CHALMERS UNIVERSITY OF TECHNOLOGY
Department of Computer Science and Engineering

Examination in Databases, TDA357/DIT620

Wednesday 14 December 2011, 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 the middle of January at the latest.
Exam review:	See course web page for time and place: http://www.cse.chalmers.se/edu/year/2011/course/TDA357/HT2011/
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.

12 p

A cinema wants to use a database to manage information about the films that it shows, and customer bookings. The cinema has several rooms where films are shown, and each room is identified by a unique room number. Within each room, a seat is identified by its row number and the seat number within the row. Films are identified by their titles. Films can be classified according to genre (e.g. 'action' or 'thriller'), and one film can belong to more than one genre (e.g. 'action' and 'thriller'). For publicising films, a film's duration and language need to be recorded in the database. Each film is typically shown on many occasions, so the date and starting time of each showing need to be represented in the database. Each film can be shown several times on the same day. It's possible that the same film could be shown on the same day and at the same starting time in two or more of the rooms of the cinema. Seat bookings also need to be represented in the system. A booking, identified by a unique booking reference number, can include several seat reservations for the same showing of a film (same date, starting time and room), and for each booking it must be possible to identify which seats have been reserved. The customer's name must be stored with the booking.

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 that information about people, the books that they own, and the cars that they drive are stored in relation $R(p, n, i, t, a, r, man, mod)$, where the attributes are as follows:

10 p

p — person identification number
n — a person's name
i — ISBN number identifying a book
t — a book's title
a — a book's author
r — a car's registration number
man — a car's manufacturer
mod — a car's model

This relation has {p,i,r} as its key.
The following functional dependencies hold for this relation:

$p \rightarrow n$
$i \rightarrow t$
$i \rightarrow a$
$r \rightarrow man$
$r \rightarrow mod$

a) 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)

b) We would expect one of the relations in your solution to part (a) of this question to violate Fourth Normal Form (4NF).

i) State which relation, and give the multivalued dependencies (also referred to as MVDs, or independencies, or INDs) that we would expect to hold for that relation.

ii) Decompose this relation to Fourth Normal Form.

(3p)

(If you didn't answer part (a), then you can answer part (b) for the original relation R.)

c) Suppose we have relation $R(A, B, C, D)$, with FDs $AB \rightarrow C$, $C \rightarrow A$ and $C \rightarrow D$.

i) Does FD $AB \rightarrow AD$ hold for R?
Explain your answer.

ii) It does not necessarily follow that FD $B \rightarrow D$ holds for R.
Give some example data for relation R such that $AB \rightarrow C$, $C \rightarrow A$ and $C \rightarrow D$ hold, but $B \rightarrow D$ does not hold.

(3p)

Question 3. 9 p

A database system used by a company to record the percentage of employees' working time that is assigned to different departments has the following relations:

Departments(deptName, location)
Employees(empId, name)
WorksIn(employee, dept, location, percentage)

Employee identifiers (*empId*) are unique.
Attribute *deptName* describes the department's function (e.g. 'accounting', 'sales', or 'marketing') and attribute *location* is the name of the city where that department is located. Note that there can be several departments at the same location, and different locations can have departments with the same name (despite having the same name (e.g. 'sales'), these are considered to be different departments).
Attribute *percentage* is the percentage of an employee's time that is assigned to a particular department, and must be an integer value between 0 and 100, inclusive. If the percentage of time is not stated when an employee is assigned to a department, then the database management system should use 0% as a default value.

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

b) It should not be possible to assign more that 100% of an employee's time in total. Write an assertion that checks this.
(2p)

c) If an attempt is made to make a new assignment of an employee to a department (i.e. adding a new row to relation *WorksIn*), and the assigned time would cause this employee to have more that 100% of their working time assigned, then the assignment of time should be changed so that only that employee's remaining available time is assigned (e.g. if 90% of an employee's time is already assigned, and an attempt is made to assign a further 30% of that employee's to another department, then only 10% should in fact be assigned).
Write a trigger that implements this.
(3p)

Question 4. 6 p

Assume the same relations as in Question 3:

Departments(deptName, location)
Employees(empId, name)
WorksIn(employee, dept, location, percentage)

a) Write a relational algebra expression that finds the employee name and department name of employees who are assigned to work more than 50% in a department in Stockholm.
(2p)

b) Write a relational algebra expression that finds the employee names, department names and locations of the employees working in departments to which more than three employees are assigned.
(4p)

