Omtenta i DATABASER

Svar:

Obs! Lärarversion, med lösningar

DATE: August 30, 2002	**TIME:** 8.45 – 12.45	**PLACE:** M-building

Questions:	Patrik Jansson, ext. 5415
Points:	maximum 60 p.
Grades:	CTH: 3 from 24 p., 4 from 36 p., 5 from 48 p.
	GU: G from 28 p., VG from 48 p.
	Ph.D. students: G from 28 p.
Result:	will be announced 18 Sept. 2002

Note:

- Use your time wisely — the questions are not ordered by difficulty.
- Write clearly and avoid writing too much on each page.
- Start every new exercise on a new sheet.
- All answers should be **well motivated** and not unnecessarily complicated!
- Your answers may be in Swedish or English.

Good luck!

Question 1. 12p For each equivalence, either show that it holds (by reasoning on the meaning of the relational algebra operations in terms of the sets of tuples) or give a counter example (with concrete, small relations).

a) $\pi_A(R - S) = \pi_A(R) - \pi_A(S)$

Svar: If R and S have only the column A then the equality holds, as the projection in that case does nothing. The next simplest case is if they have two columns, A and B. The simplest counter example is if only the B column differs, for example: $R = \{(1,1)\}$ and $S = \{(1,2)\}$. Then $LHS = \pi_A(R - S) = \pi_A(R) = \{(1)\}$ but $RHS = \pi_A(R) - \pi_A(S) = \{(1)\} - \{(1)\} = \{\}$.

b) $R \bowtie_\theta (S - T) = (R \bowtie_\theta S) - (R \bowtie_\theta T)$

Svar:

Let $t = (t_1, t_2)$ where t_1 is the part from R and t_2 is the part from S.

$t \in ((R \bowtie_\theta S) - (R \bowtie_\theta T))$
$= (t \in R \bowtie_\theta S) \ \wedge \ t \notin R \bowtie_\theta T$
$= (t \in \sigma_\theta(R \times S)) \ \wedge \ t \notin \sigma_\theta(R \times T)$
$= (\theta(t) \ \wedge \ t \in R \times S) \ \wedge \ \neg(\theta(t) \ \wedge \ t \in R \times T)$
$= (\theta(t) \ \wedge \ t_1 \in R \ \wedge \ t_2 \in S) \ \wedge \ \neg(\theta(t) \ \wedge \ t_1 \in R \ \wedge \ t_2 \in T)$
$= \theta(t) \ \wedge \ t_1 \in R \ \wedge \ t_2 \in S \ \wedge \ (\neg\theta(t) \ \vee \ t_1 \notin R \ \vee \ t_2 \notin T)$
$= \theta(t) \ \wedge \ t_1 \in R \ \wedge \ t_2 \in S \ \wedge \ t_2 \notin T$
$= \theta(t) \ \wedge \ t_1 \in R \ \wedge \ t_2 \in (S - T)$
$= \theta(t) \ \wedge \ t \in R \times (S - T)$
$= t \in \sigma_\theta(R \times (S - T))$
$= t \in R \bowtie_\theta (S - T)$

Question 2. 12p Consider the following actions taken by a transaction T1 on database objects X and Y:

R(X), W(X), R(Y), W(Y)

a) Give an example of another transaction T2 that, if run concurrently to the transaction T1 without some form of concurrency control, could interfere with T1. Motivate your answer by giving a schedule S of T1 and T2, and a serializability graph of S such that S is not conflict-serializable.

b) Explain how the use of strict two phase locking would prevent interference between the two transactions. If the transaction manager receives requests for the operations in the order of your schedule S, what would be the actual sequence of events (all locks, reads, writes and unlocks)?

c) Strict two phase locking is used in many database systems. Give two reasons for its popularity. Compare with plain two phase locking and conservative two phase locking.

Svar: a) T2: W(X)

S = 1RX 2WX 1WX 1RY 1WY

Conflicting pair: same object, different transactions, at least one is W

Here we have (1RX,2WX) and (2WX, 1WX).

Graph: T1 ¡-¿ T2, has a loop, thus not conflict-serializable.

b) With strict 2PL, the lock on X would not be released until the end of transaction T1, thus preventing T2 to change X.

S' = 1LX 1RX 1WX 1LY 1RY 1WY 1UY 1UX 2LX 2WX 2UX

c) 2PL guarantees serializability (and thus non-interference) while still leaving room for parallel execution. Strict 2PL also avoids cascading rollbacks. Conservative 2PL is needed to completely avoid deadlock, but this requires that all locks are known in advance.

All kinds of 2PL are easy to implement (transactions only “communicate” through locks, each transaction knows locally if it conforms or not).

Question 3. 12p Consider a relation scheme $R = (ISCDAO)$, satisfying the functional dependencies $S \to D$, $I \to A$, $IS \to C$, $A \to O$.

a) Find and list all candidate keys. Justify your answer with a complete derivation of the candidate key dependencies.

b) Consider the decomposition $(ISCD, IAO)$. Show that it is not in BCNF.

c) Refine the decomposition of b) with additional splits of the schemes, so that the final result is in BCNF.

d) Is your decomposition from c) dependency preserving? Is it loss-less? Motivate your answers.

Svar: The only candidate key is IS. The decomposition given in b) is not fully in BCNF (due to $S \to D$, for instance). A refinement would be (SD, IA, AO, ISC). This would be dependency preserving.

```
  ISCDAO

S->D
I->A
IS->C
A->O

a)

LHS = IS
both= A
RHS = DCO

IS has to be in the candidate key and
IS is enough (IS->ISC->ISCDA->ISCDAO, that is IS+ = R)
=> IS is the only candidate key

b)

in ISCD there is S->D where S is not a candidate key => not in BCNF

in IAO there is A->O where A is not a candidate key => not in BCNF

c)

ISCD split using S->D gives ISC, SD

IAO split using A->O gives IA, AO
```

```
In total we get the tables ISC, SD, IA, AO, each with one of the
original dependencies. These are all in BCNF.

d)

Each of the original dependencies are preserved so, yes, it is dependency
preserving.

The first split could have resulted from the (inferred) dependency I->AO.
And because the splitting algorithm always produces a loss-less join
decomposition then, yes, it is loss-less.
```

Question 4. 8p Explain, briefly, the rules for translating the following ER-diagram concepts into relational tables: strong entity set, weak entity set, M-to-N relation, multi-valued attribute.

Svar: See the book.

Question 5. 16p A database is needed to store information about doctors, patients, and tests for a hospital. Obviously, a doctor takes care of many patients, and also a patient can have several doctors (since doctors have different specialisations). A test is performed on a single, particular patient, by one or more doctors in the hospital. Besides identification data, for each patient dates of admission and checking out are kept. For each test, besides date and time when it was administered, some description of the kind of test (for instance, blood sample) and its result are kept.

a) Construct a complete E-R diagram for the database.

b) Define all the tables corresponding to your diagram in SQL. You must capture all cardinality, key and participation constraints in the SQL definitions if it is possible. If you cannot capture some constraint in the SQL definitions, you must explain why it is not possible to do so.

c) Give an SQL expression returning all patients who haven't received any tests.

d) Define a view in SQL for all the triples consisting of patient name, name of a doctor administering the test, and test name.