W4111 Introduction to Databases
Spring 2016
Cumulative In Class Test

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

Instructor: Evan Jones
Thursday, April 28th, 2016

Your Name: ____

Your UNI: ____

Problem	**Points**	**Score**
1	10	
2	20	
3	14	
4	26	
TOTAL	70	

Good Luck!

1 (10 points) Short answer

(2 points each) Answer the questions below in 1-2 sentences each.

1. What is a minimum cover of a set of functional dependencies?

2. A table has a B+Tree index on attributes (a, b, c, d). A query specifies `WHERE a=5 AND b<500 AND d>100`. Can this query use the index? Why or why not?

3. Describe one kind of query where a B+Tree clearly better than a hash index? Why is it better?

4. What is required in order to be able to use an index nested loops join to execute a join between two tables?

5. A student who has not taken W4111 creates a program that issues 10 read-only queries. Since their program only reads values, they decide they don't need to use a transaction. Describe one anomaly that could happen because they aren't using a transaction.

2 (20 points) Intensive Care Unit Normalization

Suppose you have the following schema representing the duties of nurses in the ICU:

```
icu_duty(duty_id, nurse_id, icu_stay_id, therapy_id, patient_id, length, report)
```

We denote this schema DNITPLR, representing each attribute by its first letter. Suppose you have the following functional dependencies:

D → NITPLR	`duty_id` is the key for the table
I → PL	Each `icu_stay` applies to one patient and has a fixed duration
N → TP	A nurse always applies the same therapy to the same patient

1. **(4 points)**: For the following example table, are there any rows which violate the stated functional dependencies? If not, explain why. If so, identify the duty_ids, attributes and functional dependency that is violated.

duty_id	**nurse_id**	**icu_stay_id**	**therapy_id**	**patient_id**	**length**	**report**
3000	Evan	5	pain killer	2007	10	report1
3001	Eugene	17	blood pressure	2007	5	report2
3002	Neha	4	blood sample	2003	7	report3
3003	Jinyang	17	blood sample	2007	4	report4
3004	Jinyang	6	temperature	2001	7	report5
3005	Neha	4	blood sample	2003	7	report6

2. **(6 points)**: Write a **BCNF** decomposition of this table in the space below. You can denote each table by the first letter of the attributes in it.

3. **(6 points)**: Is the schema you came up with in the previous problem dependency preserving? If so, explain why. If not, why not, and what could be done to address the issue?

4. **(4 points)**: Consider a simplified version of this schema with only the attributes DTP, and the projection of the previous functional dependencies: D $\rightarrow$ TP. A consultant decides that it should be decomposed into two tables: DT and TP. Being a good W4111 student, you know that this decomposition does not have the lossless join property.

   Create example tuples for the DTP table to prove to the consultant that the lossless join property does not hold. Write one tuple that is incorrectly produced by the join of DT and TP, that does not exist in your original DTP tuples to prove that this does not hold.

   Hint: Two rows in the DTP table is sufficient to prove this property, but you can use more than that if it makes it easier. It also might be helpful to write out the decomposed DT and TP tables from your example DTP table, but no marks will be given for it.

3 (14 points) Transactions and recovery

A W4111 student has created a database to store the number of items sold in a charity auction. The initial sold items table contains the following values:

item_id	**price**	**number**
1	10.00	1
2	5.00	0

The charity auction was a success, and the student needs to update the information. She writes two transactions, Transaction A and Transaction B:

Transaction A

```
BEGIN
UPDATE items SET number =
  number + 10 WHERE item_id = 1;
X = SELECT SUM(number) FROM items;
COMMIT;
PRINT X;
```

Transaction B

```
BEGIN
UPDATE items SET number =
  number + 1 WHERE item_id = 2;
X = SELECT SUM(number) FROM items;
COMMIT;
PRINT X;
```

The statement `X = SELECT ...` means the application variable `X` takes the value of the `SELECT` statement on the right. The statement `PRINT X` displays the variable `X` on the screen.

1. **(4 points)**: If the student executes these transactions using a database that correctly implements strict two-phase locking, what are the different possible results printed on the screen for transactions A and B if she executes them at the exact same time?

2. **(4 points)**: The student executes these transactions on a new database called FooDB. It breaks the transactions into individual read and write operations, and executes them in the following interleaved order:

Operation	**Transaction A**	**Transaction B**
1	Read(item_id=1)	
2		Read(item_id=2)
3	Write(item_id=1)	
4		Write(item_id=2)
5	Read(item_id=1)	
6		Read(item_id=1)
7	Read(item_id=2)	
8		Read(item_id=2)
8	Commit	
9		Commit

   Is this a serializable order? If yes, what is the equivalent serial order. If not, describe which operation(s) violate serializability.

3. **(2 points)**: What would be the result of executing this set of interleaved operations on a database that implements strict two-phase locking with shared read locks and exclusive write locks?

4. **(4 points)** The charity auction uses a database that provides durable transactions, correctly implemented with a write-ahead redo-only log as described in class. The database crashes in the middle of processing some operations. The on-disk state and the redo log are listed below. Reconstruct the state of the database after it recovers using the redo log in the space below.

