CHALMERS UNIVERSITY OF TECHNOLOGY
Department of Computer Science and Engineering

Examination in Databases, TDA357/DIT620

Friday 19 December 2008, 14:00-18:00

Examiner:	Graham Kemp (telephone 772 5411, room 6475 EDIT) The examiner will visit the exam room at 15:00 and 17:00.
Results:	Will be published by the middle of January at the latest.
Exam review:	see course web page for time and place http://www.cs.chalmers.se/Cs/Grundutb/Kurser/dbas/DbasHT2008/index.cgi
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.

If you have not yet completed the course evaluation form, please do this after the exam.

Specific instructions:

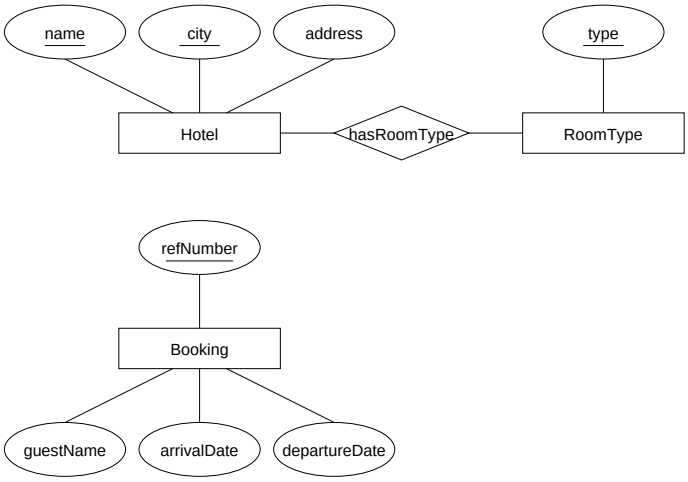
- Please answer in English where possible. You may clarify your answers in Swedish if you are not confident you have expressed yourself correctly in English.
- Begin the answer to each question on a new page.
- Write clearly; unreadable = wrong!
- Fewer points are given for unnecessarily complicated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

10 p

Each hotel in a hotel chain is identified by the hotel's name and the city where the hotel is located. The address of each hotel is also stored. Each hotel has several types of room (e.g. single, double, twin). The information stored about each booking includes a unique booking reference number, the name of the guest who made the booking, the arrival date and departure date. *(This part of the domain description is shown in the incomplete E-R diagram, below.)*

Within each hotel the room price is determined by the room type (i.e. all single rooms in the same hotel have the same price, but single rooms in different hotels can have different prices). The number of rooms of each type in each hotel should be stored. Rooms within each hotel are identified by a room number. The type of each room must be stored in the database, so that the price of each booking can be calculated by database application programs. A single booking can include many rooms (e.g. a guest may book several rooms for a group of friends as part of the same booking).



a) Complete the E-R diagram so that it correctly models this domain (include the information from the second paragraph of the domain description). (5p)

b) Translate this E-R diagram into a set of relations, clearly marking all references and keys. (5p)

Question 2. 10 p

a) Suppose we have relation $R(A, B, C, D)$ and functional dependencies $B \rightarrow C$, $D \rightarrow B$.

i) By considering the closures of all subsets of attributes, find **all** non-trivial FDs, superkeys and keys. State which FDs violate BCNF. (4p)

ii) Decompose relation R to BCNF in two different ways:

— by first decomposing on $B \rightarrow C$

— by first decomposing on $D \rightarrow B$.

Show each step in the normalisation process, and at each step indicate which functional dependency is being used. (3p)

b) Consider relations $Takes(student, courseCode)$ and $Classes(room, day, hour, courseCode)$. Let R be the relation $Takes \bowtie Classes$.