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

10 p

Departments(deptName, location)
Employees(empId, name)
WorksIn(employee, dept, location, percentage)

a) Write an SQL query that finds the names of employees who work in the sales department in Stockholm, and at least one other department. Each employee in the result should be listed only once, and the results should be sorted by name.
(3p)

b) Write an SQL query that finds departments where nobody is assigned to work more than 50% of their working time (note: it is possible that some departments will have no employees, and these should also be listed as part of the result).
(3p)

c) Write an SQL query that finds the department name and location of the department(s) with the largest amount of allocated working time (calculate this as the sum of allocated percentages for the department).
(4p)

Question 6. Suppose we have relation *Prices(<u>store</u>,<u>item</u>,price)* and the following PSM procedure:

4 p

```
CREATE PROCEDURE AdjustPrice(IN i VARCHAR(20), IN a INTEGER)
DECLARE x INTEGER;
BEGIN
  -- Step 1
    SELECT price INTO x FROM Prices WHERE item = i;
  -- Step 2
    UPDATE Prices SET price = x + a WHERE item = i;
END;
```

a) Suppose relation *Prices* contains the tuple (s001,i001,100).
Suppose user A issues the command:

```
CALL AdjustPrice('i001', 10)
```

and user B issues the command:

```
CALL AdjustPrice('i001', 20)
```

at the same time, and that the procedures are run as transactions with isolation level READ COMMITTED.
Give all possible interleavings of the steps (refer to the steps of A's transaction as A_1 and A_2, and the steps of B's transaction as B_1 and B_2), and state the corresponding outcomes.
(2p)

b) Explain whether the following indexes will improve the performance of procedure AdjustPrice:

i) CREATE INDEX isIndex ON Prices(item, store)
ii) CREATE INDEX piIndex ON Prices(price, item)

(2p)

Question 7. Consider the following piece of XML:

9 p

```
<Cookbook>
  <Recipe name="boiled egg">
    <Ingredient name="eggs" quantity="1" />
    <Step number="1">Put egg in cold water</Step>
    <Step number="2">Boil the egg</Step>
  </Recipe>
  <Recipe name="pancakes">
    <Ingredient name="eggs" quantity="1" />
    <Ingredient name="flour" quantity="0.75" unit="cup" />
    <Ingredient name="milk" quantity="0.25" unit="pint" />
    <Ingredient name="sugar" quantity="1" unit="tablespoon" />
    <Ingredient name="oil" quantity="1" unit="teaspoon" />
    <Step number="1">Heat oil in a pan</Step>
    <Step number="2">Mix ingredients</Step>
    <Step number="3">Pour some of the mixture into the pan</Step>
    <Step number="4">Turn pancake over after a few minutes</Step>
  </Recipe>
  <Recipe name="toast">
    <Ingredient name="bread" quantity="1" unit="slice" />
    <Step number="1">Put bread in toaster</Step>
    <Step number="2">Switch on toaster</Step>
  </Recipe>
</Cookbook>
```

a) Write a Document Type definition (DTD) for the XML that is given above.

(3p)

b) Write an XPath expression that finds the step 1 elements of each recipe, i.e.

```
<Step number="1">Put egg in cold water</Step>
<Step number="1">Heat oil in a pan</Step>
<Step number="1">Put bread in toaster</Step>
```

(1p)

c) Write an XQuery expression that finds the names of all recipes that contain eggs. The result of this query should look as follows:

```
<Result>
  <EggRecipe name="boiled egg"/>
  <EggRecipe name="pancakes"/>
</Result>
```

(2p)

d) Write an XQuery expression that finds the recipe elements of the recipe(s) with the most steps (your solution should find a sequence of recipe elements if there are several recipes sharing the same greatest number of steps). The result of this query should look as follows:

```
<Result>
  <Recipe name="pancakes">
    <Ingredient name="eggs" quantity="1"/>
    <Ingredient name="flour" quantity="0.75" unit="cup"/>
    <Ingredient name="milk" quantity="0.25" unit="pint"/>
    <Ingredient name="sugar" quantity="1" unit="tablespoon"/>
    <Ingredient name="oil" quantity="1" unit="teaspoon"/>
    <Step number="1">Heat oil in a pan</Step>
    <Step number="2">Mix ingredients</Step>
    <Step number="3">Pour some of the mixture into the pan</Step>
    <Step number="4">Turn pancake over after a few minutes</Step>
  </Recipe>
</Result>
```

(3p)

