Question Booklet

W4111 Introduction to Databases
Spring 2019
Midterm 2 Exam Solutions
Instructor: Eugene Wu

Closed Book, 1 page notes: 8.5x11" letter paper, both sides
Duration: 75 minutes
Havemeyer 309

Instructions

This is the question booklet, which contains questions for the exam.
There is a separate answer booklet for your answers.

1. You are supposed to write your answers on the answer sheets.
2. The staff will ignore text written on the question sheets.
3. You will submit the Quesion AND Answer booklets at the end of the exam. If we do not receive both booklets with your UNI, **you will receive a zero**.

Your Name: Alice H. Acker
Your UNI: aa0000

ALL regrade requests due by 5/6 10AM EST

If we added your score incorrectly, submit regrade request on Gradescope.
If there is an error in the solutions, please let us know with a **private piazza message**.
If you want a regrade of a question, submit a regrade request on Gradescope—we will *regrade it carefully and deduct points for syntactic errors that we ignored the first time around.*

1 (20 points) Query Execution

Consider a database with the following statistics, tables, and indexes. Assume:

- all attributes are integers.
- fanout is the number of directory entries in a directory page.
- values are uniformly distributed, unless otherwise specified.
- hash index will hash different search key values to different buckets (no collisions). Note that there may be overflow pages.
- no nulls

Fill Factor	0.25
Dir Entry Size	5 bytes
Tuple Size for all tables	50 bytes
Page Size	2000 bytes
ICARD(T1.a)	100
ICARD(T2.b)	1,000
ICARD(T3.c)	50
minmax(T1.a)	[0, 99]
minmax(T2.b)	[0, 1000]
minmax(T3.c)	[0, 99]

Table Name	Cardinality	Indexes
T1	100 tuples	Primary tree on T1(a)
T2	10,000 tuples	Secondary hash on T2(b) Primary tree on T2(b)
T3	100,000 tuples	Primary tree on T3(c)

1. **(3 points)** Consider the query $\sigma_{b=10}(T2)$. What is the best access path for this query?

> Each page has the capacity to store 100 directory entries, or 10 tuples (when taking into account fill factor). The query cardinality is $\frac{1}{1000} * 10000 = 10$. File scan takes 1000 pages. Secondary hash requires at least 10 pages (one for each output tuple). Primary tree has height 2, and takes 1 data page to store the 10 results. **Primary tree**.

2. **(3 points)** How many pages does the best access path cost for the above query?

> 3 pages. 4 is acceptable if answer notes that the 10 tuples may be spread across two data pages.

3. **(3 points)** Write a relational algebra statement over $T2$ with cardinality ≥ 1 that is faster using the secondary hash index access method than using the primary tree index.

> $\pi_b\sigma_{b=10}(T2)$. The key is to specify an index-only query with an equality predicate.

4. **(4 points)** Assuming T3.c is a foreign key that references T1.a, what is the *exact* cardinality of the following SQL query? The answer should be a single number.

```
SELECT *
  FROM T1, T2, T3
 WHERE T1.a = T2.b AND T2.b = T3.c AND
       T1.a = 10 AND T2.b > 50
```

0.
No records satisfy `T1.a = 10 AND T2.b > 50 AND T1.a = T2.b`. Partial credit given if you used the standard selectivity formula and took foreign key reference into account.

5. **(3 points)** Assume no foreign key relationships in our database schema. The selinger optimizer optimizes the following relational algebra query: $T3 \bowtie_{T3.c=T1.a} T1$. What is the cardinality of the resulting plan?

$100 * 100,000 * \frac{1}{max(100,50)} = 100,000$

6. **(4 points)** What is the cost of the query plan that the selinger optimizer will generate for the query in Problem 1.5 above?

500 bytes per page is used
10 tuples per page
100 directory entries per page
10 pages for T1
T1 outer relation
10,000 leaf pages for T3
height of 2 in T3 primary tree index
1000 T3 tuples expected to match each tuple in T1
100 leaf pages retrieved for each tuple in T1
$10 + 100 * (2 + 100) = 10 + 10200 = 10210$

2 (18 points) Loose Lips Sink Ships

Sailors sail ships. Sailors gossip. A lot. But *loose lips sink ships*. The wise admiral Grace Hooper has installed devices on all ships to record all gossip. She stores the information using the following schema:

