

# CompSci 516: Database Systems

Final

Fall 2018

This booklet has 18 pages (including the cover page and two blank pages at the end).

You can use the reverse sides as additional space for writing your answers.

## INSTRUCTIONS

1. No external help (books, notes, laptops, tablets, phones, etc.) or collaboration is allowed.
2. You have **180 minutes** to answer questions that add up to **100 points**. i.e. you have about 9 mins for 5 points, 18 mins for 10 points, and about 36 mins for 20 points.
3. Some questions may need more time, some questions less. Please budget your time accordingly.
4. If you cannot solve a problem fully, write partial solution for partial credit. **Even if explanations are not sought in a question, if you cannot fully solve a problem, writing down your thought process may lead to some partial credit.**
5. Do not spend too much time on a problem that you find difficult to solve - move on to other problems.
6. **The problems are organized in no particular order, easier problems may appear later.**

**\*\*Write Your Name Here\*\* (1 bonus point):**

First Name:

Last Name:

All the best!

**DO NOT WRITE BELOW THIS LINE**

---

Problem 1	/ 12	Problem 2	/ 32	Problem 3	/ 14
Problem 4	/ 20	Problem 5	/ 5	Problem 6	/ 10
Problem 7	/ 7	bonus point	/ 1	Total	/ 100

## Q1. (12 pts) Short Q/A

Q1a. (9 \* 1 = 9 pts) Write True or False.

No explanations are needed.

1. If  $X \subseteq Y$ , then  $X^+ \subseteq Y^+$ : (**True/False**) (Recall:  $X^+$  is the closure of attributes in  $X$ ).
2.  $(X^+)^+ = X$ : (**True/False**)
3.  $(X \cup Y)^+ = X^+ \cup Y^+$ : (**True/False**)
4. In non-quiescent logs, new transactions can start between START CKPT and END CKPT: (**True/False**)
5. Nested loop join always returns the same results as hash join: (**True/False**)
6. Given an instance of a relation, we can infer all functional dependencies that HOLD for that relation schema: (**True/False**)
7. Static hashing is a better choice than dynamic hashing when there are frequent updates and skew in the data: (**True/False**)
8. In frequent itemset mining, an itemset  $\{A, B, C\}$  has larger *support* (no. of transactions containing this itemset) than the itemset  $\{A, B, C, D\}$ : (**True/False**)
9. In two-phase commit protocol, any site can unilaterally decide to commit, but needs consensus to abort: (**True/False**)

**Q1b. (3 \* 1 = 3 pts) Write short answers (one/two sentence(s) should suffice).**

1. Consider the following relation instance  $R(A, B, C)$ . Fill out the table saying: **Yes** (the decomposition is lossless), or **No** (it is not lossless). No explanations are needed.

R

A	B	C
a1	b1	c1
a2	b1	c2

Decomposition	Lossless? (Yes/No)
$R1(A, B)$ and $R2(B, C)$	
$R1(A, C)$ and $R2(A, B)$	

2. Write **one** key difference between OLAP and OLTP.

3. Write **one** key difference between NOSQL and traditional RDBMS.

## Q2. (32 pts) Miscellaneous

### Q2a. (3 + 3 = 6 pts)

Consider three relations  $R(A,B), S(B,C), T(C,A)$ , each with  $N$  (distinct) tuples.

(I) What is the maximum possible number of tuples in the natural join  $Q(A,B,C) = R \bowtie S \bowtie T$ ?

Choose the correct answer from (i)  $N/3$ , (ii)  $N$ , (iii)  $N^2$ , (iv)  $N^3$ :

Briefly explain your answer.

(II) Suppose  $A$  is key in  $R(\underline{A}, B)$ .

What is the maximum possible number of tuples in the natural join  $Q(A,B,C) = R \bowtie S \bowtie T$ ?

Choose the correct answer from (i)  $N/3$ , (ii)  $N$ , (iii)  $N^2$ , (iv)  $N^3$ :

Briefly explain your answer.

### Q2b. (4 pts)

Consider two relations  $R(A,B,C)$  and  $S(D,E,F)$  satisfying the following functional dependencies:

$$R : A \rightarrow B$$

$$S : DE \rightarrow F$$

Let  $T(A,B,C,D,E,F)$  be the following table:

```
create table T as
  (select R.A, R.B, R.C, S.D, S.E, S.F
   from R, S
   where R.C = S.D and S.E = 5)
```

Find the all the key(s) of  $T$  and list below:

**Q2c. (4 pts)**

Consider an employee relation  $Emp(eid, name, age)$  stored in a heap file, and the following query:

```
SELECT name
FROM Emp
WHERE age > X
```

Suppose the range of the *age* of employees is 21 – 120, and suppose the employees are uniformly distributed for this entire age range.