On Disk State

item_id	**number**
w	1
x	2
y	3
z	4

Redo Log

transaction	**operation**
A	w=50
B	y=60
C	x=70
D	z=80
B	COMMIT
C	y=90
A	w=100
C	COMMIT

After Recovery State

item_id	**number**
w	
x	
y	
z	

4 (26 points) Query execution

To store a database of movies and actors, we create the following tables:

```
CREATE TABLE movies(
  m_id INTEGER PRIMARY KEY,
  m_name TEXT NOT NULL
);

CREATE TABLE actors(
  a_id INTEGER PRIMARY KEY,
  a_name TEXT NOT NULL
);

CREATE TABLE acts_in(
  m_id INTEGER REFERENCES movies
  a_id INTEGER REFERENCES actors
  PRIMARY KEY(m_id, a_id)
);
```

The tables are all stored in unordered heap files, without any indexes. For these questions, we will consider the cost to read a page from disk as being P, and assume the database must always read tables from disk. We will ignore all other costs.

1. **(2 points)**: What will the average cost be to execute the following query?

```
SELECT * FROM actors WHERE a_id = 1042;
```

2. **(2 points)**: What will the average cost be to execute the following query?

```
SELECT * FROM actors WHERE 1042 <= a_id and a_id <= 5072;
```

Hint: the following three parts are related; read all three before answering

3. **(4 points)**: We create a primary B+Tree index on `actors(a_id)`. Assume the height of the tree is 2 (2 levels of directory pages then the leaf pages). What is the average cost for the query in the previous question? Assume the query matches 1% of the tuples in the actors table, and the leaf nodes in the primary index occupy 10000 pages.

4. **(4 points)**: What would the cost of the query in the previous two questions be if we create a *secondary* B+Tree index on `actors(a_id)`? Assume the query matches 1% of the tuples in the actors table, and 10 leaf pages in the secondary index, and the unordered heap file occupies 8000 pages.

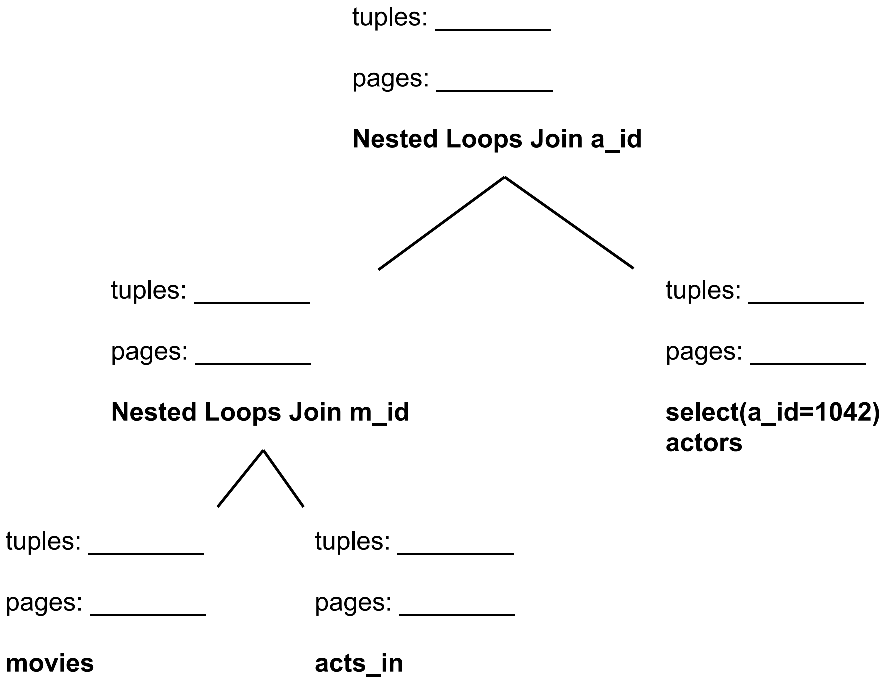
5. **(4 points)**: According to this simple model, the cost for answering this range query with a secondary index or a primary index are similar. This will not likely be true in reality. Which is probably faster for this query in a real system? Why?

6. **(10 points)**: Assume that we have no indexes, and our only join algorithm is a nested loops join. We execute the following query:

```
SELECT m_name, a_name
FROM movies, actors, acts_in
WHERE movies.m_id = acts_in.m_id AND actors.a_id = acts_in.a_id
  AND actors.a_id = 1042;
```

The database optimizer decides to execute the query with the following query plan. For each table and operator, estimate the number of output tuples and the total number of pages read up to that point in the query plan. (Note: this means the top pages value is the total for the entire query.) Assume that id values are uniformly distributed. The tables have the following number of rows and pages on disk:

Table	**Rows**	**Pages**
movies	1,000 rows	10 pages
actors	100,000 rows	10,000 pages
acts_in	10,000 rows	100 pages



Write any assumptions or calculations you make below: