**QUESTION 1 (6 points)**

Relation ADDRESS has attributes STREET, CITY, STATE and ZIP.

For any given zipcode, there is just one city and state. Also, for any given street, city, and state, there is just one zipcode.

a) Infer all possible functional dependencies for this relation. (2)

Let us denote the attributes STREET, CITY, STATE, ZIP as A, B, C and D respectively. Then D→BC and ABC→D.

\[ A^+ = \{A\}; \ B^+ = \{B\}; \ C^+ = \{C\}; \ D^+ = \{DBC\} \]
\[ AB^+ = \{AB\}; \ AC^+ = \{AC\}; \ AD^+ = \{ABCD\}; \ BC^+ = \{BC\}; \ BD^+ = \{BDC\}; \ CD^+ = \{CDB\}; \ ABC^+ = \{ABCD\}; \ ABD^+ = \{ABCD\}; \ ACD^+ = \{ABCD\}; \ BCD^+ = \{BCD\} \]

FD’s: D→BC; AD→BC; BD→C; CD→B; ABC→D; ADB→C; ACD→B;

b) Which are possible minimal keys? (1)

ABC, AD

c) Give a reason why the relation is not in BCNF? (1)

D→BC and D is not a key

d) Give a better design of the relation in BCNF. (2)

R₁(BCD); R₂(DA)

**QUESTION 2 (6 points)**

Consider the following relational schema

employee(emp-name, street, city)
works(emp-name, comp-name, salary)
company(comp-name, city)
manages(emp-name, man-name)

(The first attribute is the primary key in each of the relations)

Give an SQL expression for the following queries

a. Delete all tuples in the works relation for employees of BCV. (1)

DELETE FROM works WHERE comp_name='BCV';
b. Give all managers of UBS a 5 percent raise.  \((1)\)

```
UPDATE works
SET salary=salary*1.05
WHERE comp_name='UBS'
AND emp_name IN (  
    SELECT man_name  
    FROM manages);  
```

c. Give all managers of UBS a 10 percent raise unless the salary becomes greater than CHF 150.000; in such cases give only a 3 percent raise.  \((2)\)

```
UPDATE works
SET salary = salary * 1.03
WHERE comp_name='UBS'
AND salary*1.1 > 150000
AND emp_name IN (SELECT man_name FROM manages);

UPDATE works
SET salary = salary * 1.1
WHERE comp_name='UBS'
AND salary*1.1 <= 150000
AND emp_name IN (SELECT man_name FROM manages);
```

Please note that the order matters.

d. For all employees, give a salary rise of 10% if they live in a different city than where their company is located in, provided their new salary does not exceed their manager's present salary.  \((2)\)

```
CREATE VIEW managersalary AS
SELECT manages.emp_name, manages.man_name, works.salary as man_salary
FROM works,manages
WHERE manages.man_name=works.emp_name

CREATE VIEW raise AS
SELECT emp_name from employee,works,company
WHERE employee.emp_name=works.emp_name
AND works.comp_name=company.comp_name
AND NOT company.city=employee.city
AND emp_name IN (  
    SELECT emp_name FROM managersalary,works
    WHERE mansalary.emp_name=works.emp_name
    AND works.salary*1.1<managersalary.man_salary);  

UPDATE works SET salary=salary*1.1
WHERE emp_name IN (SELECT emp_name FROM raise);  
```
QUESTION 3 (6 points)

Consider the following two transactions

T1: read(A);
    read(B);
    if A=0 then B:=B+1;
    write(B);

T2: read(B);
    read(A);
    if B=0 then A:=A+1;
    write(A);

a. Give an interleaved execution of T1 and T2 that produces a non-serializable schedule. (1)

b. Is there an interleaved execution of T1 and T2 that produces a serializable schedule? (1)

NO

c. Add lock and unlock instructions to transactions T1 and T2, so that they observe the two-phase locking protocol. Assume that transactions can upgrade shared locks to exclusive locks (if the acquisition of the lock is compatible with locks acquired by other transactions) and that they acquire always the least restrictive lock needed for performing the next action. (2)