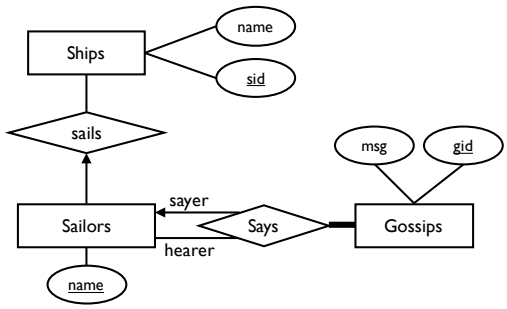
```
CREATE TABLE Ships(
  sid    int PRIMARY KEY,
  name   text NOT NULL
);
CREATE TABLE Sailors(
  sid    int NOT NULL REFERENCES Ships,
  name   text PRIMARY KEY
);
CREATE TABLE Says(
  sayer  text REFERENCES Sailors(name),
  hearer text REFERENCES Sailors(name),
  gid    int  REFERENCES Gossips(gid),
  PRIMARY KEY (sayer, hearer, gid)
);
CREATE TABLE Gossips(
  gid    int PRIMARY KEY,
  msg    text NOT NULL
);
```

1. **(2 Points)** Let the sailors table contain 1000 records and ships table contain 100 records. What is the cardinality of the query $sailors \bowtie_{sid} ships$?

1000

2. **(6 Points)** The following ER diagram is supposed to express the schema above but may contain errors. An ER constraint is an error if it contradicts the SQL schema (it allows data that the SQL schema disallows), or if it is incorrect.

   On the **answer sheet**, identify and circle up to 3 errors in the diagram. In cases where there is an error of omission (something should be in the ER diagram, but is not) circles the place where it should be and draw what is missing. If there are no errors, circle `NO ERRORS` to receive full points. Circles that contain more parts than necessary (e.g., circling the whole diagram) do not receive points.



Arrow from sailors to sails should be a thick arrow (eactly once),
sayer arrow is incorrect,
thick line between gossips and says not enforced in schema.
Sailors sid not in ER diagram is not strictly needed, but was accepted.

Circling sayer and hearer edges together did not recieve points.

3. **(5 points)** The *Lucielle* is the name of a top secret ship. Admiral Hooper wants to know how far knowledge of the ship as spread throughout the fleet. Write a SQL query that finds the number of sailors on each ship that have heard (a "hearer") a piece of gossip whose message contains the ship name. The query should return the ship id and count, with schema: `(sid, count)`

```
  SELECT sid, count(distinct sailors.name)
    FROM gossips, says, sailors
   WHERE gossips.msg like '%Lucielle%' AND gossips.gid = says.gid AND
         sailors.name = says.hearer
GROUP BY sailors.sid
```

4. **(5 points)** The admiral is furious that knowledge of the *Lucielle* has spread so far and is convinced that a Sailor named "Turing" is responsible. Turing *knows* that he has not participated in any gossip at all! Help Turing clear his name by writing a SQL query that finds the names of sailors that have never said (a "sayer") any piece of gossip. The query should return the name of each such sailor, with schema: `(name)`

```
SELECT name
FROM Sailors
```

```
          WHERE (SELECT COUNT(*)
                   FROM Gossips, Says
                  WHERE Says.gid = Gossips.gid AND
                        Sailors.name = Says.sayer) = 0;
```

Other solutions using `NOT IN` or `EXISTS` were great too.

3 (6 points) Transactions

Consider a table `R(a, b)`, with a primary tree index on `R.b`. It contains the tuples A, B, and C. Assume that the database stores one tuple per page.

```
A:  (1, 1)
B:  (2, 2)
C:  (3, 2)
```

Alice submits the transaction T1 `UPDATE R SET a = a + 1 WHERE b` $\leq$ `1` at the same time Bob submits the transaction T2 `UPDATE R SET a = a * 2 WHERE b = 2`. The database uses the available indexes to execute both queries.

1. **(3 points)** Write a possible schedule for T1 and T2 that exhibits the dirty read anomaly. Write `NONE` if no such schedule exists.

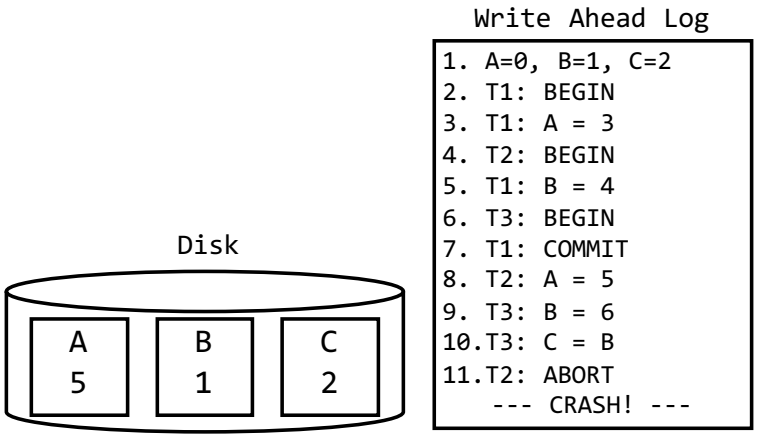
   NONE. Partial credit if assumed full table scan and produced a schedule with dirty read.

2. **(3 points)** Write a possible schedule for T1 and T2 that exhibits the lost write anomaly. Write `NONE` if no such schedule exists.

