CHALMERS UNIVERSITY OF TECHNOLOGY
Department of Computer Science and Engineering

Examination in Databases, TDA357/DIT620

Saturday 18 December 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 the middle of January at the latest.
Exam review:	See course web page for time and place: http://www.cse.chalmers.se/edu/year/2010/course/TDA357/HT2010/
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 university wants to use a database to store information about its departments, divisions and employees. Each department at the university has a unique name. Each department contains several divisions. Divisions in different departments can have the same name, but the division names within each department are unique. There can be many employees in each division, but each employee is employed at only one division. For each employee, their name and their unique personNumber should be stored. There are two kinds of employee at the university: faculty members and PhD students. For each PhD student, one faculty member is appointed to be their examiner. Each PhD student also has one main supervisor, but they can have zero or more co-supervisors. One faculty member at each department is appointed to be the head of that department. Similarly, one faculty member at each division is appointed to be the head of that division.

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
12 p $AB \to C$, $CD \to E$, $E \to B$, $B \to F$, $E \to D$, $E \to F$.

a) State whether each of the following is a *key* and/or a *superkey* of R, or neither.
   i) AB
   ii) ABD
   iii) ABE
   iv) ACD
   v) AE
   vi) BCDEF

   (3p)

b) The keys identified in part (a) are the only keys of relation R.
   i) State, with reasons, which FDs listed above violate BCNF.
   ii) 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.

   (5p)

c) i) State which FDs listed above violate third normal form (3NF).
   ii) Decompose relation R to 3NF.

   (4p)

Question 3. A database system used by a university's examinations office has the following relations:

10 p

Exams(course, examDate, examTime)
Students(studentId, name)
registeredFor(student, course, examDate)

Student identifiers (*studentId*) are unique.
Attribute *course* is a six-character course code.
A course cannot have more than one exam on the same date, and that exam will either be in the morning (`'AM'`) or in the afternoon (`'PM'`).
Attribute *examDate* is a date (e.g. `'2010-12-18'`) and *examTime* is either `'AM'` or `'PM'`.

a) Suggest keys and references for these relations.
Write SQL statements that create these relations with constraints in a DBMS.
(5p)

b) No student is allowed to register for more than two exams on the same day.
Write an assertion that checks this.
(1p)

c) If a clash occurs for a student, special arrangements will be made for the student to sit the exam at another time on the same date. For example, if the clash is with another `'AM'` exam, then a special exam will be arranged for that student in the afternoon (`'PM'`), and vice versa.
Write a trigger that, when a student registers for an exam that is at the same time on the same date as another exam for which they are already registered, adds a row to relation *SpecialExams(student, course, examDate, examTime)*. Here, *examTime* should be the time of the specially arranged exam (`'AM'` or `'PM'`).
A row should be added to the *registeredFor* relation even when the student will sit a special exam.
(4p)

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

6 p

Exams(course, examDate, examTime)
Students(studentId, name)
registeredFor(student, course, examDate)

a) Write a relational algebra expression that finds the names of students who have registered for the exam in course `'TDA357'` on `'2010-12-18'`.
(2p)

b) Write a relational algebra expression that finds the average number of students who have registered for the exams in each course (for example, if there have been three exams in course `'TDA357'` and 100 students registered for the exam on the first occasion, 150 students registered for the second occasion and 80 students registered for the third occasion, then the average number of students registering for an exam in course `'TDA357'` would be 110).
The result should contain the course code and the average number of students registered for exams in that course, and the results should be sorted by course code.
(4p)

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

8 p

Exams(course, examDate, examTime)
Students(studentId, name)
registeredFor(student, course, examDate)

a) Write an SQL query that finds the names of all students who are registered for exams in courses TDA357 and TIN092 on the same day. The results should be sorted by name.
(2p)

b) Write an SQL query that finds dates where the only exam scheduled is the exam for course TDA357.
(2p)

c) Create a view *V(course, avgSt)* which contains the average number of students who have registered for the exams in each course (for example, if there have been three exams in course `'TDA357'` and 100 students registered for the exam on the first occasion, 150 students students registered for the second occasion and 80 students registered for the third occasion, then the average number of students registering for an exam in course `'TDA357'` would be 110).
(4p)

Question 6. 4 p

a) We can tell the database management system that the next transaction is read-only by executing the following command:

```
SET TRANSACTION READ ONLY;
```

Why is it useful to execute this statement before a read-only transaction begins? (1p)

b) Suppose you are the owner of table `T`.

i) Describe what user123 will be able to do after you execute the following command:

```
GRANT REFERENCES ON T TO user123;
```

ii) Explain why executing the statement in part (i) will limit what you are able to do with table `T`.

(3p)

Question 7. 3 p

Suppose relation $R(A, B, C)$ is as follows:

A	**B**	**C**
a1	d	80
a1	k	20
a1	r	30
a2	m	60
a4	g	90
a5	d	60
a6	m	40
a7	c	80
a8	k	60
a8	s	30

Draw a picture that shows a *secondary index* on column B of relation R.

