Student number:_____

UNIVERSITY OF VICTORIA
Faculty of Engineering
Department of Computer Science

26 June 2015

CSC 370 (Database Systems)
Instructor: Daniel M. German

Duration: 50 minutes

This is a closed-book exam. You are only allowed one, hand written, letter-size sheet of paper.

This examination paper consists of **7** pages and **4** questions. Please bring any discrepancy to the attention of an invigilator. The number in brackets at the start of each question is the number of points the question is worth.

Answer all questions on exam paper.

Please write your answers clearly.

For instructor's use:

	Score
1 (15)	
2 (5)	
3 (5)	
4 (5)	
Total (30)	

Feel free to remove this page from the exam.

For this exam, consider the following schema and instances of the relations. Broadly speaking, **Emp** records employees of an organization, **Dept** departments of the organization, and **Works** records which employees work for a given department. **managerid** corresponds to the **eid** of the manager of a department.

This is only a sample instance of the database. Attributes with an empty value should be assumed to be NULL.

`Emp(eid: integer, ename: string, email: string, salary: real)`

- Primary Key: **eid**

eid	*ename*	*age*	*salary*
11564812	John Williams	35	74098
15487874	Gene Edwards	51	41008
15645489	Daniel Evans	25	
51135593	Maria White	22	24998
54879887	Dorthy Lewis	33	

`Dept(did: integer, dname: string, budget: real, managerid: integer)`

- Primary Key: **did**
- For every tuple in **Dept** , the managerid should be a valid employee id (*eid*).

did	*dname*	*budget*	*managerid*
1	Hardware	100000	15645489
5	Software	200000	15487874
7	Marketing	150000	

`Works(eid: integer, did: integer, pct_time: integer)`

- Primary Key: **eid, did**
- For any tuple in **Works**, its *eid* should exist in**Emp**, and its *did* should exist in **Dept**.

eid	*did*	*pct_time*
11564812	1	100
15487874	5	60
15645489	5	40

1. **RA and SQL**

Given the database schema in page two, write queries to answer the following questions. Your solutions should include both Relational Algebra and its equivalent SQL. If the SQL does not match the Relational Algebra, the answer will be considered incorrect. You can assume $E = Emp$, $D = Dept$ and $W = Works$.

(a) [3] For every department in **Dept** list the number of employees who work in it. The result should include even departments with zero employees. Your result should have two attributes: *did* and *count* (number of employees working in it).

$\gamma^{did}_{count(eid)} (D \overset{\circ}{\bowtie} W)$

```
SELECT did, count(eid)
FROM D NATURAL FULL JOIN W
GROUP BY did
```

(b) [3] List the *eid*, the *ename*, and *salary* of the employee(s) that earn the highest salary. Your result should have three attributes.

$M = \gamma_{max(salary) \to salary} E$

$\pi_{eid, ename, salary} \; E \bowtie M.$

```
WITH M AS ( SELECT max(salary) AS salary FROM E)
    SELECT eid, ename, salary FROM
        E NATURAL JOIN M;
```

(c) [3] List the number of employees in **Emp** that have a salary that is not NULL and the number of different employees working in at least one department (according to table **Works**). Your result should consist of exactly one tuple with two attributes; the first attribute is the number of employees without a salary (call it *nonsalary*) and the second is the number of employees working in at least one department (call it *atleastone*).

$NS = \gamma_{count(*) \to nonsal} \sigma_{salary \text{ is NULL}} E$

$WO = \gamma_{count(distinct\ eid) \to atleastone} W$

```
NS x WO
WITH NS AS (SELECT count(*) AS nonsalary FROM E
            WHERE Salary IS NULL),
     WO AS (SELECT count(distinct eid) as atleastone
            FROM W),
SELECT * FROM
    NS, WO;
```

(d) [3] For every employee that exists in **Works**, list his/her *eid*, his/her salary, the deparment this person works for, and the difference between this person's salary and the average salary of all employees in the department this person works for. The result should include one tuple per each department this person works for. The result should have 5 attributes: *eid*, *salary*, *did*, average salary of department (call this attribute *avg*), and the difference between the salary and and the average of the department (call this attribute *diff*).

$A = \gamma_{did,\ avg(salary) \rightarrow avg}\ D.$

$\pi_{eid,\ salary,\ did,\ avg,\ (salary - avg) \rightarrow diff}\ E \bowtie A$

(e) [3] How many departments have less than 3 full time employees? A full time employee is one that works in a department 100% of their time (*pct_time* is 100). Write a query that returns one tuple with one attribute (call it *dcount*).

$F = \gamma_{did,\ count(*) \rightarrow ftcount}\ \sigma_{pct\_time = 100}\ W$

$\gamma_{count(*) \rightarrow dcount}\ \sigma_{ftcount < 3}\ F$

2. Constraints

- [5] Provide a CREATE TABLE for the table **Works**. You **do not not** have to provide a CREATE TABLE for **Dept** nor **Emp**. Your table declaration should satisfy the following constraints (you can ignore those that do only apply to Works). You should **only** use attribute and tuple constraints.

  (a) All primary keys.

  (b) All foreign key constraints.

