Database Systems — CSci 4380
Final Exam
May 19, 2016

SOLUTIONS

Question 1 (10 points). You are given the schedule below. List all conflicts. Draw the conflict graph. Discuss whether it is serializable or not. If it is serializable, write down one serial order equivalent to this schedule.

$$r_1(W)\ r_2(X)\ r_2(Y)\ r_1(X)\ w_2(Z)\ r_3(Z)\ w_3(Z)\ w_1(X)\ w_3(X)\ r_3(Y)\ w_3(Y)$$

Solutions: Conflicts: $r_2(X), w_1(X)$
$r_2(X), w_3(X)$
$w_1(X), w_3(X)$
$r_1(X), w_3(X)$
$r_2(Y), w_3(Y)$
$r_2(Y), w_3(Y)$
$w_2(Z), r_3(Z)$
$w_2(Z), w_3(Z)$

Conflict graph: $T_2 \to T_1$, $T_2 \to T_3$,$T_1 \to T_3$.

A serial order is given by T_2, T_1, T_3.

Question 2 (6 points). You are given the following log and data page entries found on disk after a crash. Based on this information, answer the following questions and give an explanation for all your answers.

(a) Can you conclude whether this DBMS uses FORCE or NO FORCE? Why?

(b) Can you conclude whether this DBMS uses STEAL or NO STEAL? Why?

LSN	Log Entry	PrevLSN
1	Update T2 P1	-
2	Update T1 P3	-
3	Update T3 P2	-
4	Update T2 P4	1
5	Update T3 P5	3
6	Commit T2	4
7	Update T1 P4	2
8	Commit T1	7

pageLSN	Data Page
1	P1
3	P2
2	P3
7	P4
5	P5

Solutions:

(a) There is not enough information. All pages by committed transactions are written to disk, so FORCE could be in place but it could also be a coincidence.

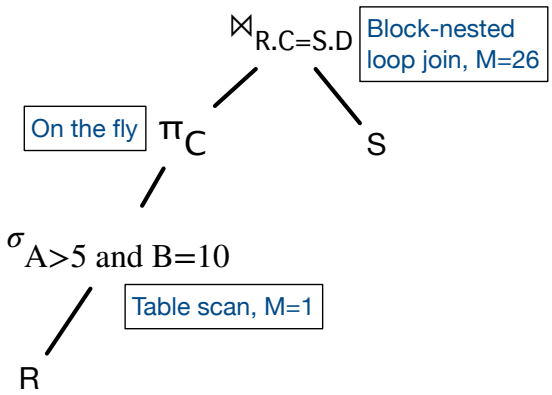
(b) Yes, STEAL is used since update to page P5 is written to disk even though the transaction changing this page is not committed.

Question 3 (18 points). You are given the following query tree and the following statistics. Answer the following and show your work with detailed computations.

TUPLES(R)=60,000	TUPLES(S)=100,000	VALUES(R.A)=3,000	VALUES(R.B)=20
PAGES(R)=8,000	PAGES(S)=5,000	VALUES(R.C)=20,000	VALUES(S.D)=8,000

(a) Find the cost of this operation in terms of number of pages. You can assume that after projection on attribute C, you can store about 20 tuples of R in a single memory block. Show your work.

(b) Propose at least one good index for this query and discuss why this is a good index. An index without explanation will get no points.



Solutions:

(a) Table scan: 8,000 pages

60000/(3*20) = 1,000 tuples pass selection, fits in 1000/20=50 pages

We will read R into 25 pages, so S will be read twice (50/25).

Total cost = 8,000 + 2*5,000 = 18,000

(b) An index on R.A and R.B alone is not selective, but adding C will allow for index only scan. Best index is on R(B,A,C).

Question 4 (18 points). Suppose you are given a relation Proj(EmpNo, DeptNo, ProjNo, Name, Desc) with the following information:

- TUPLES(Proj)=60,000, PAGES(Proj)=12,000
- There are 500 employees (EmpNo), 200 projects (ProjNo) and 100 departments (DeptNo) in this relation. Assume employees are uniformly distributed across departments and projects, and you can use the usual cardinality estimation methods.
- Index I1 is on Proj(ProjNo, DeptNo) with height 2 (root, internal level and leaf) and 250 leaf nodes (with 240 pointers to tuples at each leaf node).

Find the cost of the following queries by using this index. Explain your answers in detail.

```
Q1: SELECT Name, Desc FROM Proj WHERE ProjNo=12 AND DeptNo=5;
Q2: SELECT ProjNo     FROM Proj WHERE DeptNo>3;
Q3: SELECT Name, Desc FROM Proj WHERE ProjNo=5 AND EmpNo=5;
```

Solutions:

```
Q1: Index scan for ProjNo=12 AND DeptNo=5, 60,000/(100*200)=3 tuples
expected in 1 leaf node.

Cost = 3 (index) + 3 (data pages) = 6

Q2: Scan all leaf since index is not sorted by DeptNo. Index only scan
is sufficient to answer this query.

Cost = 2 + 250 = 252

Q3: Scan index for ProjNo=5, expect 60,000/200=300 tuples which fit in
2 leaf nodes

Cost = 4 (index) + 300 (data pages, assuming each tuple is in a
different page in the worst case) = 304
```

For questions 5 and 6, use the data model below representing the leadership positions held by students in the Student Union of a large technological university. Go student leadership!

