CHALMERS UNIVERSITY OF TECHNOLOGY
Department of Computer Science and Engineering

Examination in Databases, TDA357/DIT620

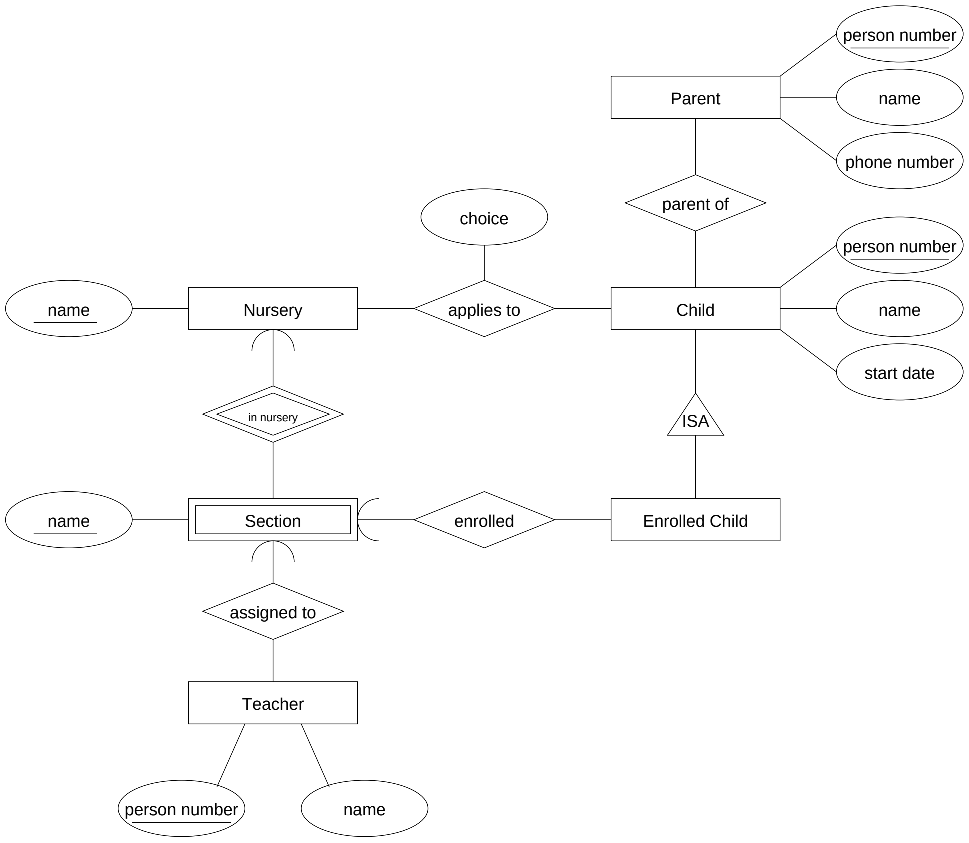
Tuesday 17 December 2013, 14:00-18:00

Solutions

Updated 2013-12-18

Question 1. 12 p

a) (Here is one suggestion. Several other designs are also accepted. For example, modelling 'Teacher', 'Parent' and 'Child' as subclasses of 'Person'.)



b) $Nurseries(\underline{name})$

$Sections(\underline{nursery}, \underline{name})$
  $nursery \rightarrow Nurseries.name$

$Teachers(\underline{personNumber}, name, nursery, section)$
  $(nursery, section) \rightarrow Sections.(nursery, name)$

$Parents(\underline{personNumber}, name, phoneNumber)$

$Children(\underline{personNumber}, name, startDate)$

$EnrolledChildren(\underline{personNumber}, nursery, section)$
  $(nursery, section) \rightarrow Sections.(nursery, name)$

$AppliesTo(\underline{child}, \underline{nursery}, choice)$
  $child \rightarrow Children.personNumber$
  $nursery \rightarrow Nurseries.name$

$ParentOf(\underline{parent}, \underline{child})$
  $parent \rightarrow Parents.personNumber$
  $child \rightarrow Children.personNumber$

Question 2. 11 p

a) In addition to the FDs listed in the question, we also have:

```
          AB -> C
          AC -> B
          AD -> B
          AD -> C
          BD -> C
          ABD -> C
          ACD -> B
          BCD -> A

     Superkeys: AD, BD, ABD, ACD, BCD, ABCD
     Keys:      AD, BD
```

```
b) i)   FDs violating BCNF: A->B, B->C, A->C, AB->C, AC->B

  ii)   FDs violating 3NF:  B->C, A->C, AB->C
```

```
c)  Decompose R on A->B
    {A}+ = {ABC}

            R1(_A,B,C)
            R2(_A,_D)
                    A -> R1.A

    Decompose R1 on B->C
    {B}+ = {BC}

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

    Update reference for R2: A -> R12.A
```

```
d)  (a1,b2,c2,d2)
    (a2,b1,c1,d3)
    (a2,b3,c3,d1)
```

Question 3. a) $Offices(\underline{city}, supplement)$
9 p
$Departments(\underline{city}, \underline{dname}, departmentHead)$
$city \rightarrow Offices.city$
$departmentHead \rightarrow Employees.empId$
$Employees(\underline{empId}, name, salary, dept, city)$
$(city, dept) \rightarrow Departments.(city, dname)$

```
CREATE TABLE Offices (
    city        VARCHAR(20) PRIMARY KEY,
    supplement  INT DEFAULT 0,
);

CREATE TABLE Departments (
    city           VARCHAR(20),
    dname          VARCHAR(20),
    departmentHead CHAR(10),
    PRIMARY KEY (city, dname),
    FOREIGN KEY (city) REFERENCES Offices(city)
    FOREIGN KEY (departmentHead) REFERENCES Employees(empId)
        ON DELETE SET NULL
        ON UPDATE CASCADE,
);

CREATE TABLE Employees (
    empId          CHAR(10) PRIMARY KEY,
    name           VARCHAR(30),
    salary         INT,
    dept           VARCHAR(20),
    city           VARCHAR(20),
    FOREIGN KEY (city, dept) REFERENCES Departments(city, dname)
);
```

Several of the solutions we saw used the policy "CASCADE" instead of "SET NULL" for the foreign key constraint on attribute departmentHead in table Departments. Consider what would be the consequences of this.

