CHALMERS UNIVERSITY OF TECHNOLOGY
Department of Computer Science and Engineering

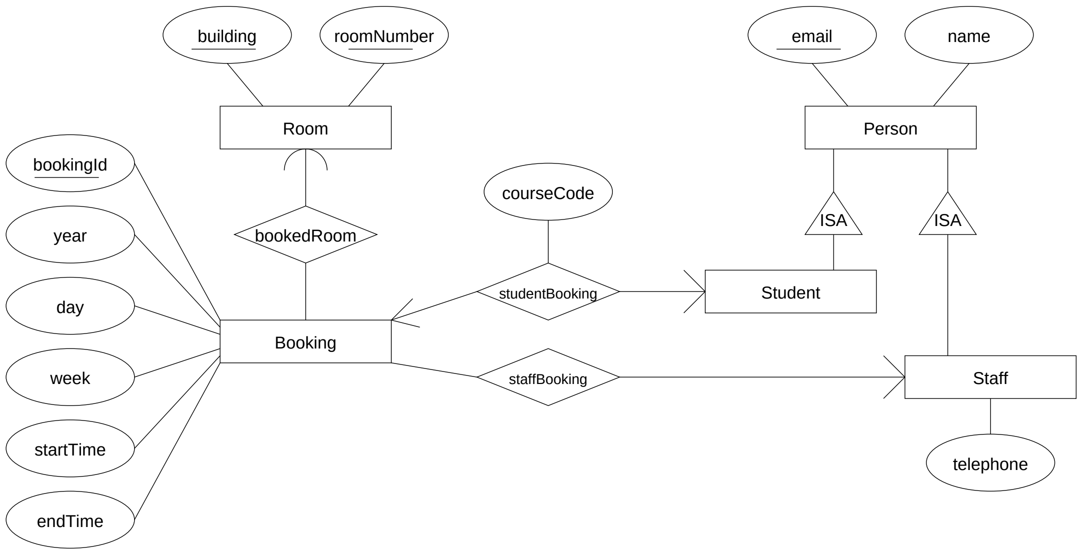
Examination in Databases, TDA357/DIT620

Tuesday 13 January 2015, 14:00-18:00

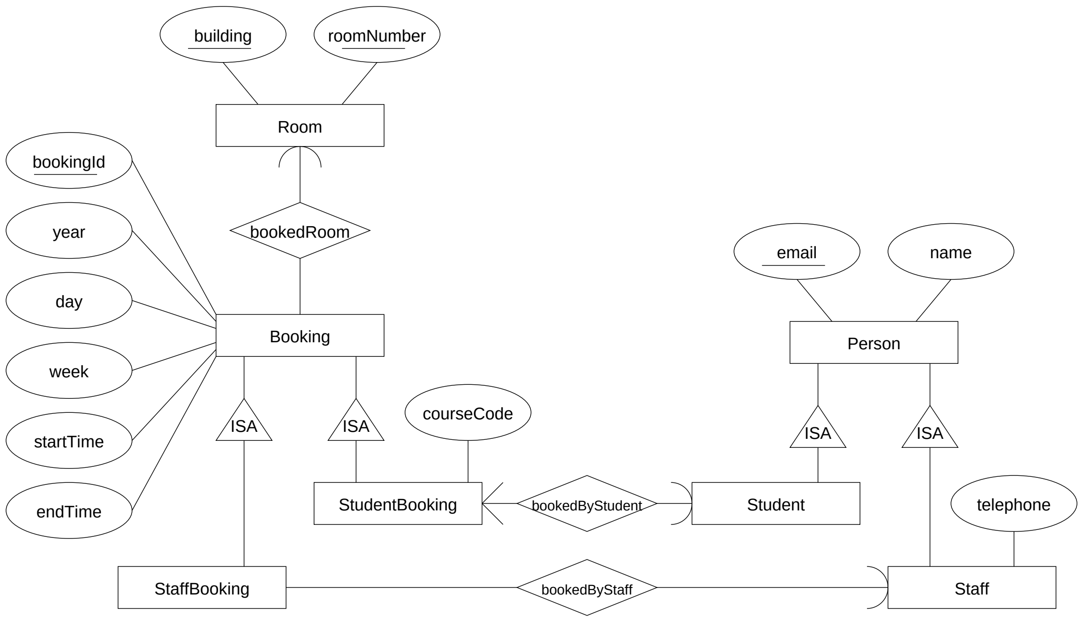
Solutions

Updated 2015-03-30

Question 1. a) This suggestion is acceptable:
12 p



However, that suggestion does not model the multiplicity of the relationships between booking, students and staff in a good way. These can be represented in a better way with subclasses of booking:



