The University of British Columbia

Computer Science 304

Practice Final Examination

Instructor: **Time**: 2.5 hours **Total marks**: 100

Name_____ **Student No**_____

(PRINT) (Last) (First)

Signature_____

This examination has 13 pages. Check that you have a complete paper.

This is a closed book exam. Notes, books or other materials are not allowed.

Answer all the questions on this paper. Give **short but precise** answers. Always use point form where it is appropriate. The marks for each question are given in {}. Use this to manage your time.

Good Luck

READ AND OBSERVE THE FOLLOWING RULES:

1. Each candidate should be prepared to produce, upon request, his or her Library/AMS card.

2. No candidate shall be permitted to enter the examination room after the expiration of one-half hour, or to leave during the first half-hour of the examination.

3. Candidates are not permitted to ask questions of the invigilators, except in cases of supposed errors or ambiguities in examination questions.

4. **CAUTION** --- Candidates guilty of any of the following, or similar, dishonest practices shall be immediately dismissed from the examination and shall be liable to disciplinary action.

    a. Making use of any books, papers or memoranda, calculators or computers, audio or visual cassette players, or other memory aid devices, other than those authorized by the examiners.

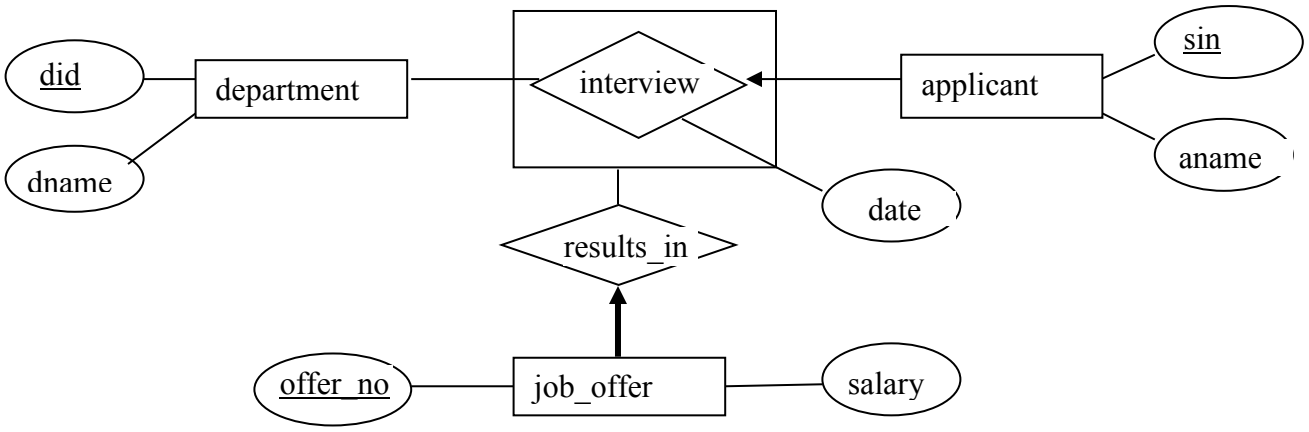
    b. Speaking or communicating with other candidates.

    c. Purposely exposing written papers to the view of other candidates. The plea of accident or forgetfulness shall not be received.

Question 1. {26 marks}

Answer the following questions in the space provided. Give short but complete answers.

a. {6 marks} The following ER diagram shows the interviews and the job offers that are made by the different departments of a company:



where *sin*, *aname*, *did*, *dname* are the id's and names of an applicant and a department, and *offer_no* is an id of an offer.

List a **minimum number of tables** we need to create in order to record this information. For each table, list its attributes and underline its primary key.

Department(did, dname)

Applicant(sin, aname, did, date)

JobOffer(offerNo, salary, sin)

b. {3 marks}What are the benefits and disadvantages that the **strict two-phase locking** protocol provides, compared to the **regular two-phase locking**?

Benefits:

Schedules are recoverable
Avoids cascading aborts

Disadvantages:

Restricts concurrency

c. {11 marks} Consider the table

T(A, B, C, D, E, F, G)

and the set of functional dependencies

fd1: A B → C
fd2: A → D E
fd3: B → F
fd4: E → G

(i) {3 marks} List all the candidate keys for this table. Show that what you find are all and the only keys.

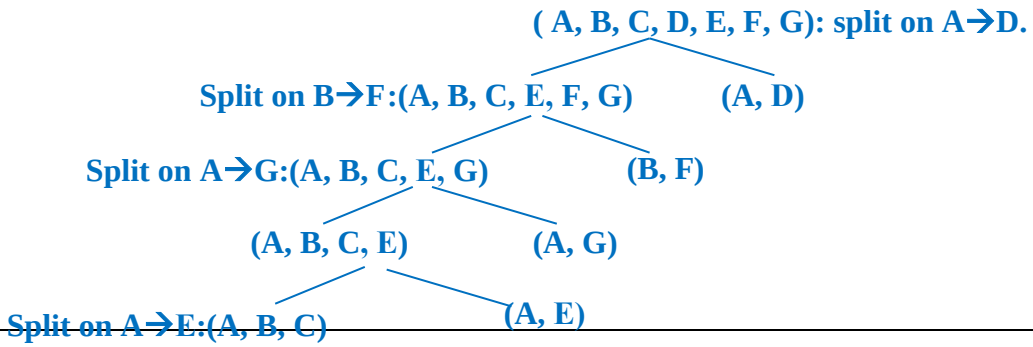
A and B do not appear on the RHS of any FD and so must be present in every key. Check $(AB)^+$ = ABCDEFG, so AB is a super key. Since every key must contain A and B, it follows AB is the unique key of the given relation scheme T.