The solution shown above would give an error if executed, due to the forward references from the Departments table definition to the Employees table, which hasn't been created yet. This complication was ignored when marking the exam. In practice, we could omit this foreign key constraint when creating the Departments table, and then add this constraint after the Employees table has been created. This can be done using the ALTER TABLE statement, e.g.

```
ALTER TABLE Departments ADD CONSTRAINT departmentREFemployee
    FOREIGN KEY (departmentHead) REFERENCES Employees(empId)
    INITIALLY DEFERRED DEFERRABLE;
```

For more information on this, see the section on "Deferring Constraint Checking" on the website for the course textbook:
http://infolab.stanford.edu/~ullman/fcdb/oracle/or-triggers.html

b)
```
CREATE ASSERTION HeadOfOwnDept CHECK
   ( NOT EXISTS (
         SELECT departmentHead
         FROM   Departments JOIN Employees ON departmentHead = empId
         WHERE  dname <> dept
                OR Departments.city <> Employees.city ) )
```

c)
```
CREATE PROCEDURE Merge (
    IN city1 VARCHAR(20),
    IN dept1 VARCHAR(20),
    IN city2 VARCHAR(20),
    IN dept2 VARCHAR(20)
)
BEGIN
    IF ( SELECT COUNT(empId)
         FROM   Employees
         WHERE  city = city1 AND dept = dept1 ) >
       ( SELECT COUNT(empId)
         FROM   Employees
         WHERE  city = city2 AND dept = dept2 )
    THEN
        UPDATE Departments
        SET    departmentHead =
                   ( SELECT departmentHead
                     FROM   Departments
                     WHERE  city = city1 AND dname = dept1 )
        WHERE  city = city2 AND dept = dept2;
    ENDIF;

    UPDATE Employees
    SET    city = city2,
           dept = dept2
    WHERE  city = city1 AND dept = dept1;

    DELETE FROM Departments WHERE city = city1 AND dname = dept1;
END;
```

Question 4. 6 p

a)
$$\tau_{name}(\pi_{empId,name,salary+supplement}(Employees \bowtie Offices))$$

b) If we assume that all sales departments have at least one employee:

$$\gamma_{city,AVG(salary)\rightarrow avgSalary}(\sigma_{dept=\text{"}sales\text{"}}(Employees))$$

If there can be sales departments with no employees, we might want to include those in the result, with ‘0’ as the average. This can be done by forming the union of the relational algebra expression given above with:

$$\pi_{city,0}(\pi_{city}(\sigma_{dname=\text{"}sales\text{"}}(Departments)) - \pi_{city}(\sigma_{dept=\text{"}sales\text{"}}(Employees)))$$

Question 5. 10 p

a)
```
SELECT   empId, name, salary + supplement AS totalSalary
FROM     Employees NATURAL JOIN Offices
ORDER BY name
```

b) i)
```
SELECT  dname
FROM    Departments
WHERE   city = "London"
        AND dname NOT IN (
                SELECT  dname
                FROM    Departments
                WHERE   city = "Paris" )
```

ii)
```
( SELECT  dname
  FROM    Departments
  WHERE   city = "London" )
      EXCEPT
( SELECT  dname
  FROM    Departments
  WHERE   city = "Paris" )
```

c)
```
CREATE VIEW SalaryBill AS
SELECT   city, SUM(salary + supplement) AS amount
FROM     Employees NATURAL JOIN Offices
GROUP BY city
```

Question 6. 5 p

a) The result printed by transaction T1 could be different if transaction T1 and T3 are run concurrently. Good answers will discuss the concept of *phantoms* (see Example 6.47 in the course textbook) and the schedule of operations that causes different results to be printed.

b) i) task 1: 2
task 2: 30

ii) task 1: 4
task 2: 6

iii) It would be better to have an index on *city* (cost: 420 vs. 480).

Question 7. 7 p

a) Corrected DTD is:

```
<!DOCTYPE A [
<!ELEMENT A (B*) >
<!ELEMENT B (C) >
<!ELEMENT C (#PCDATA) >
<!ATTLIST A
  a1 CDATA #REQUIRED >
<!ATTLIST B
  b1 CDATA #REQUIRED
  b2 CDATA #IMPLIED >
<!ATTLIST C
  c1 CDATA #REQUIRED >
]>
```

b) i)

```
<B b1="B1" b2="15">
  <C c1="red">first</C>
</B>
<B b1="B4" b2="35">
  <C c1="red">fourth</C>
</B>
```

ii)

```
<C c1="blue">third</C>
<C c1="red">fourth</C>
```

c)

```
<Result>
  {
    for $b in (doc("exam.xml")//B)
    order by $b/C/@c1
    return <C c1="{$b/C/@c1}"><B b1="{$b/@b1}" /></C>
  }
</Result>
```