State all FDs and MVDs that you would expect to hold for relation R. (3p)

Question 3. 4 p

The regional health service uses a relational database to record information about the doctors who work at the health centres in the region, and appointments that patients have with doctors. The database has the following relations:

$Doctors(\underline{person\_number}, health\_centre)$

$Appointments(\underline{appointment\_id}, patient, doctor, day, month, year)$
$doctor \rightarrow Doctors.person\_number$

Each tuple in relation *Doctors* contains a doctor's personal identification number and the name of the health centre where that doctor works. Each tuple in relation *Appointments* contains a unique appointment identifier, the personal identification number of the patient, the personal identification number of the doctor, the day of the month (1-31), the month (1-12) and the year.

a) Write a relational algebra expression that computes then names of the health centres, sorted in alphabetical order, where the patient with identification number 6006064444 had appointments in year 2000 or more recently. (2p)

b) Write a relational algebra expression that computes, for each health centre, the total number of appointments at that health centre in each month of 2007. (2p)

Question 4. 8 p

Consider the following relations that contain information about students and their exam points:

$Students(id, name)$

$Points(code, month, year, student, points)$
$student \rightarrow Students.id$

Assume that the FD $code, student \rightarrow month, year, points$ holds for relation $Points$.

a) Write an SQL query that prints the names of all students in alphabetical order, and the points that they scored in course TDA357. Students who do not have a grade for this course should be listed with a null value in the output instead of the number of points. (2p)

b) Write two SQL queries that find the name(s) of the student(s) who scored the highest number of points in the December 2007 exam in course TDA357.

   i) One query must use MAX. (2p)

   ii) The other must not use MAX. (2p)

c) Create a view $V(course, month, year, avgPoints)$ which contains the course month, year and average points scored in exams that were taken by over 100 students. (2p)

Question 5. 10 p

An airline stores information about its flights and ticket sales in a relational database with the following relations:

Flight(flightNumber, day, month, year, numSeats, price)
Passengers(passengerId, name, address)
Booking(bookingReference, flightNumber, day, month, year, passenger)

The number of seats for a flight must be between 50 and 200.

When details of a new flight are added to the database, a default price of 2000 SEK is used unless another price is given.

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

   ii) Motivate the update and delete policies that you choose for foreign key in your answer to part (i).
(2p)

b) The number of bookings for a particular flight may not exceed the number of seats on that flight. Write an assertion to check this constraint.
(2p)

c) When there are fewer than 20 seats remaining on a flight on a particular day then the airline's policy is to set the price of remaining seats on that flight to 4000 SEK. Write a trigger to implement this policy. (You may assume that bookings cannot be cancelled.)
(2p)

Question 6. 4 p

Suppose we have relation $Accounts(accNo, custNo, balance)$, and that this relation contains the following tuples:

accNo	custNo	balance
A001	C001	10000
A002	C001	20000

Suppose that customer C001 transfers 1000 from account A001 to A002. Suppose also that a bank official runs a query to find the total amount in customer C001's accounts. These transactions (let's call them T1 and T2) can be summarised by the following steps:

T1:

$Step\ T1_A:\ get\ balance\ of\ A001\ into\ X$
$Step\ T1_B:\ set\ balance\ of\ A001\ to\ X - 1000$
$Step\ T1_C:\ get\ balance\ of\ A002\ into\ Y$
$Step\ T1_D:\ set\ balance\ of\ A002\ to\ Y + 1000$

T2:

$Step\ T2_A:\ get\ balance\ of\ A001\ into\ V$
$Step\ T2_B:\ get\ balance\ of\ A002\ into\ W$
$Step\ T2_C:\ print\ V + W$

a) What are the possible outcomes if neither T1 nor T2 is run as a serializable transaction? Explain your answer. (2p)

b) What are the possible outcomes if T1 is run as a serializable transaction, and T2 is run with isolation level `READ UNCOMMITTED`? Explain your answer. (2p)

Question 7. 6 p

Suppose we have relation $GivenCourses(course, period, teacher)$, and that this relation is stored in 10 disc blocks.

Each course can run more than once and, on average, each course has records stored in two disc blocks. Similarly, on average, each teacher has records stored in three disc blocks.

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

- task 1: inserting a new row;
- task 2: finding the period and teacher for a given course code;
- task 3: finding the period and course code for courses taught by a given teacher.