Suppose you have the choice to use

- file scan (i.e. no index), or
- a combination of
  - clustered / unclustered
  - B+-tree / hash-based index
  - on any of the attributes *eid* / *name* / *age*.

Assume that **one index lookup requires 4 I/O and, 1 page can hold about 1% Emp tuples.**

(i) Which index combination (or no index) will you choose when  $X = 117$ ? No explanations are needed.

(ii) Which index combination (or no index) will you choose when  $X = 21$ ? No explanations are needed.

**Q2d. (4 pts)**

Suppose you have  $N = 20$  pages for a relation. Number of buffers in memory  $B = 4$ . Fill out the following table for the number of sorted runs and I/O cost in each pass of an **external merge sort** algorithm (for pass  $= 0, 1, 2, \dots$ ).

- DO NOT count the final cost of writing the result to disk.
- No explanations are necessary.
- All rows may not be needed and add more rows if necessary.
- Hint: you do not need to remember any formula to solve this.

Pass	No. of sorted runs after the pass	I/O cost in the pass

**Q2e. (3 pts)**

Fill out the blanks each with one of the options below:

Under \_\_\_\_ (a) \_\_\_\_\_ policy, **committed** transactions need to \_\_\_\_ (b) \_\_\_\_\_ in the event of a crash to ensure \_\_\_\_ (c) \_\_\_\_\_.

- (a) (i) STEAL, (ii) NO FORCE
- (b) (i) REDO, (ii) UNDO
- (c) (i) ATOMICITY, (ii) DURABILITY

**Q2f. (4 pts)**

Consider two relations  $R(A, B)$  and  $S(B, C)$ , and the following query:

```
SELECT *  
FROM R, S  
WHERE R.B < S.B
```

Consider the following join algorithms:

- Block-nested loop join
- Index-nested loop join with B+-tree index on  $R$
- Index-nested loop join with hash index on  $R$
- Index-nested loop join with B+-tree index on  $S$
- Index-nested loop join with hash index on  $S$
- Sort-merge join
- Hash join

(i) Which join algorithm is likely to perform the best assuming  $R$  is outer and  $S$  is inner relation? Select one choice – no explanations are needed.

(ii) Pick any one option from the above that cannot be used at all.

**Q2g. (2 pts)**

Given a relation  $R(A, B)$ , write a query to **select “distinct” values** of  $A$  **NOT** using the keyword “DISTINCT”.

**Q2h. (2 pts)**

Given two relations  $R(A,B)$  and  $S(B,C)$ , the following VIEW  $V$  has been defined. When Alice was asked to compute the average value of  $B$  for each value of  $A$  from the relation  $R$ , she wrote the query  $Q$ . Both  $V$  and  $Q$  are shown below. Is her query correct? Explain your answer.

```
V:
CREATE VIEW V1 AS
SELECT A, C, SUM(B) AS s, COUNT(B) as c
FROM R, S
WHERE R.B = S.B
GROUP BY A, C
```

```
Q:
SELECT A, (SUM(s) * 1.0) / SUM(c)
FROM V1
GROUP BY A
```

**Q2i. (3 pts)**

The table

*Scores(Team, Day, Opponent, Runs)*

gives the scores in the Japanese Baseball League for two consecutive days. The Opponent is NULL if the Team did not play on that day. The number of Runs is given as NULL if either the team did not play, or will play on that day but the game is not yet concluded. The data in this table is as follows:

Team	Day	Opponent	Runs
Tigers	Sunday	Bay Stars	9
Carp	Sunday	NULL	NULL
Dragons	Monday	Carp	NULL
Swallows	Monday	Giants	0

What will be the output of the query  $Q$ ?

```
Q:
SELECT Team, Day
FROM Scores
WHERE Opponent IS NULL
      OR NOT (Runs >= 2)
```



### Q3. (7 + 7 = 14 pts) RC + SQL

Suppose a university maintains the job interview history of its students as follows.

- Students(sid, sname, dept) (*Student id, names, and department of the students*).
- Employers(eid, ename, location)  
(*Employer id, name, and location of the companies that are recruiting/recruited in the past*)
- Interviews(eid, sid, year) (*Which company interviewed which student in what year*)
- Offers(eid, sid, year, salary) (*Which company made an offer to which student in what year, and the annual salary in the offer*)

Assume that

- A company can interview or make an offer to a student at most once in a year.
- A company can be located in exactly one location (clearly not true in practice), and
- if an (*eid*, *sid*) pair appears in the Offers table, it also appears in the Interviews table, i.e. no students get an offer without an interview (ok, this is mostly true in practice).