`Students(studid, fname, lname, email, address)`
This table contains all students (current and past) with their current email and address.

`Offices(oid, name, studtype, duration)`
Available officer positions in the union, `studtype` is the type of student for this office (either `grad` or `undergrad`), and `duration` is the length of the office.

`HeldOffices(hid, studid, oid, fromyear, toyear)`
Stores the offices held by a student. If the student is currently in office, `toyear` is `null`.
A student may only hold one office in any given year.
Each office can be held by at most one student in a calendar year.
These constraints are managed by separate application rules.

Question 5 (18 points). Find students who have held (i.e. in the past) at least two different `undergrad` offices (`Offices.studtype`) and held the same undergrad office at least twice. Return the first name, last name and email of the student, and total duration of time the student served (using `fromyear, toyear` attributes).

Note that a student who served from 2015 to 2015 would have served for 1 year, and 2015 to 2016 for 2 years. We only count past offices held, not the current one if any.

Solutions:

```
SELECT
   s.fname
   , s.lname
   , sum( ho.toyear-ho.fromyear+1 ) as total
FROM
   students s
   , offices o
   , heldoffices h
WHERE
   s.studid = h.studid
   AND o.oid=h.iod
   AND o.studtype='undergrad' and
   h.toyear is not null
GROUP BY
   s.studid
HAVING
   count(distinct o.oid) >= 2
   and count(*) > count(distinct o.oid);
```

```
Students(studid, fname, lname, email, address)
Offices(oid, name, studtype, duration)
HeldOffices(hid, studid, oid, fromyear, toyear)
```

Question 6 (18 points). Find all offices that have been vacant since 2014. These offices currently have no officer and the last officer for this office finished her term in 2014 (term ended on `Heldoffices.toyear`). Return id, name and duration (i.e. `Offices.duration`) for the offices.

Solutions:

```
SELECT
   o.oid
   , o.name
   , o.duration
FROM
   offices o
   , heldoffices h
WHERE
   o.oid = h.oid
   AND ho.toyear = 2014
   AND NOT EXISTS
       (SELECT
            1
        FROM
            heldoffices h2
        WHERE
            h2.oid = o.oid AND
            (h2.endyear IS NULL OR h2.fromyear > 2014)
```

```
SELECT
   o.oid
   , o.name
   , o.duration
FROM
   offices o
   , heldoffices h
WHERE
   o.oid = h.oid
   AND ho.toyear = 2014
EXCEPT
SELECT
   o.oid
   , o.name
   , o.duration
FROM
   offices o
   , heldoffices h
WHERE
   o.oid = h.oid
   AND (h.fromyear > 2014 or h.toyear IS NULL)
```

Question 7 (12 points). Suppose you decide to change the data model from Questions 5 and 6 and store more information for each student. For each student, you will store which years she attended the university and which degrees she obtained. You will also store the various emails and addresses she has used in the past. As a result, the Students relation will change to the following:

```
Students(studid, fname, lname, email, address, from, to, degree, major)
```

with the following functional dependencies:

```
studid → fname lname
studid from to → degree major
```

(a) Put the above relation in BCNF using BCNF decomposition. Show your work by finding the keys of resulting relations and showing that the relation is in BCNF. (We will also accept 3NF decomposition here but check whether the resulting relations are in BCNF or not and mark the keys.)

(b) Using the relations found in part (a) above, check if each relation is in 4NF based on the multi-valued dependencies given below. If it is not, use 4NF decomposition to put them in 4NF.

```
studid ↠ email
studid ↠ address
```

Solutions:

```
BCNF decomposition:
Students(studid, fname, lname, email, address, from, to, degree, major)

Key: studid, from, to, email, address
Both fds violate BCNF

first take out first fd.
S1(studid, fname, lname), key studid, in BCNF
studid->fname, lname

S2(studid, email, address, from, to, degree, major)
studid from to -> degree, major
key: same as student, still not in BCNF

then take out second fd.

S21(studid, from, to, degree, major) key: studid, from, to. In BCNF
S22(studid, email, address, from, to) key: all attributes, in BCNF

Result: S1, S21, S22

(b) S22 is not in 4NF, we must decompose:

S3(studid, email)  key: both attrs
S4(studid, address) key: both attrs
S5(studid, from, to) key: all attrs
```

This page is left blank for scratch work, random thoughts and pictures!
Congratulations, DBS is done.
Let me know if you wish to be a mentor for DBS next semester. It is a great learning experience.
Also, have a great summer.