CHALMERS UNIVERSITY OF TECHNOLOGY
Department of Computer Science and Engineering

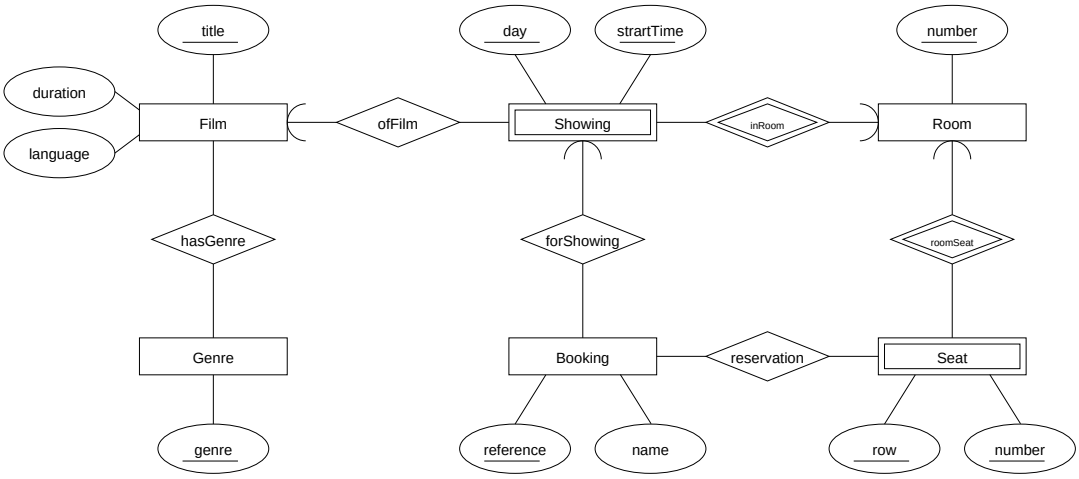
Examination in Databases, TDA357/DIT620

Wednesday 14 December 2011, 08:30-12:30

Solutions

Updated 2011-12-15

Question 1. a) E-R diagram:
12 p



b) $Films(\underline{title}, language, duration)$

$Genres(\underline{genre})$

$HasGenre(\underline{film}, \underline{genre})$
$film \rightarrow Films.title$
$genre \rightarrow Genres.genre$

$Rooms(\underline{number})$

$Seats(\underline{room}, \underline{row}, \underline{number})$
$room \rightarrow Rooms.number$

$Showings(\underline{day}, \underline{startTime}, \underline{room}, film)$
$room \rightarrow Rooms.number$
$film \rightarrow Films.title$

$Bookings(\underline{reference}, name, day, startTime, room)$
$(day, startTime, room) \rightarrow Showings.(day, startTime, room)$

$Reservations(\underline{booking}, \underline{room}, \underline{row}, \underline{number})$
$booking \rightarrow Bookings.reference$
$(room, row, number) \rightarrow Seats.(room, row, number)$

Question 2. 10 p

a)
```
Decompose on p -> n
{p}+ = {p,n}

        R1(_p,n)
        R2(p,i,t,a,r,man,mod)
            p -> R1.p

Decompose R1 on i -> t
{i}+ = {i,t,a}

        R21(_i,t,a)
        R22(p,i,r,man,mod)
            i -> R21.i

Decompose R22 on r -> man
{r}+ = {r,man,mod}

        R221(_r,man,mod)
        R222(p,i,r)
            r -> R221.r

The key of R222 is (p,i,r).

Should update references to decomposed relations.
```

b)
```
Relation R222 has MVDs p ->> i and p ->> r

Decompose R222 on p ->> i

        R2221(_p,_i)
        R2222(_p,_r)

( The original relation R has MVDs p,n ->> i,t,a and p,n ->> r,man,mod )
```

c)
```
i) Yes.
   AB -> AD can be rewritten as 2 FDs: AB->A and AB->D
   The first of those is trivial.
   The second is true due to transitivity: AB->C and C->D, so AB->D
```

ii)

