UNIVERSIDADE FERNANDO PESSOA
Computer Engineering
DATABASE MANAGEMENT SYSTEMS
14/01/2019

Individual exam, only access to your personal notes allowed (1 sheet). 2h. Read carefully before answering. Unreadable answers will be discarded. Good luck. Sign your answer sheets please.

1. Consider the following database schema (the primary keys are underlined):

```
VEHICLE(plate text, maker text, model text, year int, condition text)
OWNER (id serial, name text, address text)
TRANSACTION(plate text, date date, amount real, owner int)
```

   a. List in SQL the name of the owners with more than two transactions of vehicles of maker "V" since the beginning of the year. (1)

select name from owner where id in (select t.owner from transaction t, vehicle v where t.plate=v.plate and v.maker='V' and t.date >= '2019-01-01' group by t.owner having count(*) >= 2)

   b. List in SQL the name of the owners and the maker of their current vehicles. Note: do not list vehicles they owned but already sold (1)

select o.name, v.maker from owner o, transaction t, vehicle v where o.id=t.owner and t.plate=v.plate and t.date = (select max(date) from transaction where plate=t.plate)

   c. If the queries by date in `TRANSACTION` are done frequently justify if an index would be useful and why. (1)

A B+-Tree index could be used in range queries. Probably dates are distributed along the year, so the index will be not unique.

   d. If the queries by date range are done frequently justify if an hashing index is useful (1)

No, hashing indexes do not allow ordering, so a range query could not use efficiently the index.

   e. Suppose the condition of the vehicles is mostly "new", with few "used". If you were asked to create an index which one would you create and why (1)

A partial index for the value "used". Queries using "new" would probably require reading most of the table anyway so an index would not be useful.

   f. Could this query use an index) Why? (1)

```
select * from vehicle where plate like '*22'
```

No because the leading characters are not known so the whole index would have to be visited resulting in probably lots of reads. So the table would be read instead.

2. Consider the following commands in Snapshot Isolation / Read Committed:

```
T1: start transaction isolation level read committed
T2: start transaction isolation level read committed
T1: update course set ects=1 where ects=2;
T1: commit
T2: update course set ects=3 where ects=2;
T2: commit
```

   a. Justify if the schedule is possible. (1)

In RC yes. As T1 commits before T2, T2 uses the new snapshot, re-reading values if needed.

   b. Before the start of the transactions courses had 2 and 4 ECTS. And in the end? (1)

T2 updates after T1 commits, that is, it used the snapshot resulting from T1. After T1 there are only courses with 1 and 4 ECTS. So T2 updates no row, and these are the ECTS at the end of T2

   c. If the isolation level was Repeatable Read, what would happen? (1)

In RR there would be a "cannot serialize error". When T2 tries to commit T1 has already committed changes to the same rows.

3. Check if the following schedules are serializable using the precedence graph:
   a. $r_1(x)$, $r_2(x)$, $r_3(y)$, $w_1(x)$, $r_2(y)$, $w_2(y)$ $r_3(x)$. (1)

2->1, 3->2, 1->3 => cycle, not serializable

   b. $r_2(x)$, $r_3(y)$, $w_1(x)$, $r_2(x)$, $r_2(y)$, $w_1(z)$. (1)

2->1, 1->2 => cycle, not serializable

c. $r_1(x)$, $r_1(y)$, $r_1(v)$, $w_2(x)$, $w_2(y)$, $r_3(x)$, $w_3(v)$, $r_4(z)$, $r_4(v)$, $w_4(y)$, $w_5(y)$, $w_5(z)$. (1)

1->2, 2->3, 1->3, 3->4, 2->4, 2->5, 4->5 => no cycles, serializable

Consider that all transactions finish after their last operation.

4. Execute the three previous schedules with Strict 2PL. Consider there are no waits, if a lock is refused the transaction is rolled back (3).

a) FL1(x) $r_1(x)$, FL2(x) $r_2(x)$, FL3(y) $r_3(y)$, FE1(x) denied rollback T1, FL2(y) $r_2(y)$, FE2(y) denied rollback T2 FL3(x) $r_3(x)$

b) FL2(x) $r_2(x)$, FL3(y) $r_3(y)$, FE1(x) denied T1 rolled back, FL2(x) $r_2(x)$, FL2(y) $r_2(y)$

c) FL1(x) $r_1(x)$, FL1(y) $r_1(y)$, FL1(v) $r_1(v)$, T1 ends, FE2(x) $w_2(x)$, FE2(y) $w_2(y)$, T2 ends, FL3(x) $r_3(x)$, FE3(v) $w_3(v)$, T3 ends, FL4(z) $r_4(z)$, FL4(v) $r_4(v)$, FE4(y) $w_4(y)$, T4 ends, FE5(y) $w_5(y)$, FE5(z) $w_5(z)$, T5 ends. Note this schedule is serial so there are no concurrency issues.

