CHALMERS UNIVERSITY OF TECHNOLOGY
Department of Computer Science and Engineering

Examination in Databases, TDA357/DIT620

Thursday 8 April 2010, 08:30-12:30

Examiner:	Graham Kemp (telephone 772 5411, room 6475 EDIT) The examiner will visit the exam room at 09:30 and 11:30.
Results:	Will be published by 27 April 2010 at the latest.
Exam review:	see course web page for time and place http://www.cse.chalmers.se/edu/year/2009/course/TDA357/HT2009/
Grades:	Grades for Chalmers students (TDA357) are normally determined as follows: ≥ 48 for grade 5; ≥ 36 for grade 4; ≥ 24 for grade 3. Grades for GU students (DIT620) are normally determined as follows: ≥ 42 for grade VG; ≥ 24 for grade G.
Help material:	One A4 sheet with hand-written notes. You may write on both sides of that sheet. That sheet must be handed in with your answers to the exam questions. English language dictionaries are allowed.

Specific instructions:

- Please answer in English where possible. You may clarify your answers in Swedish if you are not confident you have expressed yourself correctly in English.
- Begin the answer to each question on a new page.
- Write clearly; unreadable = wrong!
- Fewer points are given for unnecessarily complicated solutions.
- Indicate clearly if you make any assumptions that are not given in the question.
- Write the page number and question number on every page.

Question 1. Consider the following domain description.
12 p

> A university's Examinations Office wants to use a relational database management system to store data about examinations and examination rooms. For each examination room (e.g. room "VV11"), the Examinations Office wants to store information about the seat positions within that room. Within an examination room a seat is identified by its row number and column number (e.g. all exam rooms will have a sear with row number 1 and column number 1). For each examination, the date and start time must be recorded, and also the course code. Since some courses can have many students, an examination will typically be held at the same time in several examination rooms (e.g. the "TDA357" examination on a particular day might be held in "VV11" and "VV12"). Since data covering many years will be stored in the database, the course code alone is insufficient to uniquely identify an examination. The Examinations Office wants to record which students are registered for which examinations. Student names will be stored, but students will be identified using unique student identifiers. When a student registers for a particular examination, they will be assigned an anonymous code for that examination; the same student will be assigned different anonymous codes for different examinations. For each examination, the Examinations Office needs to record where each student was sitting (e.g. it should be possible to find out which student was sitting in row 1, column 1, in room "VV11" for the exam in "TDA357" on a particular day).

a) Draw an E-R diagram that correctly models this domain. (6p)

b) Translate this E-R diagram into a set of relations, clearly marking all references and keys. (6p)

Question 2. Suppose we have relation $R(A, B, C, D, E, F)$ with functional dependencies $AB \rightarrow D$, $BD \rightarrow E$, $CE \rightarrow F$, $F \rightarrow A$, $C \rightarrow B$.
11 p

a) State, with reasons, whether each of the following is a *key* of R.

i) AB
ii) AC
iii) BDF
iv) CE
v) CDF

(5p)

b) Which attributes of R are prime?
(1p)

c) State, with reasons, which FDs listed above violate BCNF.
(1p)

d) Decompose relation R to BCNF. Show each step in the normalisation process, and at each step indicate which functional dependency is being used. Indicate keys and references for the resulting relations.
(4p)

Question 3. A database used by a university to record students' attendance at lectures has the following relations:
9 p

Students(studentId, name)
Lectures(course, week, day, hour, room)
AttendsLecture(student, course, week, day, hour)

Student identifiers (*studentId*) are unique, but student names might not be unique. Attribute *course* is a course code. Attribute *week* is a week number e.g. week number 7), *day* is a day of the week (e.g. "Thursday"), and *hour* is a lecture's start time. No lecture can have a start time before 8 or after 17.

a) Suggest keys and references for these relations.
Write SQL statements that create these relations with constraints in a DBMS.
(4p)

b) No course can have more than 3 lectures in the same week. Write an assertion that checks this.
(2p)

c) No lectures may be scheduled on Tuesday of week 7.
Write a trigger that checks whether a new lecture being added to the Lectures relation is on Tuesday of week 7 and, if it is, reschedules the lecture for 17 on the following day (Wednesday of week 7). You may assume that the same lecture room will be available at that time.
(3p)

