CS541 Fall 2007 Final, 12 December 2007
Prof. Chris Clifton

WATCH THE TIME! There are several easy questions, make sure you don't get stuck on a hard one and end up without time for the easy ones.

1 Concurrency Control (20 minutes, 11 points)

You are given the following table:

Table Payroll:

Name	*title*	*Salary*
A	Staff	22
B	Staff	56
C	Prof	105
E	Prof	145
S	GS	17
T	GS	18

And the following transactions, in SQL embedded in a programming language. (The items beginning with a :, e.g., :profsal, are host language variables. For example, the first statement saves the result into the variable profsal, which is then used in the last UPDATE statement – from the point of view of SQL, they are treated as constants in the last UPDATE.) The idea of the transactions are to decrease faculty salaries and increase staff and grad salaries, while keeping the total payroll budget the same. (And yes, it works.)

Transaction 1

```
SET TRANSACTION ISOLATION LEVEL
SERIALIZABLE;

SELECT sum(salary)*.1 AS :profsal
FROM payroll
WHERE title = 'Prof';

UPDATE payroll
SET salary = salary * .9
WHERE title = 'Prof';

SELECT count(*) AS :grads
FROM payroll
WHERE title = 'GS';

UPDATE payroll
SET salary = salary + :profsal/:grads
WHERE title = 'GS';

COMMIT;
```

Transaction 2

```
SET TRANSACTION ISOLATION LEVEL
SERIALIZABLE;

SELECT sum(salary)*.05 AS :profsal
FROM payroll
WHERE title = 'Prof';

UPDATE payroll
SET salary = salary * .95
WHERE title = 'Prof';

SELECT sum(salary) AS :staffsal
FROM payroll
WHERE title = 'Staff';

UPDATE payroll
SET salary = salary * (1 + :profsal/:staffsal)
WHERE title = 'Staff';

COMMIT;
```

For the following questions, assume tuple-level operations, and use the *Name* column to identify the tuple to be read/written. Assume no indexes.

1.1 Give a read/write schedule for Transaction 1 (5 minutes, 4 points)

Example (for the first select): $R_1(A), R_1(B), R_1(C), R_1(E), ...$

1 for read in selects, 1 for read all, 1 for read all in updates, 1 for write Prof, 1 each for GS

For the following questions, you are welcome to abbreviate or show ranges - e.g., circle a set of reads/writes in Question 1.1 and label as 'Part1', and just put 'Part1' in place of a bunch of reads/write; or unambiguous ranges (e.g., $R_1(A), ..., R_1(E)$), or "reads/writes from first two statements of T1, reads/writes from first two statements of T2, ..." Note that the reads/writes for Transaction 1 and Transaction 2 are symmetric, so I'll assume that your reads/writes for T2 are basically the same as T1 if not listed explicitly.

1.2 Give a serializable schedule (3 minutes, 2 points)

1 point for operations from both, 1 for serializable

1.3 Give a non-serializable schedule (5 minutes, 3 points)

Also describe why it is non-serializable.

1 point for operations from both, 1 for non-serializable, 1 for why.

1.4 True or False (5 minutes, 2 points)

True or False: "Since the only values used by both Transaction 1 and Transaction 2 are from tuples C and E, all the operations are multiplication, and multiplication is commutative, we still get a valid result (total sum of the salaries being unchanged) with any schedule, even a non-serializable one." Explain/prove your answer.

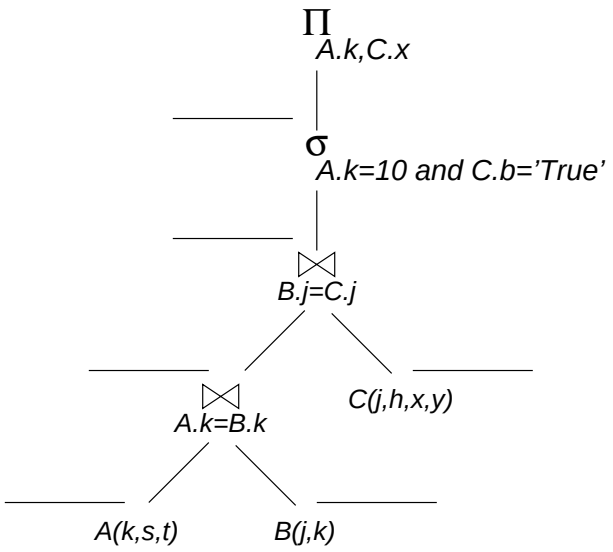
1 for false, 1 for demonstrating with schedule / noting order matters.