5. Execute the three previous schedules with READ UNCOMMITTED (no read locks) and READ COMMITTED (read locks are released after reads). (3)

Read uncommitted

a) $r_1(x)$, $r_2(x)$, $r_3(y)$, FE1(x) $w_1(x)$, $r_2(y)$, FE2(y) $w_2(y)$ $r_3(x)$

b) $r_2(x)$, $r_3(y)$, FE1(x) $w_1(x)$, $r_2(x)$, $r_2(y)$, FE1(z) $w_1(z)$

c) $r_1(x)$, $r_1(y)$, $r_1(v)$, T1 ends, FE2(x) $w_2(x)$, FE2(y) $w_2(y)$, T2 ends, $r_3(x)$, FE3(v) $w_3(v)$, $r_4(z)$, $r_4(v)$, FE4(y) $w_4(y)$, T4 ends, FE5(y) $w_5(y)$, FE5(z) $w_5(z)$, T5 ends

Read committed

a) FL1(x) $r_1(x)$ -FL1(x), FL2(x) $r_2(x)$ -FL2(x), FL3(y) $r_3(y)$ -FL3(y), FE1(x) $w_1(x)$, FL2(y) $r_2(y)$ -FL2(y), FE2(y) $w_2(y)$ FL3(x) $r_3(x)$ -FL3(x)

b) FL2(x) $r_2(x)$ -FL2(x), FL3(y) $r_3(y)$ -FL3(y), FE1(x) $w_1(x)$, FL2(x) denied T2 rolled back, FE1(z) $w_1(z)$

d) FL1(x) $r_1(x)$ -FL1(x), FL1(y) $r_1(y)$ -FL1(y), FL1(v) $r_1(v)$ -FL1(v), T1 ends, FE2(x) $w_2(x)$, FE2(y) $w_2(y)$, T2 ends, FL3(x)) $r_3(x)$ –FL3(x), FE3(v) $w_3(v)$, T3 ends, FL4(z) $r_4(z)$ -FL4(z), FL4(v) $r_4(v)$ -FL4(v), FE4(y) $w_4(y)$, T4 ends, FE5(y) $w_5(y)$, FE5(z) $w_5(z)$, T5 ends

6. Explain the execution of the following schedule X=3, Y=2, $r_1(x)$,$w_1(x, x+5)$, $r_2(x)$, $r_2(y)$, $w_2(y, y+x)$, $w_2(x, x * 3)$, showing values of X and Y for each transaction after each operation:
   a. Using Snapshot Isolation in Read Committed (1).

X0=3, Y0=2, $r_1(x)$, X1=3, $w_1(x, x+5)$, X1=8, $r_2(x)$, X2=3, $r_2(y)$, Y2=2, $w_2(y, y+x)$ Y2=5, $w_2(x, x * 3)$ X2=9

For those considering that T1 commits before T2:

X0=3, Y0=2, $r_1(x)$, X1=3, $w_1(x, x+5)$, X1=8, C1, $r_2(x)$, X2=8, $r_2(y)$, Y2=2, $w_2(y, y+x)$ Y2=10, $w_2(x, x * 3)$ X2=24

   b. Using Strict 2PL (1).

X=3, Y=2, FL1(x) $r_1(x)$, X=3, FE1(x) $w_1(x, x+5)$, X=8, FL2(x) denied T2 rolled back

For those considering that T1 commits before T2:

X=3, Y=2, FL1(x) $r_1(x)$, X=3, FE1(x) $w_1(x, x+5)$, X=8, C1, FL2(x) $r_2(x)$, X2=8, FL2(y) $r_2(y)$, Y2=2, FE2(y) $w_2(y, y+x)$ Y2=10, FE2(x) $w_2(x, x * 3)$ X2=24