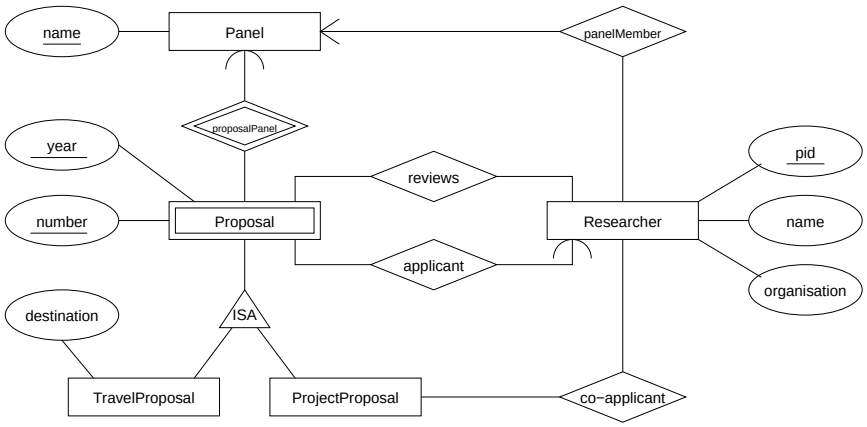
CHALMERS UNIVERSITY OF TECHNOLOGY
Department of Computer Science and Engineering

Examination in Databases, TDA357/DIT620
Saturday 19 December 2009, 08:30-12:30

Solutions

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



b) $Researchers(\underline{pid}, name, organisation)$

$Panels(\underline{name})$

$Proposals(\underline{panel}, \underline{year}, \underline{number}, applicant)$
$panel \rightarrow Panels.name$
$applicant \rightarrow Researchers.pid$

$TravelProposals(\underline{panel}, \underline{year}, \underline{number}, destination)$
$(panel, year, number) \rightarrow Proposals.(panel, year, number)$

$ProjectProposals(\underline{panel}, \underline{year}, \underline{number})$
$(panel, year, number) \rightarrow Proposals.(panel, year, number)$

$CoApplicants(\underline{panel}, \underline{year}, \underline{number}, \underline{researcher})$
$(panel, year, number) \rightarrow ProjectProposals.(panel, year, number)$
$researcher \rightarrow Researchers.pid$

$Reviews(\underline{panel}, \underline{year}, \underline{number}, \underline{researcher})$
$(panel, year, number) \rightarrow Proposals.(panel, year, number)$
$researcher \rightarrow Researchers.pid$

$PanelMembers(\underline{researcher}, panel)$
$researcher \rightarrow Researchers.pid$
$panel \rightarrow Panels.name$

Question 2. 10 p

a) i)
```
AB -> C
B -> D
CE -> F
D -> G
F -> A
```

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

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

Decompose R1 on B->D
{B}+ = {BDG}

        R11(_B,D,G)
        R12(A,B,C)
                B -> R11.B

Decompose R11 on  D->G
{D}+ = {DG}

        R111(_D,G)
        R112(_B,D)
                D -> R111.D

Decompose R2 on F -> A
{F}+ = {AF}

        R21(A,_F)
        R22(_B,_E,_F)
                F -> R21.F
```

b) i)
```
A,B,C,E and F are prime
```

ii)
```
B -> D
D -> G
```

iii)
```
Compute the minimal closure of F
        (Remove A->C if we have A->B and B->C, etc.)

Group together FDs with the same LHS

        ABE -> FG

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

        R1(A,B,C)
        R2(B,D)
        R3(C,E,F)
        R4(A,B,E,F,G)
        R5(D,G)
        R6(A,F)

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

        Relation R4 contains a key of R.
```

Question 3. 9 p

a) $Employees(\underline{empId}, name, year, salary, entitlement, branch)$
$ParentalLeave(\underline{employee}, \underline{startDay}, \underline{startYear}, endDay, endYear)$
$employee \rightarrow Employees.empId$

```
CREATE TABLE Employees (
    empId               INT PRIMARY KEY,
    name                VARCHAR(40),
    year                INT,
    salary              INT,
    entitlement         INT DEFAULT 30,
    branch              VARCHAR(40),
);

