CS4320 Final Exam

December 11th, 2018

(150 minutes working time)

Name: _____ Cornell NETID: _____

I understand and will adhere to the Cornell Code of Academic Integrity.

Signature

Maximum number of points possible: 120. This exam counts for 35% of your overall grade. Questions vary in difficulty. Do not get stuck on one question.

In all problems, whenever you think a problem is underspecified, make assumptions and clearly state them.

Good luck!

You have 150 minutes working time for this exam.

Part A) SQL Queries. (20 points)

Consider the database schema created by the following SQL commands:

```
CREATE TABLE Sailors (sid integer PRIMARY KEY,
sname varchar(20), rating integer, age real);

CREATE TABLE Boats (bid integer PRIMARY KEY,
bname varchar(20), color varchar(20));

CREATE TABLE Reserves(sid integer, bid integer,
day date, PRIMARY KEY (sid, bid, day),
FOREIGN KEY (sid) references Sailors (sid),
FOREIGN KEY (bid) references Boats (bid));
```

The database stores information on sailors (table `Sailors`), on boats (table `Boats`), and reservations for boats by sailors (table `Reserves`). The result of the solution queries for A.1 to A.4 must contain only one single column, duplicates in the query result are okay (e.g., if the question asks to retrieve sailor names, it is okay if the query returns the same name multiple times). You may assume that the database contains no `NULL` values and that no table is empty.

A.1) Write an SQL query retrieving the IDs (`bid`) of popular boats. We call a boat popular if it received more reservations than the average number of reservations per boat.
(5 points)

SELECT bid FROM boats B WHERE (SELECT COUNT(*) FROM Reserves R WHERE B.bid = R.bid) > ((SELECT COUNT(*) FROM Reserves) / (SELECT COUNT(*) FROM Boats));

A.2) Write an SQL query retrieving the names (attribute `sname`) of all sailors who made a reservation for at least one boat which was not reserved by anyone else.
(5 points)

SELECT sname FROM sailors S WHERE EXISTS (SELECT * FROM Boats B WHERE EXISTS (SELECT * FROM Reserves R WHERE R.sid = S.sid AND R.bid = B.bid) AND NOT EXISTS (SELECT * FROM Reserves R WHERE R.sid <> S.sid AND R.bid = B.bid));

A.3) Write an SQL query retrieving the names of all sailors (attribute `sname`) who seem to have a preference for the boat color (attribute `color`) red, i.e. who made more reservations for red boats than for boats of any other color.
(5 points)

SELECT sname FROM Sailors S WHERE (SELECT COUNT(*) FROM Reserves R, Boats B WHERE R.bid = B.bid AND B.color = 'red' AND R.sid = S.sid) > ALL(SELECT COUNT(*) FROM Reserves R, Boats b WHERE R.bid = B.bid AND B.color <> 'red' AND R.sid = S.sid);

A.4) Write an SQL query retrieving the names of all sailors (attribute `sname`) who made reservations for boats of at least three different colors (attribute `color`).
(5 points)

SELECT sname FROM Sailors S WHERE 3 <= (SELECT COUNT(DISTINCT color) FROM boats B, Reserves R WHERE B.bid = R.bid AND R.sid = S.sid);

Part B) Processing Cost. (20 points)

For the following questions, we assume two relations X and Y. X consumes 100 disk pages with 50 tuples per page. Y consumes 200 pages with 100 tuples per page. Calculate processing cost according to the cost model we saw in class (do not count the cost of writing out final results). X and Y are initially stored on disk.

B.1) We have an unclustered B+ tree index for X with four levels (including root node and leaf nodes). Each index leaf page contains 50 references to records in X (Alternative 2). The entire index is stored on disk (nothing in memory).
We use that index to retrieve all tuples satisfying an equality condition (on the index search key) with selectivity 10%. Calculate the total processing cost for using the index and retrieving the tuples.
(5 points)

We have cost 3 before finding the first leaf page, cost 10 for reading all leaf pages, and cost 10 * 50 = 500 for reading the associated data. Total cost: 513.