(ii) {2 marks} Is this table in 3NF? Justify your answer.

No. All FD's except the first violate 3NF.

(ii) {6 marks} Is this table in BCNF? If you think it is not in BCNF, decompose T into a **minimum number** of tables that are in BCNF.

No. All FD's except the first violate BCNF. Need to split:



H

Hence our final answer is R1(A,D), R2(B,F), R3(A,B,C), R4(A,G), R5(A,E)

d. {4 marks} Find a 3NF decomposition of T.

First, find a minimal cover of the given FD set. First, write every FD in standard form: AB→C, A→D, A →E, B→F, E→G. Second, try to eliminate attributes from the LHS w/o changing the closure. It can be seen that no attributes can be eliminated. Third, check if any FDs are redundant. No FD can be eliminated. So the above set of FDs is minimal.

To decompose, decompose all the way down to BCNF. This is as above. Now we need to check to see if there are any FDs that we have lost - R3 has A, B, and C. R1 has A and D. R5 has A and E. R2 has B and F. However, there is no relation that contains both E and G, so we need to add in a new relation R6(E,G)

e. {2 marks} Consider the following database about the employees and the hours they work in each department:

```
employee(eid, ename, ecity)
dept(did, dname, dcity)
works(eid, did, hours)
```

and the Relarional Algebra query:

$(\pi_{eid,\ did}\ (works) \div \pi_{did}\ (\sigma_{dcity='Vancouver'}\ (dept))\ )\ -\ \pi_{eid}\ (\sigma_{ecity='Vancouver'}\ (employee))$

Circle the sentence that **best** describes the result of this query

a) Employees who live in Vancouver, but not work in any Vancouver department

b) Employees who work in some Vancouver department, but do not live in Vancouver

c) Employees who live in Vancouver, but not work in all Vancouver departments

d) Employees who work in every Vancouver department, but do not live in Vancouver

e) None of the above

Question 3. {24 marks, 6 marks each}

Consider the following database:

```
Book(id, title, author, publisher, year, price)
Bookstore(name, address, city)
Sold(name, id, quantity)
```

which records how many copies of each book have been sold by each bookstore. In these tables, a book is identified by its id and each bookstore is identified by its name. Only the first author for each book is recorded in the book table. The quantity attribute in Sold is at least 1 (i.e. quantity >=1).

Write an expression for each of the following queries in the query language indicated.

a. Find the names of the bookstores that are located in Vancouver and have sold some book published after the year 2000 (quantities do not matter). Write in RA.

$$\pi_{name}(\sigma_{city=`Vancouver' \wedge year>2000}(\\ (book \bowtie_{book.id=sold.id} sold) \bowtie_{bookstore.name=sold.name} bookstore)).$$

b. Find the names of the bookstores that have sold every book which is sold by the bookstore named "Chapters" (quantities do not matter). Write in Datalog and SQL.

$$ans(N) \leftarrow bookstore(N,\_,\_),\ \text{not}\ bad(N).$$
$$bad(N) \leftarrow sold(`Chapters', I, \_),\quad bookstore(N,\_,\_),\ \text{not}\ sale(N,I).$$
$$sale(N,I) \leftarrow sold(N,I,Q).$$

```
SELECT name
FROM Bookstore B
WHERE  NOT EXISTS
        ( SELECT id
          FROM Sold
          WHERE name =  "Chapters" )
       EXCEPT
       (SELECT id
        FROM Sold S
        WHERE S.name =  B.name )
```

c. Find the names of the bookstores that have sold at least 100 different books (of any quantity) published after the year 2000.

```
SELECT name

FROM Sold S,  Book B

WHERE S.id = B.id  AND B.year >  2000

GROUP BY name

HAVING  count(DISTINCT  id ) >= 100
```

d. Find the names of the Vancouver bookstores which have sold more books than the bookstore named "Chapters". That is, the total quantity of books sold by each of these bookstores is greater than the total quantity of books sold by the bookstore named "Chapters".

```
SELECT name

FROM Sold S, Bookstore B

WHERE S.name =B,name AND B.city = 'Vancouver"

GROUP BY name

HAVING  sum( quantity ) >

                ( SELECT  sum( quantity )

                 FROM Sold

                 WHERE name = "Chapters" )
```