1.5 (non)Bonus (1 minute, 0 points)

Would you prefer the equivalent serial ordering be T1 then T2, or T2 then T1?

2 Query cost estimation (20 minutes, 14 points)

You are given the following query tree:



And the following statistics on the relations A, B, and C:

$|A| = 100$ tuples
$A.k$ has 10 distinct values

$|B| = 200$ tuples
$B.j$ has 50 distinct values
$B.k$ has 20 distinct values

$|C| = 50$ tuples
$C.j$ has 50 distinct values
$C.h$ has 20 distinct values
$C.b$ has 2 distinct values

The page size is 1000 bytes, and each tuple in A and C is 100 bytes. Tuples of B are 50 bytes.

2.1 Query Cost Estimation (10 minutes, 9 points)

Give estimates for the intermediate result sizes (in pages) at each step. Do this by filling in the blanks on the diagram. If you show your work somewhere (e.g., on the back of a page), it may help you to get credit if your answer doesn't agree with mine. For partial credit, you can give the result in tuples rather than pages. If you had to make any assumptions (e.g., because I didn't provide sufficient statistics), state them here:

1 each for relations, 2 each for joins, 2 for select, 1 for pages *note: Mistake - should have been C.b in diagram. Accept either answer.*

2.2 Show a More Efficient Tree (5 minutes, 3 points)

Sketch a tree that will give more efficient query processing. There are several things you can do to improve - each of which is worth a point (up to three.) You don't need to show cost estimates. Assume no indexes.

1 point for each improvement (push select, change join, push select, push project

2.3 Is your tree optimal? (5 minutes, 2 points)

Explain why or why not, or explain why you are unable to answer this question ("I don't know how" won't get you any points...)

1 for idea, 1 for good answer.

3 Concurrency Control, Locking, and Recovery (30 minutes, 11 points)

You are given the following read/write schedules for three transactions:

$$R_1(X), R_1(Y), W_1(X), R_1(Z), W_1(X)$$

$$R_2(Y), W_2(Y), R_2(X), W_2(Z)$$

$$R_3(Z), R_3(X), R_3(Y)$$

You will need to show schedules with locking (Shared and eXclusive locks, and Unlocks). Assume two-phase locking. Write a schedule as follows:

$$S_1(X), R_1(X), S_2(Y), R_2(Y), X_2(Y), W_2(Y), R_2(X), X_2(Z), U_2(X), ...$$

3.1 What is wrong with the above example schedule (5 minutes, 2 points)

Describe *why* the above sample schedule is invalid, other than the fact that it isn't finished.

1 for showing knowledge of what "invalid" means, 1 for pointing out error.

3.2 Give a schedule that deadlocks (10 minutes, 3 points)

Obviously you should only show the schedule up to the point where the deadlock occurs. Also show the waits-for graph.

1 for 2PL schedule, 1 for deadlock, 1 for waits-for graph.

3.3 Give a schedule that doesn't deadlock (5 minutes, 2 points)

1 for 2PL schedule, 1 for no deadlock.

3.4 Give non-recoverable schedule (5 minutes, 2 points)

Note that it must be valid two-phase locking and serializable. Give the complete schedule, and show where a crash occurring would cause recovering into an inconsistent state. You might be able to just show where a crash would occur in your answer to Question 3.3.

1 for 2PL schedule, 1 for non-recoverable demonstration.

3.5 Explain why strict two-phase locking prevents a non-recoverable schedule (5 minutes, 2 points)

For one point, describe in terms of your answer to Question 3.4. For full credit, explain why it prevents it in general (e.g., a proof sketch.)

4 Query Processing (35 minutes, 10 points)

Given the following query on the table Payroll from page 1:

SELECT a.name, a.salary, b.name, b.salary
FROM payroll a, payroll b
WHERE a.title = b.title AND a.salary > b.salary;

4.1 Join Algorithm (30 minutes, 8 points)

Sketch how a join algorithm OTHER THAN NESTED-LOOP JOIN would work. For full credit, describe the specific steps taken with the actual table on page 1.

2 points for knowing algorithms, up to 4 for decent sketch or showing knowledge through example, 2 for example.

4.2 Algorithm Choice (5 minutes, 2 points)

Under what conditions (if any) would your algorithm be the best choice? A poor choice?

1 for idea, 1 for good answer.