(3p)

Question 8. Consider the following piece of XML:

5 p

```
<Exams>
 <Exam course="TDA357" room="ML14" date="2010-04-08">
  <Student studentId="s003">
   <Points>30</Points>
  </Student>
  <Student studentId="s004">
   <Points>10</Points>
  </Student>
 </Exam>
 <Exam course="TIN092" room="ML14" date="2010-12-17">
  <Student studentId="s001">
   <Points>50</Points>
  </Student>
  <Student studentId="s002">
   <Points>40</Points>
  </Student>
 </Exam>
 <Exam course="TDA357" room="VV12" date="2010-12-18">
  <Student studentId="s001">
   <Points>30</Points>
  </Student>
 </Exam>
 <Exam course="TDA506" date="2010-12-19" />
</Exams>
```

a) Write a Document Type definition (DTD) for the XML that is given above. (2p)

b) Write an XPath expression that finds Student elements that are within Exam elements for the course TDA357, i.e.

```
<Student studentId="s003">
 <Points>30</Points>
</Student>
<Student studentId="s004">
 <Points>10</Points>
</Student>
<Student studentId="s001">
 <Points>30</Points>
</Student>
```

(1p)

c) Write an XQuery expression that finds the student identifier and exam date for all students who have passed an exam (i.e. scored at least 24 points) for course TDA357. The results should be ordered by student identifier. The result of this query should look as follows:

```
<DatabasesPass studentId="s001" date="2010-12-18" />
<DatabasesPass studentId="s003" date="2010-04-08" />
```

(2p)

CHALMERS UNIVERSITY OF TECHNOLOGY
Department of Computer Science and Engineering

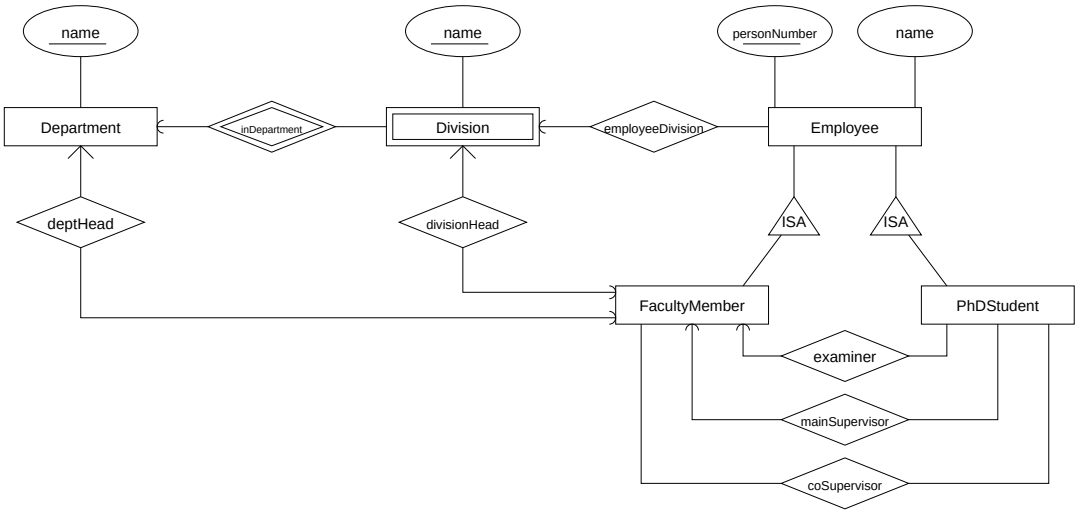
Examination in Databases, TDA357/DIT620

Saturday 18 December 2010, 08:30-12:30

Solutions

Updated 2011-12-07

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



b) $Departments(\underline{name}, head)$
$head \rightarrow FacultyMembers.personNumber$

$Divisions(\underline{dept}, \underline{name}, head)$
$dept \rightarrow Departments.name$
$head \rightarrow FacultyMembers.personNumber$

$Employees(\underline{personNumber}, name, dept, division)$
$(dept, division) \rightarrow Divisions.(dept, name)$

$FacultyMembers(\underline{personNumber})$
$personNumber \rightarrow Employees.personNumber$

$PhDStudents(\underline{personNumber}, examiner, mainSupervisor)$
$personNumber \rightarrow Employees.personNumber$
$examiner \rightarrow FacultyMembers.personNumber$
$mainSupervisor \rightarrow FacultyMembers.personNumber$

$CoSupervisors(\underline{student}, \underline{supervisor})$
$student \rightarrow PhDStudents.personNumber$
$supervisor \rightarrow FacultyMembers.personNumber$

Question 2. 12 p

a) i) AB is neither
ii) ABD is a key (and superkey)
iii) ABE is a superkey
iv) ACD is a key (and superkey)
v) AE is a key (and superkey)
vi) BCDEF is neither

b) i) All violate BCNF, since none has a superkey on the left hand side.

ii) Decompose on AB -> C
{AB}+ = {ABCF}

```
R1(_A,_B,C,F)
R2(A,B,D,E)
        (A,B) -> R1.(A,B)
```

Decompose R1 on B -> F
{B}+ = {BF}

```
R11(_B,F)
R12(_A,_B,C)
        B -> R11.B
```