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 *course* (assume that this index fits into a single disc block) (1p)
   ii) there is an index on $(teacher, period)$ (assume that this index fits into a single disc block) (1p)
   iv) both of the indexes mentioned in (ii) and (iii) are used. (1p)

b) Suppose that 60% of the operations performed on this relation are inserting new rows (task 1), that 20% of the operations are finding information for a given course (task 2) and that the remaining 20% are finding information for a given teacher (task 3).
   In this case, is it better to have no indexes, an index on only *course*, an index on only $(teacher, period)$, or both of those indexes?
   Show the calculations that support your answer. (2p)

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

```
<?xml version="1.0" standalone="no">

<LabReports>
 <GroupMembers>
  <GroupMember group="20" student="101" />
  <GroupMember group="20" student="102" />
  <GroupMember group="21" student="103" />
  <GroupMember group="22" student="104" />
 </GroupMembers>
 <LabsPassed>
  <LabPassed lab="1" group="20" />
  <LabPassed lab="1" group="21" />
  <LabPassed lab="2" group="20" />
  <LabPassed lab="2" group="22" />
 </LabsPassed>
</LabReports>
```

a) Write an XQuery expression that finds the student numbers of all students who have passed lab 2, i.e.

```
<Result>101</Result>
<Result>102</Result>
<Result>104</Result>
```

(2p)

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 group 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 XPath expression that finds elements with information about the groups that have passed lab 2. (2p)

CHALMERS UNIVERSITY OF TECHNOLOGY
Department of Computer Science and Engineering

Examination in Databases, TDA357/DIT620

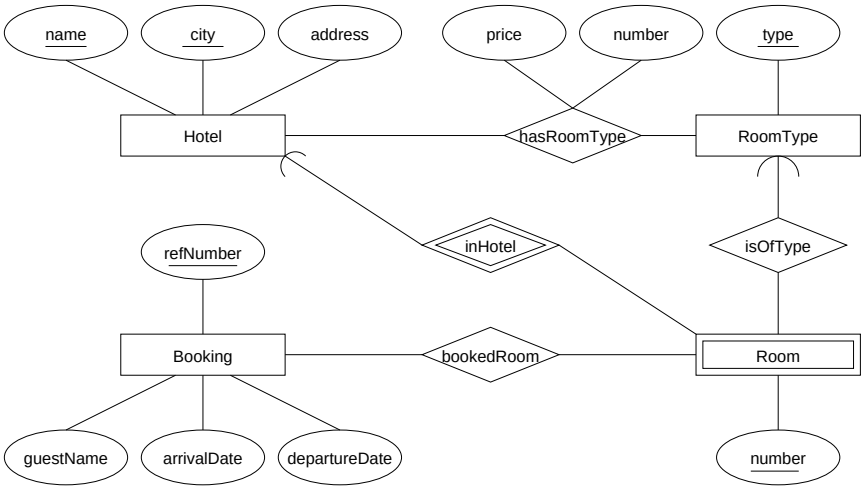
Friday 19 December 2008, 14:00-18:00

Solutions

Updated 2011-12-09

Question 1. 10 p

a) E-R diagram:



b) $Hotels(\underline{name}, \underline{city}, address)$

$RoomTypes(\underline{type})$

$HasRoomType(\underline{name}, \underline{city}, \underline{roomType}, price, number)$

$(name, city) \rightarrow Hotels.(name, city)$

$roomType \rightarrow HasRoomType.type$

$Rooms(\underline{name}, \underline{city}, \underline{number}, roomType)$

$(name, city) \rightarrow Hotels.(name, city)$

$roomType \rightarrow HasRoomType.type$

$Bookings(\underline{refNumber}, guestName, arrivalDate, departureDate)$

$BookedRooms(\underline{name}, \underline{city}, \underline{number}, \underline{refNumber})$

$(name, city, number) \rightarrow Rooms.(name, city, number)$

$refNumber \rightarrow Bookings.refNumber$

Question 2. 10 p

a) i) After considering the closures of all subsets of attributes, we find the following additional non-trivial FDs:

$D \rightarrow C$
$AB \rightarrow C$
$AD \rightarrow B$
$AD \rightarrow C$
$BD \rightarrow C$
$CD \rightarrow B$
$ABD \rightarrow C$
$ACD \rightarrow B$

Superkeys are: AD, ABD, ACD, ABCD. There is one key: AD.
FDs that violate BCNF:

$B \rightarrow C$
$D \rightarrow B$
$D \rightarrow C$
$AB \rightarrow C$
$BD \rightarrow C$
$CD \rightarrow B$

ii) — By first decomposing on $B \rightarrow C$, we first get $R_1(B, C)$ and $R_2(A, B, D)$. R_2 is not in BCNF, so we must decompose further.
— By first decomposing on $D \rightarrow B$, we first get $R_1(B, C, D)$ and $R_2(A, D)$. R_1 is not in BCNF, so we must decompose further.
In both cases, we end up with three relations, $R_a(A, D)$, $R_b(B, C)$ and $R_c(B, D)$.

b) FD:

$$room, day, hour \rightarrow courseCode$$

MVDs:

$$courseCode \twoheadrightarrow room, day, hour$$
$$courseCode \twoheadrightarrow student$$

Question 3. 4 p

a) $\tau_{health\_centre}(\pi_{health\_centre}($
$Doctors \bowtie_{person\_number=doctor} (\sigma_{patient=\text{“}6006064444\text{”} AND year \geq 2000}(Appointments))))$

b) $\gamma_{health\_centre,month,count(*)\rightarrow numApps}($
$Doctors \bowtie_{person\_number=doctor} (\sigma_{year=2007}(Appointments)))$

Question 4. 8 p

a)
```
SELECT   name, points
FROM     Student NATURAL LEFT OUTER JOIN (
             SELECT student AS id, points
             FROM   Points
             WHERE code = 'TDA357' )
ORDER BY name;
```

b) i)
```
WITH DBStudents AS (
        SELECT id, name, points
        FROM   Students JOIN Points ON id = student
        WHERE  code = 'TDA357' AND year = 2007 AND month = 'December' )
   SELECT name
   FROM   DBStudents
   WHERE  points = ( SELECT MAX(points) FROM DBStudents );
```

ii)
```
WITH DBStudents AS (
        SELECT id, name, points
        FROM   Students JOIN Points ON id = student
        WHERE  code = 'TDA357' AND year = 2007 AND month = 'December' )
   SELECT name
   FROM   DBStudents
   WHERE  points >= ALL ( SELECT points FROM DBStudents );
```

c)
```
CREATE VIEW V AS
  SELECT   course, month, year, AVG(points) AS avgPoints
  FROM     Points
  GROUP BY course, month, year
  HAVING   COUNT(student) > 100;
```

Question 5. a) i) *Flight(flightNumber, day, month, year, numSeats, price)*
10 p *Passengers(passengerId, name, address)*
Booking(bookingReference, flightNumber, day, month, year, passenger)
$flightNumber \rightarrow Flight.flightNumber$
$(flightNumber, day, month, year) \rightarrow Flight.(flightNumber, day, month, year)$
Solutions that make different assumptions about the keys and foreign keys might be accepted.

```
CREATE TABLE Flight (
    flightNumber     VARCHAR(8),
    day              INT,
    month            INT,
    year             INT,
    numSeats         INT CHECK (numSeats BETWEEN 50 and 200),
    price            INT DEFAULT 2000,
    PRIMARY KEY (flightNumber, day, month, year)
);

CREATE TABLE Passengers (
    passengerId      VARCHAR(20) PRIMARY KEY,
    name             VARCHAR(30),
    address          VARCHAR(50)
);

CREATE TABLE Booking (
    bookingReference VARCHAR(20) PRIMARY KEY,
    flightNumber     VARCHAR(8),
    day              INT,
    month            INT,
    year             INT,
    passenger        VARCHAR(20),
    FOREIGN KEY (flightNumber, day, month, year)
        REFERENCES Flight(flightNumber, day, month, year)
        ON DELETE SET NULL
        ON UPDATE CASCADE,
    FOREIGN KEY passenger REFERENCES Passengers.passengerId
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
```

ii) Here are some suggestions, but some other policies will be accepted if these are well motivated.
If a passenger's ID changes, then we want to change the passenger ID also in that passenger's bookings. If a passenger is deleted from the database, then we might want to delete all of that passenger's bookings.
Regarding the references between bookings and flights, what we want to happen on update will probably depend on which part of the flight's key changes. For example, if a new flight number is assigned to the flight, then it would seem reasonable to cascade that change to all bookings for the flight. However, if the day changes, then we might want to SET NULL, and inform the passenger that they should contact the airline. If the flight is deleted, we could simply delete all bookings for that flight. However, the airline might prefer to SET NULL in the Booking relation, until passengers can be informed and possibly be assigned to other flights.