B.2) We sort Y using the general external merge sort algorithm seen in class. The sorting algorithm uses 10 buffer pages. Calculate the processing cost.
(5 points)

The first pass produces 20 runs of length 10. We merge nine runs together in each of the following passes (one output buffer). We have sorted everything after two more passes. We read and write all data in each pass but do not count the cost of the final write. Total cost: 2 * 2 * 200 + 1 * 1 * 200 = 1000.

B.3) We join X and Y using a block nested loops join, using X as outer operand. The join operation uses 12 buffer pages. Calculate the processing cost.
(5 points)

We have one output buffer, one buffer for reading the inner operand, and 10 buffer pages for reading blocks from the outer operand. We therefore need 10 iterations. In each iteration, we read the inner operand. Also, we count the cost for reading the outer operand once. Total cost: 100 + 10 * 200 = 2100.

B.4) Calculate the minimal number of buffer pages that we need to join X and Y via the refined sort-merge join (you may use the rule of thumb formula seen in class).
(5 points)

According to the rule of thumb formula, we need at least two times the square root of the number of pages in the larger relation (Y with 200 pages). Hence, we need ceil(2 * sqrt(200)) = 2 * 15 = 30 pages.

More precise calculations will also be accepted but are not required.

Part C) Database Design and Normalization. (20 points)

C.1) Consider the set of functional dependencies F={BC→D, A→C, G→H, C→E, EC→B}. Calculate the attribute closure of A.
(5 points)

We calculate the following attribute closures over the iterations of the algorithm seen in class: A, AC, ACE, ACEB, ACDEB. ABCDE is therefore the closure.

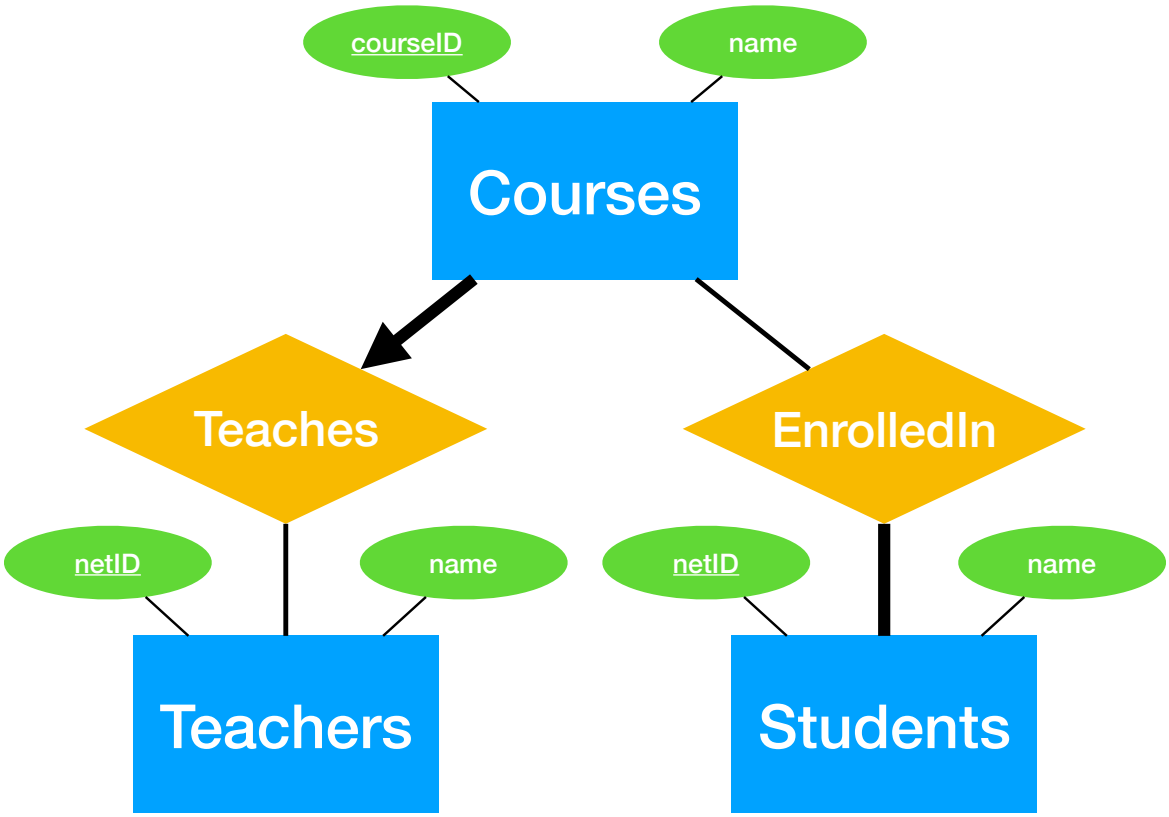
C.2) Consider relation R with attributes A, B, C, D, and E. Attribute A is a key and the following functional dependencies hold in R: {A→C, C→D, CD→E}. Bring R into Boyce-Codd Normal Form (BCNF) via decomposition (if necessary).
(5 points)