CREATE TABLE ParentalLeave (
    employee            INT,
    startDay            INT,
    startYear           INT,
    endDay              INT,
    endYear             INT,
    PRIMARY KEY (employee, startDay, startYear),
    FOREIGN KEY (employee) REFERENCES Employees(empId)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    CONSTRAINT CheckStartDay CHECK (StartDay BETWEEN 1 and 366),
    CONSTRAINT CheckEndDay CHECK (EndDay BETWEEN 1 and 366),
    CONSTRAINT CheckStartBeforeEnd (startYear < endYear OR
                (startYear = endYear AND startDay < endDay)),
);
```

b)
```
CREATE ASSERTION PayRangeLimit CHECK
   ( NOT EXISTS (
         SELECT A.empId
         FROM   Employees A, Employees B
         WHERE  A.salary - B.salary > 10000 ) )
```

or

```
CREATE ASSERTION PayRangeLimit CHECK
   ( ( (SELECT MAX(salary) FROM Employees)
         - (SELECT MIN(salary) FROM Employees) ) <= 10000);
```

c)
```
CREATE TRIGGER IncreaseEntitlement
BEFORE INSERT ON Employees
REFERENCING NEW ROW AS new
FOR EACH ROW
WHEN (new.year < 1970 AND new.entitlement < 35)
SET  new.entitlement = 35;
```

Question 4. 5 p

a) $\pi(Employees \bowtie_{empId=employee} (\sigma_{startYear=2007}(ParentalLeave)))$

b) $\tau_{branch}(\pi_{name,branch,salary}(\sigma_{salary=maxSal}($
$Employees \bowtie (\gamma_{branch,MAX(salary)\rightarrow maxSal}(Employees)))))$

Question 5. 7 p

a)
```
SELECT DISTINCT empId, name
FROM   Employees JOIN (
           SELECT employee
           FROM   ParentalLeave
           WHERE  startYear < endYear ) ON employee=empId;
```

b)
```
SELECT A.name, B.name
FROM   Employees A, Employees B
WHERE  A.empId <> B.empId
       AND A.year = B.year
       AND A.entitlement = B.entitlement
       AND A.salary < B.salary;
```

c)
```
CREATE VIEW V AS
SELECT   branch, startYear, COUNT(empId) AS numParents
FROM     Employees JOIN ParentalLeave on empId = employee
GROUP BY branch, startYear
```

Question 6. 3 p

a) Either T1 runs completely before T2, or vice versa. In either case, the output is: 2,2

b) In addition to the above, several other orderings are now possible.
T1B will see at least the same tuples as T1A.
T2A,T1A,T1B,T2B gives 1,1
T2A,T1A,T2B,T1B gives 1,2
T1A,T2A,T1B,T2B gives 2,2
T1A,T2A,T2B,T1B gives 2,3

Question 7. 6 p

a)
i) task 1: 2, task 2: 30, task 3: 30.
ii) task 1: 4, task 2: 30, task 3: 11.
iii) task 1: 4, task 2: 6, task 3: 30.
iv) task 1: 6, task 2: 6, task 3: 11.

b) (iv) has lowest cost of 6.5, with (iii) next with cost of 7.4.

Question 8. 8 p

a) (Alternative tags and structures are also acceptable.
Note: Branch numbers have been modified so that they start with a letter.
We'll accept solutions where these begin with digits.)

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

<!DOCTYPE Question8 [

<!ELEMENT Question8 (Employees, Branches)>

<!ELEMENT Employees (Employee*)>
  <!ELEMENT Employee EMPTY>
    <!ATTLIST Employee
      name   ID    #REQUIRED
      branch IDREF #REQUIRED
      salary CDATA #REQUIRED >

<!ELEMENT Branches (Branch*)>
  <!ELEMENT Branch EMPTY>
    <!ATTLIST Branch
      number ID    #REQUIRED
      city   CDATA #REQUIRED >

]>
```

b)

```
<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>
```

c) (Alternatives are acceptable. Two suggestions are given here.)

```
/Question8/Branches/Branch[@city="Paris"]

//Branch[@city="Paris"]
```

d) (Alternatives are acceptable.)

```
FOR $e IN /Question8/Employees/Employee[@salary>30000]
LET $place := $e/@branch => Branch/@city
RETURN <Result>{$e/name}: {$place}</Result>
```