TDDB48 Database techniques, May 31, 2006

Sketched answers to the written examination, provided by Lena Strömbäck and Juha Takkinen, IDA.

**Question 1.**

[“cows” include male and female, as well as “children” cows …]

**Question 2.**

[Note! The entity “Treatment” (Swe. “Behandling”) is missing an attribute. It should have “Type” or “ID” added.]

Kund(Telefon, Namn)

[Eng. Customer(Teleph, Name)]

Frisör(Namn)

[Eng. Barber(Name)]

Behandling(Namn, Type)

[Eng. Treatment(Name, Type)]

Bokning(Frisör, Nr, Datum, Tid, Kund)

[Eng. Booking(Barber, Nr, Date, Time, Customer)]
- Kund (Customer) is foreign key and refers to table Kund (Customer), as is Frisör (Barber), which refers to table Frisör (För(Frisör, Nr, Behandling))

[Eng. To(Barber, Nr, Treatment)]
- Frisör (Barber) and Nr are foreign key, which refers to Bokning (Booking)
- Behandling (Treatment) is foreign key and refers to the table Behandling (Treatment).

**Question 3.**

1. Select maträtt
   From rätter
   Where kategori = "Fågel"
       and recept = "Vår kokbok"

2. Select distinct maträtt
   From rätter, vänner, bjudningar
   Where familj = "Karlssons"
       and vänner.id = bjudningar.vän
       and bjudningar.maträtt = rätter.id

3. [We do not give minus points for syntax errors]

Create or replace procedure NAMN (dat in date, rätt in INT, fam in varchar2) AS

   CURSOR c IS SELECT ID FROM VÄNNER
       WHERE familj = fam;

   BEGIN
   FOR v IN c LOOP
     INSERT INTO BJUDNINGAR(datum, vän, rätt)
       Values(dat, v, rätt);
   END LOOP

   END;

**Question 4.**

[A company exists in only one town]

a) Definition of a primary key = a candidate key with minimum attributes that uniquely identifies a row in the table. [common errors: trying to define PK as “uniquely identifying” something, but forgetting to mention candidate keys and the role of attributes]

The candidate attributes are EmployeeID, ContractNo, and CompanyID. We select EmployeeID and ContractNo as the primary key because these two are the minimum attributes needed to uniquely identify each row in the table. (They also better reflect the
objective of the company, which is to keep track of staff that have a contract with another company.)

b) EmployeeName is *fully functionally dependent* on EmployeeID because EmployeeID --> EmployeeName, but because EmployeeID is *part of the PK*, EmployeeName is not fully functionally dependent on the PK, and therefore the table is not in 2NF.

In order to transform to 2NF, we need to *find functional dependencies on the PK*, including partial ones:

EmployeeID, ContractNo --> Hours
EmployeeID --> EmployeeName
ContractNo --> CompanyID, CompanyTown

Remove partial dependencies on the PK and place in new table:

StaffContract(EmployeeID, ContractNo, Hours)
StaffDetails(EmployeeID, EmployeeName)
ContractDetails(ContractNo, CompanyID, CompanyTown)

These tables are now in 2NF. In order to transform them into 3NF, we need to look at transitive dependencies among non-primary key attributes. The location of a company (CompanyTown) is functionally dependent on CompanyID, which is dependent on the PK for ContractDetails above. Divide the latter table into two tables:

Contracts(ContractNo, CompanyID)
Company(CompanyID, CompanyTown)

The final tables normalised into 3NF are:

StaffContract(EmployeeID, ContractNo, Hours)
StaffDetails(EmployeeID, EmployeeName)
Contracts(ContractNo, CompanyID)
Company(CompanyID, CompanyTown)

The tables are also in BCNF because there are no functional dependencies where the determinant is not a superkey.

[Common errors: not showing or motivating the steps when transforming from one NF to another; not showing the final tables; not using the two required terms (“full functional dependency” and “partially dependent”) to describe why the original tables are not in 2NF]

**Question 5.**

[Use the tables in question 2 and common sense about the purpose of the tables in real life. Also, in d) and e) the condition should be (in the Swedish version) “rätter.id=bjudningar.maträtt”. In g) and h), a table can have several attributes but no multi-valued attributes.]

F= false, T = true

a) F

b) T

c) F

d) T
Question 6.
ACID:
A = atomicity, a transaction is an atomic unit of processing, that is, it is performed wither in its entirety or not at all. This is implemented by the recovery system.

C = consistency, a transaction has a consistent state before an after the execution of the transaction. This is usually implemented by the programmer, with the aid of the database manager system, for example checking that a primary key with a single attribute is indeed unique.

I = isolation, a transaction appears as though it is being executed in isolation from others. This is implemented by the concurrency control system of the database system.

D = durability, a change applied by a committed transaction is permanent and not lost due to a failure. This is handled by the recovery system.

Question 7.
a) The blocking factor (non-spanning) = floor(block size / record size) = floor(4,096 / 500) = floor(8.192) = 8 records / block [-0.5 points for not giving the unit for the blocking factor]

b) A dense primary index has an index record for every data record --> 4,096 / 8 = 512 index records per disk block, which with a total of 500,000 index records means 500,000 / 512 = 977 index blocks (rounded up, no. of blocks with index records).

c) The primary index has problems when doing updates and inserts because of potential needs to rearrange records onto different blocks, which takes time. Another approach would be to use a multilevel index, where we can reduce the search time by reducing the portion of the index that we need to search in by a blocking factor bfr (fan-out).

Question 8.
We assume the notation write-item Tx, y, z means that transaction x sets the value of y to the new value z (the previous value of y is not shown explicitly in this notation).

a) Immediate update means that all updates are saved to disk immediately. Assume we are doing variant 2 (the general version) of the algorithm, that is, we do not require that all updates have been written to disk before commit.

Committed transactions will have their updates correctly performed, but transactions that did not commit before the failure need to be undone (UNDO/REDO).

Scan the log backwards from the end. The committed transactions T8 and T6 have their updates recorded on disk, so they will not have to be undone (do nothing). T5 has no commit, so undo its update to variable E (restored to unknown value, not given in the example). T4 has no commit either so its update to B is undone, restoring it to 100. T3
has been committed so do nothing. Then, a checkpoint is encountered, which means that all updates up until the checkpoint have been recorded in the database.

b) \( A = 50, B = 100, C = 25, D = 10, E = \text{unknown}, \) and \( F = 10. \)

c) The application may have done a division by zero, a bank account may have insufficient funds, the resource is not available due to a network error, or the user may have pressed the abort button. For example.

**Question 9.**

a) The canonical tree:

\[
\pi_{\text{namn}, \text{epost}}
\]

\[
\sigma_{\text{institution} = \text{plats AND studentID} = \text{personNr AND kurskod} = \text{kursID AND namn} = \text{‘Lena’}}
\]

b) Heuristic optimisation is the transformation of the query represented in the tree into an equivalent query that is easier or faster to execute.

c) For example the rule (step) that the more restrictive select should be done first, that is, be situated closest to the leaf where the table belonging to the selection is.