A	B	C	D
a1	b1	c1	d1
a2	b1	c2	d2

Question 3. 9 p

a) $Departments(\underline{deptName}, \underline{location})$
$Employees(\underline{empId}, name)$
$WorksIn(\underline{employee}, \underline{dept}, \underline{location}, percentage)$
$employee \rightarrow Employees.empId$
$(dept, location) \rightarrow Departments.(deptName, location)$

```
CREATE TABLE Departments (
    deptName   VARCHAR(20),
    location     VARCHAR(20),
    PRIMARY KEY (deptName, location)
);

CREATE TABLE Employees (
    empId      CHAR(10) PRIMARY KEY,
    name       VARCHAR(30)
);

CREATE TABLE WorksIn (
    employee   CHAR(10),
    dept       VARCHAR(20),
    location     VARCHAR(20),
    percentage INT DEFAULT 0 CHECK (percentage >= 0 AND percentage <= 100),
    PRIMARY KEY (employee, dept, location),
    FOREIGN KEY (employee) REFERENCES Employees(empId)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    FOREIGN KEY (dept, location) REFERENCES Departments(deptName,location)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
```

b)
```
CREATE ASSERTION NotOverFullTime CHECK
  ( NOT EXISTS
      (
        SELECT   employee
        FROM     WorksIn
        GROUP BY employee
        HAVING   SUM(percentage) > 100
      )
  );
```

c)
```
CREATE TRIGGER MaxOneHundred
BEFORE INSERT ON WorksIn
REFERENCING NEW AS new
FOR EACH ROW
DECLARE previousPercentage INT;
BEGIN
    SELECT  SUM(percentage) INTO previousPercentage
    FROM    WorksIn
    WHERE   employee = :new.employee;

    IF previousPercentage + :new.percentage > 100 THEN
        :new.percentage := 100 - previousPercentage;
    END IF;
END;
```

Question 4. 6 p

a) $\pi_{empId,deptName}(Employees \bowtie_{empId=employee} (\sigma_{percentage>50 \wedge location='Stockholm'}(WorksIn)))$

b) $\pi_{name,deptName,location}(Employees \bowtie_{empId=employee} (WorksIn \bowtie \sigma_{num>3}(\gamma_{dept,location,COUNT(*)\rightarrow num}(WorksIn))))$

Question 5. 10 p

a)
```
SELECT   DISTINCT name
FROM     Employees JOIN WorksIn w1 on empId = w1.employee
                   JOIN WorksIn w2 on empId = w2.employee
WHERE    w1.dept = 'sales'
         AND w1.location = 'Stockholm'
         AND (w2.dept <> 'sales' OR w2.location <> 'Stockholm')
ORDER BY name
```

b)
```
SELECT  *
FROM    Departments
WHERE   (deptName, location) NOT IN
            ( SELECT dept, location
              FROM   WorksIn
              WHERE percentage > 50 )
```

c)
```
SELECT   dept, location
FROM     WorksIn
GROUP BY dept, location
HAVING   SUM(percentage) >= ALL
             ( SELECT   SUM(percentage)
               FROM     WorksIn
               GROUP BY dept, location )
```

Question 6. 4 p

a) $A_1\ A_2\ B_1\ B_2$ gives 130 for price of item 'i001'
$A_1\ B_1\ A_2\ B_2$ gives 120 for price of item 'i001'
$A_1\ B_1\ B_2\ A_2$ gives 110 for price of item 'i001'
$B_1\ A_1\ A_2\ B_2$ gives 120 for price of item 'i001'
$B_1\ A_1\ B_2\ A_2$ gives 110 for price of item 'i001'
$B_1\ B_2\ A_1\ A_2$ gives 130 for price of item 'i001'

b) The index in (i) will improve the performance, but the index in (ii) will not. See section 8.3.2 of the textbook for an explanation.

Question 7. 9 p

a)
```
<!DOCTYPE Cookbook [

<!ELEMENT Cookbook (Recipe*) >

<!ELEMENT Recipe (Ingredient*, Step*) >
  <!ATTLIST Recipe
    name CDATA #REQUIRED >

<!ELEMENT Ingredient EMPTY >
  <!ATTLIST Ingredient
    name CDATA #REQUIRED
    quantity CDATA #REQUIRED
    unit CDATA #IMPLIED >

<!ELEMENT Step (#PCDATA) >
  <!ATTLIST Step
    number CDATA #REQUIRED >

]>
```

b)
```
//Step[@number="1"]
```

c)
```
<Result>
  {
    for $r in doc("cookbook.xml")//Recipe[Ingredient/@name="eggs"]
    return <EggRecipe name="{$r/@name}" />
  }
</Result>
```

d)
```
<Result>
  {
    let $d := doc("cookbook.xml")
    let $max := max( for $r in $d//Recipe
                     let $numsteps := count($r/Step)
                     return $numsteps
                   )
    for $r in $d//Recipe
    where count($r/Step) = $max
    return $r
  }
</Result>
```