CHALMERS UNIVERSITY OF TECHNOLOGY
Department of Computer Science and Engineering

Examination in Databases, TDA357/DIT620

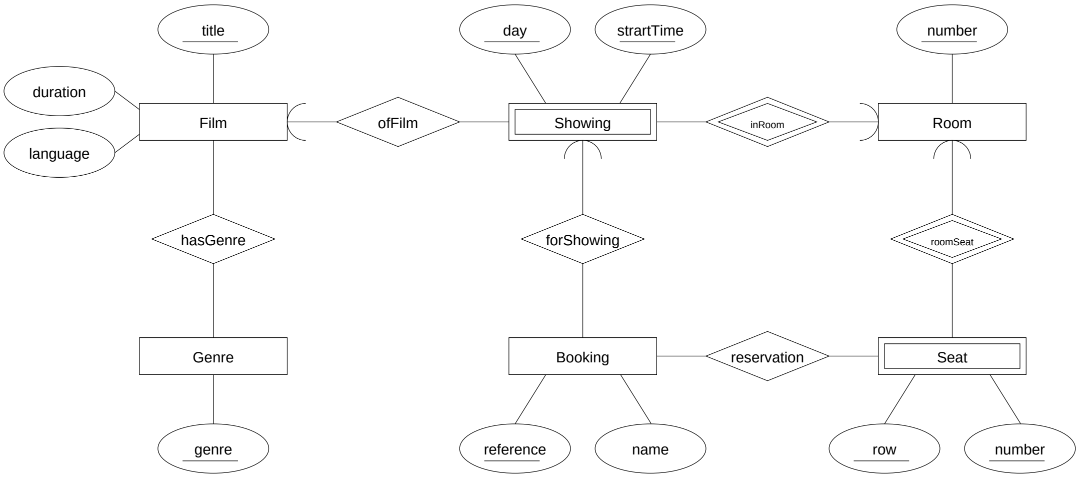
Wednesday 14 December 2011, 08:30-12:30

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

```
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>
```