  (c) Whenever an employee is deleted, all tuples in table **Works** for that employee should be deleted too.

  (d) A department cannot be deleted if it has at least one employee working in it.

  (e) The attribute *pct_time* cannot be null and is an integer between 1 and 100 (it represents the percentage of time a person works for a department).

  (f) The sum of the *pct_time* a person works for all departments should always be 100.

  (g) The manager of a department can only work for one department, it must be the department they manage, and they should work for it 100% of their time.

  Choose appropriate data types for the different attributes of the relations.

```
CREATE TABLE Works(

  PRIMARY KEY(eid, did),
  FOREIGN KEY
CONSTRAIN PCT_time > 0
        and PCT_time <= 100,
CONSTRAINT
    (SELECT sum(pct_time)
     FROM Works W
      where W.eid =eid) = 100
CONSTRAINT NOT EXISTS
    (SELECT * FROM
    Dept where managerid =eid)
OR (
    pct_time = 100 and EXISTS
    (select * From
     Dept where managerid=eid and
      Dept.did = did)
```

is not a manager

Or is a manager of where he works

3. **Security**

(a) [2] What are the minimal privileges required to perform the following operations:

```
UPDATE Dept SET managerid = 1246 WHERE dept = 5;
```

Update Dept (managerid)
Select Dept (dept)

```
SELECT ename, salary FROM Emp JOIN Dept on (managerid = eid);
```

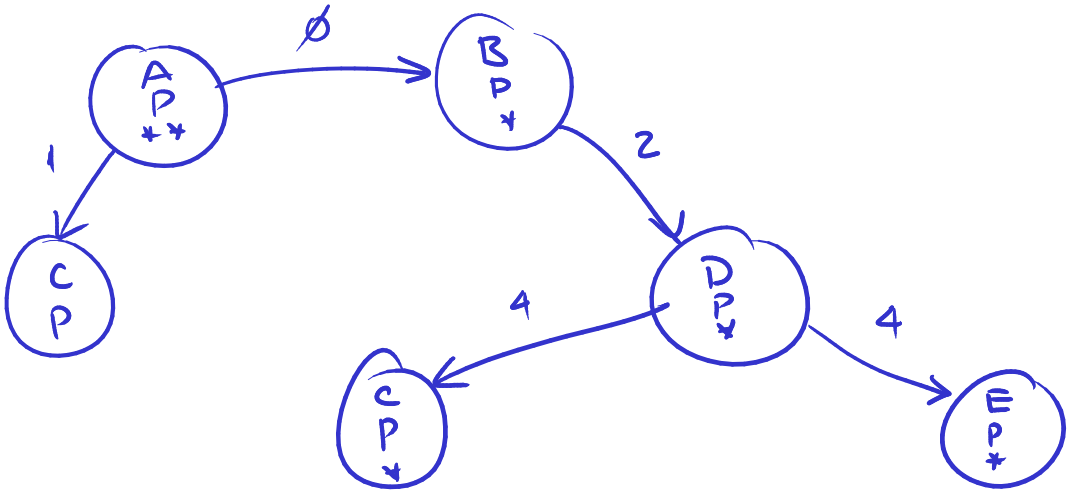
Select emp (ename, salary, eid)
Works eid (salary

(b) [3] Show the grant diagram after steps (4) and (5) of the sequence of actions listed below. Assume A is the owner of the relation to which privilege *p* refers. Column *By* corresponds to the user executing the command.

Step	By	Action
0	A	GRANT *p* TO *B* WITH GRANT OPTION
1	A	GRANT *p* TO *C*
2	B	GRANT *p* TO *D* WITH GRANT OPTION
3	D	GRANT *p* TO *B* WITH GRANT OPTION
4	D	GRANT *p* TO *C*, *E* WITH GRANT OPTION
5	B	REVOKE *p* FROM *D*

fails.

After 4.



After 5.

Same. Revoke fails (default restrict).

4. **Transactions**

We have two transactions, described in terms of their effects on two objects A and B, which we assume are integers.