C→D is neither trivial nor does C contain a key. Hence, we need to decompose R into R1(ABCE) and R2(CD). The functional dependencies A→C and C→D apply in the resulting relations but A→C is okay since A is a key and C→D is okay since C is a key in R2.

C.3) Draw an Entity-Relationship diagram representing `students`, `teachers`, `courses`, a `teaches` relationship (capturing which teacher teaches which courses) and an `enrolledIn` relationship (capturing which students are enrolled in which courses). Your Entity-Relationship diagram must represent all constraints and attributes mentioned in the following:

- Students are associated with attributes netID (a unique student ID) and name.
- Teachers are associated with attributes netID (a unique teacher ID) and name.
- Courses are associated with attributes courseID (unique) and name.
- Each course is taught by exactly one teacher.
- Each student is enrolled in at least one course.

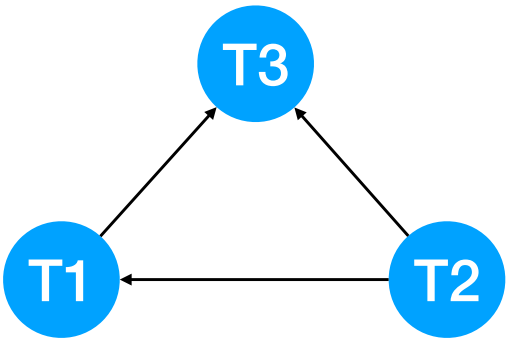
(10 points)



Part D) Concurrency Control. (20 points)

D.1) Draw the conflict graph for the following schedule and conclude whether it is conflict-serializable or not (assuming that all transactions commit).
(5 points)

W2(A) R1(A) W2(B) R3(A) R2(A) W3(A) R3(B)



There are no cycles, hence the schedule is conflict-serializable.

D.2) Is the following schedule view-serializable (assuming that all transactions commit)? Justify in one or two sentences.
(5 points)

R1(A) W2(B) R3(C) W3(A) W2(C) R1(A)

The first read of A by T1 sees the value before schedule start, the second read of A by T1 sees the value written by W3. This is not possible in any serial schedule and the schedule is therefore not view-serializable.

D.3) Is the following schedule recoverable? Justify in one or two sentences. (5 points)

W3(B) W1(A) C3 R2(A) C2

T2 reads the value of A that was written by T1 and T2 commits before T1 has committed. Hence, the schedule is non-recoverable.

D.4) Is the following schedule strict? Justify in one or two sentences. (5 points)

W1(A) W2(B) R1(B) C2 W3(C) W1(B)

It is not strict (it does not even avoid cascading aborts since T1 reads an uncommitted value of B).