#### Q3a. (7 pts) TRC

Write a **TRC expression** to output “the names (*sname*) of all students who had an interview with **all** companies from *location* = ‘CA’ that appear in the *Employer* table, but did not get an offer from any of these companies (the student may or may not have interviews with/get offers from companies from other locations).

e.g. If student *Y* appears in the answer, and if ‘*G*’, ‘*T*’, ‘*A*’, ‘*F*’ are all the companies in ‘CA’, then *Y* had interviews from all four of them, but offers from none of them. *Y* could have an interview/offer from another company ‘*M*’ that is located in *location* = ‘WA’.

### Q3b. (7 pts) SQL

Write a SQL query for the problem in Q3a, i.e.:

Write a **SQL query** to output “the names (*sname*) of all students who had an interview with **all** companies from *location* = ‘CA’ that appear in the *Employer* table, but did not get an offer from any of these companies).

Only one query is allowed, but it can have nested subqueries, or subqueries using “WITH” clauses.

The schema is repeated below:

- Students(sid, sname, dept)
- Employers(eid, ename, location)
- Interviews(eid, sid, year)
- Offers(eid, sid, year, salary)

## Q4. (20 pts) Transactions

### Q4a. (3 pts)

Consider the following transaction schedule of  $T_1, T_2, T_3$ :

$R_1(X), R_2(X), R_3(X), R_1(Y), W_2(Z), R_3(Y), W_1(Z), W_1(Y)$

Find out **all** serial schedules that the above schedule is conflict-equivalent to.

(Hint: you should be able to find the answer(s) without having to go over all possible serial schedules on  $T_1, T_2, T_3$ ).

### Q4b. (6 pts)

Consider three transactions  $T_1, T_2, T_3$  with the following sequences of operations:

- $T_1 : R_1(A), W_1(B)$
- $T_2 : R_2(B), W_2(D)$
- $T_3 : R_3(C), W_3(A)$

Consider a Timestamp Ordering scheduler named **TO** in which the three transaction  $T_1, T_2$ , and  $T_3$  have timestamps 60, 55, and 70 respectively. Assume initially all objects had timestamp 50. **Ignore commit bit (C) for all objects and transactions.**

Consider a Two-Phase Locking scheduler named **2PL**, also operating on the same set of data objects.

Examine the following two schedules:

- $S_1 : R_1(A), R_3(C), W_3(A), R_2(B), W_2(D), W_1(B)$
- $S_2 : R_3(C), R_1(A), W_1(B), R_2(B), W_2(D), W_3(A)$

**Determine which of the above schedules can be produced by TO and which can be produced by 2PL by filling out Yes/No in the following table.**

Explanations are not required.

Schedule	Can be produced by TO?	Can be produced by 2PL?
$S_1$		
$S_2$		

**Q4c. (3 + 1 + 1 + 2 + 1 + 2 + 1 = 11 pts)**

At the time of a system crash, let the log segment (in the **UNDO/REDO** logging scheme) involving four transactions  $S, T, U, V$  be as follows. Answer the following questions using this log.

1. (START S)
2. (S, X, 10, 20)
3. (COMMIT S)
4. (START T)
5. (T, X, 20, 30)
6. (START CKPT(T) )
7. (T, Y, 10, 20)
8. (START U)
9. (COMMIT T)
10. (U, X, 30, 40)
11. (END CKPT)
12. (U, Y, 20, 30)
13. (START V)
14. (START CKPT(U, V) )
15. (COMMIT U)
16. (V, Y, 30, 40)

(i) (3 pts) Fill out the table below by writing **Guaranteed** or **Maybe** or **Impossible** for each updates, which says whether that updated value of the variable was written to disk by the time of the crash:

- **Guaranteed:** this update is guaranteed to be written to disk,
- or, **Maybe:** this update might have been written to disk,
- or, **Impossible:** this update could not be written to disk.

log sequence no.	Variable = value	Status
2	$X = 20$	
5	$X = 30$	
7	$Y = 20$	
10	$X = 40$	
12	$Y = 30$	
16	$Y = 40$	

Suppose during the recovery, we first do REDO, and then UNDO.

(ii) **(1 pts)** What is the earliest (smallest) Log Sequence Number that is relevant for recovery in the REDO phase?

(iii) **(1 pts)** What is the earliest (smallest) Log Sequence Number that is relevant for recovery in the UNDO phase?

(iv) **(2 pts)** What are the updates (in order) in the recovery for REDO?

(v) **(1 pts)** What are the updates (in order) in the recovery for UNDO?

(vi) **(2 pts)** What are the final values of variables  $X, Y$  after both REDO and UNDO recovery phases are done?