Question 3. {12 marks, 6 marks each}

Consider again the bookstore database of the previous question:

```
book(id, title, author, publisher, year, price)
bookstore(name, address, city)
sold(name, id, quantity)
```

a. Write an SQL query that is equivalent to the following Datalog query:

$$ans(T, A) \leftarrow book(I, T, A, `Pearson', \_, \_), \neg bad(I).$$
$$bad(I) \leftarrow book(I, \_, \_, \_, \_, P), book(I', \_, \_, \_, \_, P'),\ P' > P.$$

SELECT B.title, B.author

FROM Book B

WHERE B.publisher = "Pearson"

AND NOT EXISTS

(SELECT *

FROM Book B2

WHERE B2.price > B.price)

c. Write one or more SQL statements that will increase the price of any book sold by the bookstore named “Chapters” by 10% and those books sold by the bookstore named “Indigo” by 5%, with the restriction that if a book is sold by both bookstores, its price is increased only by 10%.

```
UPDATE Book
SET price = price * 1.10
WHERE id IN
        ( SELECT id
          FROM Sold
          WHERE name = 'Chapters' )

UPDATE Book
SET price = price * 1.05
WHERE id IN
        ( SELECT id
          FROM Sold
          WHERE name = 'Indigo' )
        EXCEPT
        ( SELECT id
          FROM Sold
          WHERE name = 'Chapters' )
```

Question 4. {12 marks, 6 marks each}

For each of the following schedules determine whether the schedule is conflict-serializable or view-serializable. Justify your answers.

a.

T1	T2	T3
		write(C)
	write(B)	
	read(B)	
		read(C)
write(C)		
	read(C)	

Is it conflict-serializable?

WHY?

Is it view-serializable?

WHY?

b.

T1	T2	T3
	read(C)	
write(C)		
	write(C)	
read(B)		
		write(C)

Is it conflict-serializable?

WHY?

Is it view-serializable?

WHY?

Question 5. {10 marks }

Consider the following schedule:

T1	T2	T3	Operation#
read(C)			1
		read(B)	2
read(A)			3
	read(C)		4
		write(B)	5
write(A)			6
	read(B)		7
		read(A)	8
	write(C)		9
	write(B)		10
commit			11
	commit		12
		commit	13

a. {6 marks} Can this schedule be produced by the **regular two-phase locking** protocol without lock upgrades ? Justify your answer. (Ether show that the protocol is violated or show when the release phase for each transaction starts).

b. {2 marks} Can this schedule be produced by the **strict two-phase locking** protocol ? Justify your answer.

c. {2 marks} Is this schedule a **recoverable schedule**? Justify your answer.

Question 6. {6 marks }

Suppose that at some time during the execution of a database system we have the transaction scenario shown on the following table:

Transaction	Data items locked by transaction	Data items transaction is waiting for
T1	B	A, C
T2	C	D,E
T3	E, F	G
T4	D	A
T5	A, G	C

Determine whether a deadlock exists and justify your answer.

Question 7. {10 marks }

Suppose our DBMS uses 2PL with shared and exclusive locks, and the WAL logging strategy.

For simplicity, assume that the update records in the log have the format:
<Previous LSN, Transaction, "U", Variable, BeforeValue, AfterValue>
that the CLR record format is:
<Previous LSN, Transaction, "CLR", Undone Record LSN, Variable, BeforeValue, undoNextLSN>
and abort , commit and end records have the form
<Previous LSN, Transaction, action >
where action is "**abort**", "**commit**" or "**end**", respectively.

For the following sequence of actions, show the log entries that would be generated. You may assume that the log sequence numbers start at **1**, and increase by **1**. Use **0** to indicate that there is no previous record. For each action, enter the log records produced by the action on the right of the action statement. Some actions may not produce any log records, while other actions may produce more than one record.

ACTION	LSN	LOG RECORD
Start checkpoint	**1**	**< start_checkpoint >**
End checkpoint	**2**	**<end_checkpoint >**
T1 locks A		
T1 changes A from 10 to 20	**3**	**< 0, T1, U, A, 10, 20 >**
T1 locks B		
T1 unlocks A		
T1 changes B from "ABC" to "AAA"	**4**	**< 3, T1, U, B, ABC, AAA >**
T1 unlocks B		
T1 aborts	**5**	**< 4, T1, abort >**

T2 locks B		
T2 changes B from “AAA” to “BBB”	6	< 0, T2, U, B, AAA, BBB >
T2 unlocks B		
T1-Abort locks B // T1 abort // operation starts here		
T1-Abort locks A		
T1-Abort resets B back to “ABC”	7	< 5, T1, CLR, 4, B, ABC, 3 >
T3 locks C		
T3 changes C from 12 to 14	8	< 0, T3, U, C, 12, 14 >
T1-Abort resets A back to 10	9 10	< 7, T1, CLR, 3, A, 10, 0 > < 9, T1. end >
T2 commits	11	< 6, T2, commit >
T3 unlocks C		
T3 commits	12	< 8, T3, commit >
Log records for T2 have been written out	13	< 11, T2. end >
Log records for T3 have been written out	14	< 12, T3. end >