b)
```
CREATE ASSERTION NotOverbooked CHECK
  ( NOT EXISTS (
        SELECT flightNumber
        FROM   Flight F
        WHERE  numSeats < (
                    SELECT COUNT(bookingReference)
                    FROM   Booking B
                    WHERE  B.flightNumber = F.flightNumber
                       AND B.day = F.day
                       AND B.month = F.month
                       AND B.year = F.Year )
  ) );
```

c)
```
CREATE TRIGGER SetPriceOfRemainingSeats
AFTER INSERT ON Booking
REFERENCING NEW ROW AS new
FOR EACH ROW
WHEN
  ( ( ( SELECT COUNT( bookingReference)
        FROM   Booking
        WHERE  flightNumber = new.flightNumber
           AND day = new.day
           AND month = new.month
           AND year = new.year ) -
      ( SELECT numSeats
        FROM   Flight
        WHERE  flightNumber = new.flightNumber
           AND day = new.day
           AND month = new.month
           AND year = new.year ) ) > 20 )
BEGIN
    UPDATE Flight
    SET    price = 4000
    WHERE  flightNumber = new.flightNumber
       AND day = new.day
       AND month = new.month
       AND year = new.year;
END;
```

Question 6. 4 p

a) Only T1 updates the database, and accounts A001 and A002 will end up with balances of 9000 and 21000, respectively, regardless of the order in which the steps are executed. However, the amount printed by transaction T2 will depend on the order in which steps $T1_B$, $T1_D$, $T2_A$ and $T2_B$ are carried out.

Orderings [$T1_B$, $T1_D$, $T2_A$, $T2_B$], [$T2_A$, $T2_B$, $T1_B$, $T1_D$], [$T1_B$, $T2_A$, $T1_D$, $T2_B$] and [$T2_A$, $T1_B$, $T2_B$, $T1_D$] all give the correct amount of 30000.

However, [$T2_A$, $T1_B$, $T1_D$, $T2_B$] gives 31000 and [$T1_B$, $T2_A$, $T2_B$, $T1_D$] gives 29000.

b) Possible outcomes are that T2 will print 30000 (orderings [$T1_B$, $T1_D$, $T2_A$, $T2_B$] and [$T2_A$, $T2_B$, $T1_B$, $T1_D$]) or 31000 (ordering [$T2_A$, $T1_B$, $T1_D$, $T2_B$]).

Question 7. 6 p

a) i) task 1: 2, task 2: 10, task 3: 10.
ii) task 1: 4, task 2: 3, task 3: 10.
iii) task 1: 4, task 2: 10, task 3: 4.
iv) task 1: 6, task 2: 3, task 3: 4.

b) (i) and (iii) have cost of 5.2 and (ii) and (iv) have cost 5.0. There's not much difference between the four options, but according to the calculations either only an index on *course*, or both indexes, would be best.

Question 8. 8 p

a)
```
for $g in /LabReports/GroupMembers/GroupMember,
    $l in /LabReports/LabsPassed/LabPassed[@lab="2"]
where $l/[@group] = $g/[@group]
return <Result>{$g/student}</Result>
```

b) Here is one suggestion. Other solutions might be accepted.

```
<LabReports>
 <Group groupnumber="20">
  <Student>101</Student>
  <Student>102</Student>
  <LabPassed>1</LabPassed>
  <LabPassed>2</LabPassed>
 </Group>
 <Group groupnumber="21">
  <Student>103</LabPassed>
  <LabPassed>1</LabPassed>
 </Group>
 <Group groupnumber="22">
  <Student>104</LabPassed>
  <LabPassed>2</LabPassed>
 </Group>
</LabReports>
```

c)
```
<!ELEMENT LabReports (Group*)>
<!ELEMENT Group (Student+, LabPassed*)>
<!ELEMENT Student (#PCDATA)>
<!ELEMENT LabPassed (#PCDATA)>
```

d)
```
/LabReports/Group[LabPassed=2]
```