<table>
<thead>
<tr>
<th>T1</th>
<th>T2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slock(A)</td>
<td>Slock(B)</td>
</tr>
<tr>
<td>read(A)</td>
<td>read(B)</td>
</tr>
<tr>
<td>Slock(B)</td>
<td>Slock(A)</td>
</tr>
<tr>
<td>read(B)</td>
<td>read(A)</td>
</tr>
<tr>
<td>if A=0 then B:=B+1</td>
<td>if B=0 then A:=A+1</td>
</tr>
<tr>
<td>Xlock(B)</td>
<td>Xlock(A)</td>
</tr>
<tr>
<td>write(B)</td>
<td>write(A)</td>
</tr>
<tr>
<td>unlock(A)</td>
<td>unlock(B)</td>
</tr>
<tr>
<td>unlock(B)</td>
<td>unlock(A)</td>
</tr>
</tbody>
</table>
b. Can the execution of these transactions result in a deadlock? Substantiate your answer by an example (if yes) or a proof (if no).

YES

<table>
<thead>
<tr>
<th>Slock(A)</th>
<th>Slock(B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>read(A)</td>
<td>read(B)</td>
</tr>
<tr>
<td>Slock(B)</td>
<td>Slock(A)</td>
</tr>
<tr>
<td>read(B)</td>
<td>read(A)</td>
</tr>
<tr>
<td>if A=0 then B:=B+1</td>
<td>if B=0 then A:=A+1</td>
</tr>
</tbody>
</table>

Xlock(B)

write(B)

Xlock(A)

write(A)

unlock(A)

unlock(B)

unlock(B)

unlock(A)

QUESTION 4 (6 points)

Consider the DTD below:

```
<!ELEMENT Courses (Course*)>
<!ELEMENT Course (Title, Exercise?, ((Time, Place) | Website), Teacher*)>
<!ELEMENT Exercise (Type, Assistant*, (Number, Date)*)>
<!ELEMENT Teacher (Name, Phone*)>
```

Elements that are not defined are #PCDATA. The order of all elements in the document is irrelevant.
a. Design relational schema $S$ to store XML documents corresponding to this DTD with the following properties:

The schema must be in first normal form and duplicate tuples are not allowed in tables (note: this is different to practical SQL implementations). The schema must allow correct reconstructing of the XML document from the data stored in the relational representation. The schema maps each PCDATA element into a separate attribute. Relational schema $S$ does not require any NULL values in order to represent any possible instance of the DTD.

Identify the primary keys and foreign keys. (3)

Teachers(TeacherID, Name)
TeacherPhones(TeacherID, Phone)
Exercises(ExerciseID, Type, CourseID)
ExerciseAssistants(ExerciseID, Assistant)
ExerciseDetails(ExerciseID, Number, Date)

Courses(CourseID, Title)
CourseTeachers(CourseID, TeacherID)
CourseWebsite(CourseID, Website)
CourseDetails(CourseID, Time, Place)

The underlined fields are PK’s. Other TeacherID, ExerciseID and CourseID fields are FK’s to tables Teachers, Exercises and Courses respectively.

b. Consider the following data instance $X$.

```xml
<Courses>
  <Course>
    <Title> Course1 </Title>
    <Exercise>
      <Type> Theory </Type>
      <Assistant> Assistant1 </Assistant>
      <Number> Exercise1 </Number><Date> 1.1.2004 </Date>
      <Number> Exercise2 </Number><Date> 7.1.2004 </Date>
    </Exercise>
    <Time> 8:00 AM </Time>
    <Place> Room1 </Place>
    <Teacher>
      <Name> Prof1 </Name>
      <Phone> Number1 </Phone>
    </Teacher>
  </Course>

  <Course>
    <Title> Course2 </Title>
    <Time> 10:00 AM </Time>
    <Place> Room2 </Place>
    <Teacher>
      <Name> Prof2 </Name>
      <Phone> Number2 </Phone>
      <Phone> Number3 </Phone>
    </Teacher>
  </Course>
</Courses>
```
For each table in S indicate how many tuples it contains when we populate the database with X. (1)

Teachers: 3
TeacherPhones: 5

Exercises: 4
ExerciseAssistants: 1
ExerciseDetails: 4

Courses: 3
CourseTeacher: 4
CourseWebsite: 1
CourseDetails: 2

c. Consider the query Q:

for
  $x$ in /Courses/Course[Exercise[count(Dates)>10]],
  $y$ in $x$/Exercise
where
  $y$/Type = "practical"
return
  { $y/Assistants/text() } 

Translate query Q into an SQL query over the schema S. (2)