Question 3, we would like to suggest user A to friend user B if B has voted a 10 to any of A's dogs. To be concrete, we need a query that lists all the (user_A, user_B)-pairs if user_B has voted 10 to any dog of user A, where user_A and user_B are user ids. Your website is getting a ton of registered users, so we are not only concerned with result correctness but also query efficiency. Which one of the following joins do you think is sufficient for this query while involving the fewest tables possible?

(A) Awws ⋈ Awws

(B) Dog ⋈ Awws

(C) Users ⋈ Awws

(D) Users ⋈ Users ⋈ Awws

(E) Users ⋈ Dogs ⋈ Awws

Question 5 – Conceptual Design/ER [5 parts, 22 points total]

NOTE: For the SQL statements in this question we will not be checking the TYPEs

BeMAD! – The Berkeley Movie Aficionado Database

You have been hired for a brand-new startup called BeMAD! The idea is to allow users to rate movies they like and pick out movies that they can go and watch with their friends. As a database guru, your first task is to design a database schema that can store all the data. The first task you have is to store data about movies and actors. The requirements are:

i) For every movie, we want to store its title, the lead actor that stars in it, the year it was released and the genre of the movie.

ii) In addition we want to store every actor's name, birthday and a photo.

iii) We wish to ensure that no two movies released in the same year have the same title and that no two actors have the same name to prevent our users from getting confused!

iv). For every leading role that an actor plays, we wish to know the name of the character they played.

v) Finally, every movie has EXACTLY ONE lead actor, but actors can star in 0, 1 or more movies.

a) [5 points] In the ER diagram on the answer sheet underline the primary keys and connect the given entity and relationship sets using the appropriate line and/or arrow. If bolding a line/arrow, be sure to clearly make it bold.

Rubric: +2 for bold arrow from movies
+.5 for line from actors
+1 each for underlining title and year
+.5 for underlining actor_name.

b) [4 points] Complete the SQL statements given in the answer sheet which will create the table for the combined entity/relationship set in our ER diagram.

Rubric: +2 for getting title, year, actor_name, character_name (0.5 each)
+1 for each foreign key: (title, year) and (actor_name).

c) [5 points] Let's add users! As the next step in your database design, you wish track users and their friends. In the ER diagram given in the answer sheet, connect the relations for the Friends relationship set.

Rubric: +2 for each of the 2 lines from User to Friends_Of.
+1 for labeling the lines Friends_From or Friends_To (or any other name).

d) [4 points] Now, complete the SQL statement on the answer sheet to create a table for the same.

Rubric: +1 for two user_ids that are fields
+1 for primary key (both user_ids)
+2 for the two foreign keys
-1 if no role names are used (e.g., Friends_From, Friends_To)

e) [4 points] If you wished to allow users to rate movies and to store those ratings, which of the following statements are true? (Note: more than more may be correct!)

(A) Add a new attribute 'rating' to each movie

(B) Add a new relation between users and movies 'rates' with an attribute for the rating

(C) Add a new entity set ratings to store the rating and connect it to movies, users with a ternary relation

(D) None of the above

Rubric: +2 for each B and C
-2 for any other option