- $X =$

- $Y =$

(vii) **(1 pts)** Will we get the same answer if we had done UNDO first and then REDO? (Write yes/no)

### Q5. (5 pts) Recursion

Recall the standard Datalog program for reachability in a ‘directed’ graph where the directed edges are stored in the  $E(U, V)$  relation (i.e.,  $E(1, 2)$  and  $E(2, 1)$  are two different edges, and none, either, or both of them can exist) :

- $R(x, y) \text{ :- } E(x, y)$
- $R(x, y) \text{ :- } E(x, z), R(z, y)$

Write a datalog program to output the pairs of vertices  $u, v$  such that there is no path from  $u$  to  $v$ . Recall that for your query to be correct, it has to be ‘safe’. Mention the output relation (in the above example,  $R$  is the output).

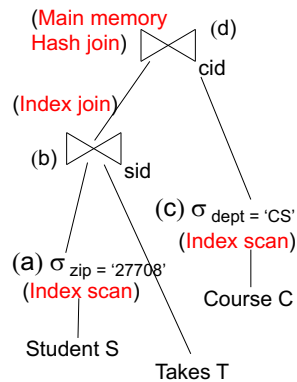
## Q6. (10 pts) Query Cost Computation

Consider the relations below ( $T$  has foreign keys to  $C$  and  $S$ ):

- Student:  $S(\underline{\text{sid}}, \text{name}, \text{zip})$ , Takes:  $T(\text{sid}, \text{cid}, \text{year})$ , Course:  $(\underline{\text{cid}}, \text{title}, \text{dept})$

Recall that  $B(R)$  = no. of pages in  $R$ ,  $T(R)$  = no. of tuples in  $R$ ,  $V(R, A)$  = no. of distinct values of  $R.A$ .

- $B(S) = 20$ ,  $T(S) = 2,000$ ,  $V(S, \text{zip}) = 100$ .
- $B(T) = 10,000$ ,  $T(T) = 100,000$ ,  $V(T, \text{sid}) = 2,000$ ,  $V(T, \text{cid}) = 50$ .
- $B(C) = 10$ ,  $T(C) = 100$ ,  $V(C, \text{dept}) = 1$ .



Assume the size of the main memory is  $M = 2000$  pages, and all indexes are unclustered and are stored in main memory (hence accessing them requires zero disk I/O's).

Write the (i) the no. of output tuples  $P$  and (ii) the I/O cost of each node  $C$  in the table below. You can show calculations in terms of  $B$ ,  $T$ ,  $V$  for partial credit.

Step	No. of output tuples $P$	I/O cost $C$
(a)		
(b)		
(c)		
(d)		
Total	final output size =	total cost =

## Q7. (7 pts) Efficient Join Evaluation for Special Queries

You are given  $k$  relations  $R_1(A_1, A_2), R_2(A_2, A_3), R_3(A_3, A_4), \dots, R_k(A_k, A_{k+1})$ . Each relation has  $n$  tuples. The values of each attribute  $A_i$  come from an arbitrary domain.

Note that the size of the join  $R_1 \bowtie R_2 \bowtie R_3 \bowtie \dots \bowtie R_k$  can be as large as  $2^k$  or bigger, e.g. if all the relations  $R_i$  have the tuples shown in the table below:

$R_i$

$A_i$	$A_{i+1}$
0	a
0	b
1	a
1	b
a	0
a	1
b	0
b	1

i.e. whatever join algorithm you use, to compute the join, you need to spend at least  $2^k$  cost (assume CPU cost for actual computation, ignore pages and disk I/Os).

However, suppose your query is

$$\pi_{\{\}}(R_1 \bowtie R_2 \bowtie R_3 \bowtie \dots \bowtie R_k)$$

The projection is on to empty set, i.e. this query returns true or false. It returns true if there are  $k$  tuples in these relations of the form  $R_1(a_1, a_2), R_2(a_2, a_3), \dots, R_k(a_k, a_{k+1})$  that join with each other, and false if no such combination exists.

**Task:** Briefly describe an  $O(kn \log n)$  time algorithm (and explain its time complexity) to compute the above query that checks if there is “path” of tuples that join with each other in these relations (no need to output the paths or combinations). Your algorithm should give  $O(kn \log n)$  time even for the above input.

**Hint:** Recall: (i) sorting  $n$  tuples on any attribute takes  $O(n \log n)$  time, (ii) The notation  $O(N)$  means that there is a constant  $c$  such that asymptotically if you increase  $N$ , the time is bounded by  $\leq cN$ . One concept that you have learned in this course in another context would help. **(Use the back of the page too if more space is needed.)**



**Blank additional page to be used for rough calculations. DO NOT write solutions here.**

**Blank additional page to be used for rough calculations. DO NOT write solutions here.**