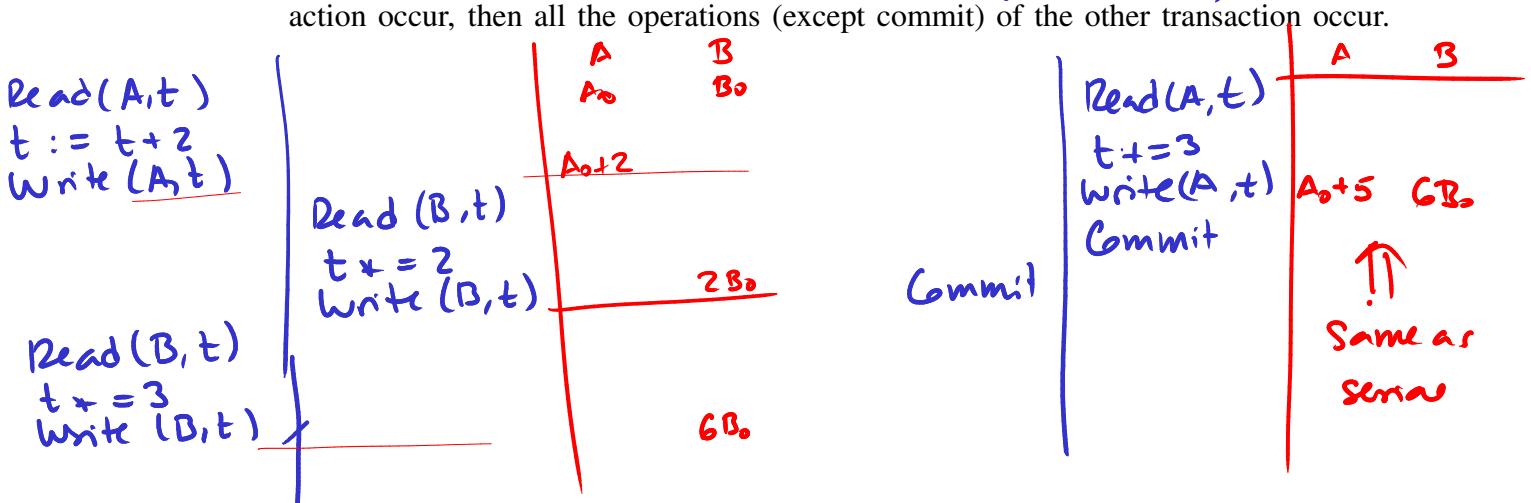
```
T1: Read(A,t); t:=t+2; Write(A,t); Read(B,t); t:=t*3; Write(B,t); commit;
T2: Read(B,t); t:=t*2; Write(B,t); Read(A,t); t:=t+3; Write(A,t); commit;
```

$T_1;T_2 \Rightarrow A = A_0+5,\ B = 6B_0$ | $T_2;T_1 \Rightarrow B = 6B_0,\ A = A_0+5$ $\quad T_1;T_2 = T_2;T_1$

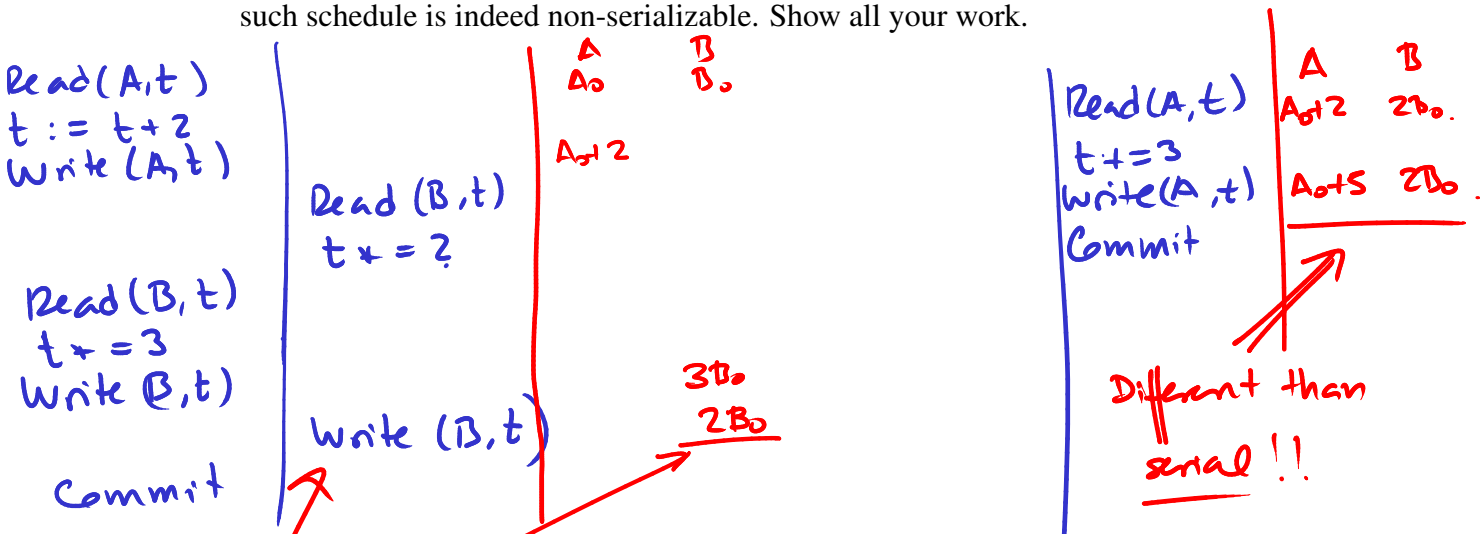
We assume that, whatever consistency constraints there are on the database, these transactions preserve them in isolation.

effect of serial schedules on A and B.

(a) [3] Give an example of a serializable schedule of T1 and T2 that is not serial. Demonstrate that such schedule is indeed serializable. Show all your work. Your answer should not be any of the schedules where all operations (except commit) of one transaction occur, then all the operations (except commit) of the other transaction occur.



(b) [2] Give an example of a non-serializable schedule of T1 and T2. Demonstrate that such schedule is indeed non-serializable. Show all your work.



End of examination. Total pages: 7 Total marks: 30