Decompose R2 on E -> B
{E}+ = {BDE}

```
R21(B,D,_E)
R22(A,E)
    E -> R21.E
```

Should update references to decomposed relations.

c) i) B -> F
E -> F

ii) Compute the minimal basis

Remove E->F since we have E->B and B->F

Group together FDs with the same LHS

```
E -> BD
```

For each group, create a relation with the LHS as the key.

```
R1(_A,_B,C)
R2(_C,_D,E)
R3(B,_E,D)
R4(_B,F)
```

If no relation contains a key of R, add one relation containing only a key of R.

```
R5(_A,_E)
```

Question 3. 10 p

a) $Exams(\underline{course}, \underline{examDate}, examTime)$
$Students(\underline{studentId}, name)$
$registeredFor(\underline{student}, \underline{course}, \underline{examDate})$
$student \rightarrow Students.studentId$
$(course, examDate) \rightarrow Exams.(course, examDate)$

```
CREATE TABLE Exams (
    course   CHAR(6),
    examDate DATE,
    examTime CHAR(2) CHECK examTime IN ('AM', 'PM'),
    PRIMARY KEY (course, examDate)
);

CREATE TABLE Students (
    studentId CHAR(10) PRIMARY KEY,
    name      VARCHAR(30)
);

CREATE TABLE registeredFor (
    student  CHAR(10),
    course   CHAR(6),
    examDate DATE,
    PRIMARY KEY (student, course, examDate),
    FOREIGN KEY (student) REFERENCES Students(studentId)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    FOREIGN KEY (course, examDate) REFERENCES Exams(course, examDate)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
```

b)

```
CREATE ASSERTION AtMostTwoExams CHECK
  ( NOT EXISTS
      ( SELECT   student
        FROM     registeredFor
        GROUP BY student, examDate
        HAVING   COUNT(course) > 2
      )
  )
```

c)
```
CREATE TRIGGER FixClash
BEFORE INSERT ON registeredFor
REFERENCING NEW ROW AS new
FOR EACH ROW
DECLARE et CHAR(2)
WHEN ( EXISTS (
        SELECT E1.course
        FROM (registeredFor NATURAL JOIN Exams) E1, Exams E2
        WHERE student = new.student
              AND E2.course = new.course
              AND E1.examDate = new.examDate
              AND E1.course <> E2.course
              AND E1.examDate = E2.examDate
              AND E1.examTime = E2.examTime
     ) )
BEGIN
    SELECT examTime INTO et
    FROM   Exams
    WHERE  course = new.course AND examDate = new.examDate;

    IF (et = 'AM') THEN
        INSERT INTO SpecialExams
          VALUES(new.student, new.course, new.examDate, 'PM');
    ELSE
        INSERT INTO SpecialExams
          VALUES(new.student, new.course, new.examDate, 'AM');
    END IF;
END;
```

Question 4. 6 p

a) $\pi_{name}(Students \bowtie_{studentId=student} (\sigma_{course='TDA357' \bigwedge examData='2010-12-18'}(registeredFor)))$

b) $\tau_{course}(\gamma_{course,AVG(nrSt)\rightarrow avgSt}(\gamma_{course,examDate,COUNT(student)\rightarrow nrSt}(registeredFor)))$

Question 5. 8 p

a)
```
SELECT   name
FROM     Students, registeredFor A, registeredFor B
WHERE    studentId = A.student
         AND A.student = B.student
         AND A.course = 'TDA357'
         AND B.course = 'TIN092'
         AND A.examDate = B.examDate
ORDER BY name
```

b)
```
SELECT  examDate
FROM    Exams A
WHERE   course = 'TDA357'
        AND NOT EXISTS (
                SELECT  course
                FROM    Exams B
                WHERE  A.examDate = B.examDate
                       AND B.course <> 'TDA357' )
```

c)
```
CREATE VIEW V AS
    WITH ExamCounts AS (
        SELECT  course, examDate, COUNT(student) AS nrSt
        FROM    registeredFor
        GROUP BY course, examDate )
    SELECT  course, AVG(nrSt) as avgSt
    FROM    ExamCounts
    GROUP BY course
```

Question 6. 4 p

a) See section 6.6.4 of the course textbook.

b) See the fourth slide on page 7 of the lecture notes for lecture 10.

Question 7. 3 p

See the first slide on page 4 of the lecture notes for lecture 12.

Question 8. 5 p

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

<!DOCTYPE Exams [

<!ELEMENT Exams (Exam*) >

<!ELEMENT Exam (Student*) >
  <!ATTLIST Exam
    course CDATA #REQUIRED
    room   CDATA #IMPLIED
    date   CDATA #REQUIRED >

<!ELEMENT Student (Points) >
  <!ATTLIST Student
    studentId CDATA #REQUIRED >

<!ELEMENT Points (#PCDATA) >

]>
```

b)
```
/Exams/Exam[@course="TDA357"]/Student
```

c)
```
for $e in /Exams/Exam[@course="TDA357"],
    $s in $e/Student[Points > 24]
order $s/@studentId
return <DatabasesPass studentId="{$s/@studentId}" date="{$e/@date}" />
```