Part E) Logging and Recovery. (20 points)

We use the ARIES algorithm for recovery. After the system comes back online after a crash, the following log entries are available for recovery (since no checkpoints have been written so far, the analysis phase starts with an empty dirty page table and empty transaction table and considers log entry number 10 first):

10 T1 writes P2
20 T3 writes P1
30 T1 writes P3
40 T1 writes P5
50 T1 aborts
60 T2 writes P2
70 CLR T1 P5

E.1) State the entries in the dirty page table after the analysis is complete (each entry consists of a page ID and the recLSN number).
(5 points)

<P2, 10> <P1, 20> <P3, 30> <P5, 40>

E.2) State the entries in the transaction table after the analysis phase is complete (each entry consists of a transaction ID, a transaction status, and the lastLSN number).
(5 points)

<T1, Aborted, 70> <T2, Running, 60> <T3, Running, 20>

E.3) Assume that pages on disk are associated with the following pageLSN counters:

PageID	pageLSN
P1	20
P2	10
P3	0
P5	0

Based on the pageLSNs and your dirty page table from before, state for each log entry whether it is redone in the redo phase (justify with one sentence per entry).
(5 points)

LSN 10 - not redone since P2.pageLSN >= LSN
LSN 20 - not redone since P1.pageLSN >= LSN
LSN 30 - is redone since P3.recLSN <= LSN and pageLSN < LSN
LSN 40 - is redone since P5.recLSN <= LSN and pageLSN < LSN
LSN 50 - nothing to redo
LSN 60 - redone since P2.recLSN <= LSN and pageLSN < LSN
LSN 70 - redone since P5.recLSN <= LSN and pageLSN < LSN

E.4) State the CLR entries that are written during the undo phase (write them in the same order as they appear in the log and use the same format for CLR entries as in the log above).
(5 points)

CLR T2 P2
CLR T1 P3
CLR T3 P1
CLR T1 P2

Part F) MapReduce. (20 points)

We are given a set of key-value pairs representing Web links. The key is the source Web site (i.e., the Web site on which the link appears) and the value is the target Web site (i.e., the Web site to which the link leads). Write pseudo-code for a MapReduce program that counts for each Web site how often it is linked to from Facebook (i.e., how often it appears as target for links where the source address starts with www.facebook.com). The output is a set of key-value pairs where the key is the target Web site and the value is the number of references on Facebook.

Example Input:
<www.facebook.com/IT, www.cornell.edu>
<www.facebook.com/JG, www.microsoft.com>
<www.facebook.com/JG, www.cornell.edu>
<www.cornell.edu, www.google.com>

Example Output:
<www.cornell.edu, 2>
<www.microsoft.com, 1>
<www.google.com, 0>

The output must contain a key-value pair for each Web address that appears as link target in the input.

On the following page, write pseudo-code for the map function and for the reduce function. Your pseudo-code may use the following auxiliary functions:
- `emit(K,V)` : emits key-value pair `<K,V>` as result in map or reduce function
- `isFB(W)` : returns true if Web address `W` starts with www.facebook.com

Beyond those functions, your code may only use variable declarations (e.g., “int sum;”), value assignments (e.g., “sum = 0;”), arithmetic operators (+, -, *, /), and simple control flow structures (if statements, for and while loops).

```
map(String source, String target):
        if (isFB(source)):
                emit(target, 1);
        else:
                emit(target, 0);

reduce(String target, IntIterator counts):
        int sum = 0;
        for count in counts:
                sum += count;
        emit(target, sum);
```

CS4320 Final Exam

This page will be used for grading your exam. Do not write anything on this page.

SECTION	QUESTION	SCORE	SECTION TOTAL
Part A	A.1 (Max: 5 points)		(Max: 20 points)
	A.2 (Max: 5 points)		
	A.3 (Max: 5 points)		
	A.4 (Max: 5 points)		
Part B	B.1 (Max: 5 points)		(Max: 20 points)
	B.2 (Max: 5 points)		
	B.3 (Max: 5 points)		
	B.4 (Max: 5 points)		
Part C	C.1 (Max: 5 points)		(Max: 20 points)
	C.2 (Max: 5 points)		
	C.3 (Max: 10 points)		
Part D	D.1 (Max: 5 points)		(Max: 20 points)
	D.2 (Max: 5 points)		
	D.3 (Max: 5 points)		
	D.4 (Max: 5 points)		
Part E	E.1 (Max: 5 points)		(Max: 20 points)
	E.2 (Max: 5 points)		
	E.3 (Max: 5 points)		
	E.4 (Max: 5 points)		
Part F	F.1 (Max: 20 points)		(Max: 20 points)
Total (Max: 120 points)			