   NONE. not possible.

4 (10 points) Recovery

The database contains three tuples A=0, B=1, C=2, and each page contains the corresponding tuple. T1, T2, and T3 start executing, and the database follows the write-ahead logging protocol as described in class. Each log record has an ID. The database crashes, and when the daabase recovers, it find the following contents of the disk and write-ahead log.



1. **(3 Points)** During the `REDO` phase of recovery, which page-write log records need to be re-executed? Fully fill in the circles next to those log record ids on the answer sheet.

5, 9, 10. One point for each correct. Also acceptable is 5 only.

2. **(3 Points)** During the `UNDO` phase of recovery, which page-write log records need to be undone? Fully fill in the circles next to those log record ids on the answer sheet.

8, 9, 10. One point for each correct. Also acceptable is 8 only.

3. **(4 Points)** Consider the following sequence of database actions for one transaction. Check the statements where, if a crash occurs immediately after the statement, the database cannot correctly recover.

```
1. write B=2 in memory
2. write A=1 in memory
3. flush log record for "Write(A=1)"
4. write C=3 in memory
5. flush log record for "Write(C=3)"
6. flush C
7. flush A
8. flush log record for "COMMIT"
```

8. This does not obey the WAL protocol. B's write was not flushed to a log. After recovery, B=2 will not be recovered. If a crash occurs before the commit log record, writes will be correctly aborted, since B's page was fortuitously not written to disk.

5 (16 points) Functional Dependencies

Consider the relation $R(ABCDEF)$ and the following functional dependencies:
$\mathcal{F} =$ { ABCD $\rightarrow$ ED, B $\rightarrow$ FC, AF $\rightarrow$ DE }

5.1

1. **(3 points)** List the key(s) of R.

AB

2. **(4 points)** List a minimal closure of $\mathcal{F}$.

```
B -> F, B -> C, AF -> D, AF -> E
```

5.2 (3 points)

Consider the following decomposition. Fill in the appropriate circles on the answer sheet.

`BCA, BADE, BF, AFDEC`

1. **(1 points)** Is the decomposition in 3NF?

FALSE. BCA violates.

2. **(1 points)** Is the above a lossless-join decomposition?

This is lossless. The original relation can be reconstructed by ((BCA join ABDE) join BF) join AFDEC

3. **(1 points)** Is the above a dependency-preserving decomposition?

FALSE

5.3 (3 points)

`CB, FB, AEDB, AFD, AFE`

1. **(1 points)** Is the decomposition in 3NF?

TRUE

2. **(1 points)** Is the above a lossless-join decomposition?

TRUE

3. **(1 points)** Is the above a dependency-preserving decomposition?

TRUE

5.4 (3 points)

Consider the following decomposition. Fill in the appropriate circles on the answer sheet.

`AB, BC, CD, DE, EF`

1. **(1 points)** Is the decomposition in 3NF?

TRUE

2. **(1 points)** Is the above a lossless-join decomposition?

FALSE

3. **(1 points)** Is the above a dependency-preserving decomposition?

FALSE

6 (12) Short Answers

(3 points each) Answer the following questions in *at most 2 short sentences* each.

1. Describe the SQL clause that enables it to perform graph analyses such as page rank, and shortest paths using a *single* SQL statement. If such analyses are not possible, write `NONE`.

> *Recursive* WITH clause. WITH clause alone is not sufficient.

2. What type of query is nearly always faster (and often much faster) to perform on a primary heap file than a primary B+-tree index? If no such query exists, write `NONE`.

> Inserts are O(1), whereas they require traversing and rebalancing the B+-tree. Full table scans are also acceptable. Large range queries are not acceptable, since primary tree index can identify the range and scan the range efficiently, whereas it will require reading ALL pages in a heap.

3. Describe the difference between Serializability and Conflict Serializability in *at most* 2 sentences:

> Conflict serializability is a subset of serializability. This part is important.
>
> Partial credit for correctly stating that serializability is that the result of a schedule is equivalent to some serial schedule. Conflict serializability describes a subset of serializable schedules that do not contain cycles in the conflict graph.

4. Provide one reason why each relation should not have more than one primary file/index:

> In creates redundancy in the storage of the relation's data, which can be expensive to keep consistent. Space, redundancy, and updates/inserts are all acceptable. Note that primary index is NOT the same as primary key! Primary index simply stores the tuple contents in the data pages, and is indexed based on the *search key*.

5. In your opinion, what was the coolest thing that you learned in this course and why? Best answer (subjectively judged by the staff) gets 2 extra credit points. You may write and draw to your heart's content.

7 Extra Credit (Up 3 Points Extra Credit)

Project 2 is intended to give students more hand-on experience with databases. Describe an alternative project 2 assignment that would be awesome to do in future 4111 classes. Points subjectively given based on awesomeness of idea, and feasibility for the staff and students.