Question 4. Assume the same relations as in Question 3:
5 p

Students(studentId, name)
Lectures(course, week, day, hour, room)
AttendsLecture(student, course, week, day, hour)

a) Write a relational algebra expression that finds the names of all students who attended a lecture in room HC4 in week 6.
(2p)

b) Write a relational algebra expression that finds how many lectures in course TDA357 were attended by each student. The result should contain the student's name, and the number of lectures that they attended in this course, and the results should be sorted by the number of lectures attended.
(3p)

Question 5. Assume the same relations as in Question 3:

8 p

Students(studentId, name)
Lectures(course, week, day, hour, room)
AttendsLecture(student, course, week, day, hour)

a) Write an SQL query that finds the rooms in which the student with studentId "s012345" attended a lecture in week 6. The result should not contain duplicate rows.
(2p)

b) Create a view $V(studentId, name, course, numAttended)$ which contains the number of lectures (the fourth column in the view) that the student with $studentId$ and $name$ attended in $course$.
(3p)

c) Write an SQL query that finds all courses that have more lectures than course "TDA357".
(3p)

Question 6. Suppose that an airline seat booking system contains the following transaction:

4 p

T:

$Step\ T_1$: *list available seats*
$Step\ T_2$: *book seat selected by user*
$Step\ T_3$: $confirm$ *booking*

Both steps T_2 and T_3 involve waiting for input from the user.
If the user does not confirm the booking in step T_3, the transaction will be rolled back.

Suppose that users A and B both try to book the same seat using transaction T.

a) Give one disadvantage of running transaction T with isolation level READ COMMITTED.
(1p)

b) Give one disadvantage of running transaction T with isolation level SERIALIZABLE.
(1p)

c) Explain how the behaviour of the system could be different if transaction T is run with isolation level READ UNCOMMITTED instead of READ COMMITTED.
(2p)

Question 7. Suppose we have relation $Accounts(accNo, custNo, balance)$, and that this relation is stored in 20 disc blocks.
4 p

Each customer can have more than one account and, on average, each customer has accounts stored in three disc blocks.

Suppose that two kinds of task are performed on this relation:

- task 1: inserting a new row;
- task 2: finding account information for a given customer.

a) For each of these tasks, state how many disc block transfers will be needed if:

   i) there are no indexes
   (1p)

   ii) there is an index on $custNo$ (assume that this index fits into a single disc block).
   (1p)

b) Suppose that 80% of the operations performed on this relation are inserting new rows (task 1), and that the remaining 20% are finding account information for given customers (task 2).

   In this case, is it better to have an index on $custNo$, or is it better to have no indexes on this relation?
   Show the calculations that support your answer.
   (2p)

Question 8. Consider the following piece of XML which has a "relational" structure.
7 p

```
<?xml version="1.0" standalone="yes" ?>

<Question8>
 <Employees>
  <Employee name="Andersson" branch="b3" salary="35000" />
  <Employee name="Jonsson"   branch="b3" salary="25000" />
  <Employee name="Larsson"   branch="b2" salary="32000" />
 </Employees>
 <Branches>
  <Branch number="b1" city="Stockholm" />
  <Branch number="b2" city="Paris" />
  <Branch number="b3" city="London" />
 </Branches>
</Question8>
```

a) Write XPath expressions that finds all employees at branch "b2".
(1p)

b) The flexibility of XML enables us to nest elements in a more natural way than in the example shown at the top of this question. Write a piece of XML that contains the same information as in the example shown above, but which uses more natural tags and nesting, and avoids duplication of branch numbers.
(2p)

c) Write a Document Type Definition (DTD) for the XML in your answer to part (b).
(2p)

d) Based on the XML in your answer to part (b), write an XQuery expression that gives the employee name and city of employees whose salary is over 30000, i.e.

```
<Result>Larsson: Paris</Result>
<Result>Andersson: London</Result>
```

(2p)