b) $E1(\underline{a}, \underline{b}, c)$
$E2(\underline{a}, \underline{b}, d)$
  $(a, b) \rightarrow E1.(a, b)$
$E3(\underline{a}, \underline{b})$
  $(a, b) \rightarrow E2.(a, b)$
$E4(\underline{a}, \underline{b}, \underline{e}, f)$
  $(a, b) \rightarrow E2.(a, b)$
$E5(\underline{g}, h, a, b, e)$
  $(a, b, e) \rightarrow E4.(a, b, e)$
$R2(\underline{a}, \underline{b}, \underline{g})$
  $(a, b) \rightarrow E3.(a, b)$
  $g \rightarrow E5.g$

Question 2. a)
10 p

```
i) ABC->F and ABC->G do not violate BCNF since their left sides are keys.
   The other 5 FDs violate BCNF.

ii) Decompose R on A->E
    {A}+ = {AEF}

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

    Decompose R2 on AB->D
    {AB}+ = {ABD}

        R21(_A,_B,D)
                A -> R1.A
        R22(A,B,C,G)
                (A,B) -> R21(A,B)

    Decompose R1 on E->F
    {E}+ = {E,F}

        R11(_E,F)
        R12(E,_A)
                E -> R11.E

    Decompose R22 on G->B
    {G}+ = {GB}

        R221(_G,B)
        R222(_A,_C,_G)
                G -> R221.G

Update reference for R21: A -> R12.A
```

b)

```
 i) A, B, C, D, G

 ii) A->E, E->F

iii) R1(_A,E)
     R2(_A,_B,D)
     R3(_A,_B,_C,F,G)
     R4(_C,_D,G)
     R5(_E,F)
     R6(_G,B)
```

Question 3. 10 p

a)
```
CREATE TABLE Programmes (
    code        CHAR(5) PRIMARY KEY,
    name        VARCHAR(50),
    department  VARCHAR(50),
    numPlaces   INT
);

CREATE TABLE Applicants (
    name        VARCHAR(30),
    address     VARCHAR(50),
    appNumber   INT PRIMARY KEY
);

CREATE TABLE AppliesFor (
    applicant REFERENCES Applicants(appNumber),
    programme REFERENCES Programmes(code),
    choiceNumber INT CHECK (choiceNumber BETWEEN 1 AND 4),
    meritScore   INT DEFAULT 0 CHECK (meritScore BETWEEN 0 AND 1000),
    status VARCHAR(30) DEFAULT 'unprocessed'
                   CHECK (status IN ('unprocessed', 'offered',
                            'accepted', 'declined',
                            'offer withdrawn', 'rejected') ),
    PRIMARY KEY (applicant, programme),
    CONSTRAINT choices_unique UNIQUE (applicant, choiceNumber)
);
```

b)
```
CREATE ASSERTION ConsecutiveChoices CHECK
   ( NOT EXISTS (
         SELECT   applicant
         FROM     AppliesFor
         GROUP BY applicant
         HAVING   MAX(choiceNumber) > COUNT(choiceNumber) ) )
```

c)
```
CREATE OR REPLACE TRIGGER CourseFull
AFTER UPDATE OF status ON AppliesFor
REFERENCING NEW AS newrow
FOR EACH ROW
WHEN ( newrow.status = "accepted" )
BEGIN
  IF ( ( SELECT COUNT(applicant)
      FROM   AppliesFor
      WHERE  programme = :newrow.programme
         AND status = "accepted" ) >= ( SELECT  numPlaces
                                        FROM Programmes
                                        WHERE code = :newrow.programme ) ) THEN

        UPDATE AppliesFor
        SET    status = "rejected"
        WHERE  status = "unprocessed" AND programme = :newrow.programme;
    END IF;
END;
```

Privilege UPDATE of attribute status in table AppliesFor is needed.

Question 4. 6 p

a) $\pi_{Applicants.name, Programmes.name}(Applicants \bowtie_{applicant=appNumber}$
$((\sigma_{department="Physics"} Programmes) \bowtie_{code=programme} (\sigma_{choiceNumber=1} AppliesFor)))$

b) $R := \gamma_{programme, COUNT(appliocant) \rightarrow numApplicants}(\sigma_{choiceNumber=1} AppliesFor)$

$\pi_{programme}(\sigma_{numApplicants=maxApplicants(\gamma_{MAX(numApplicants) \rightarrow maxApplicants} R)} R)$

Question 5. 9 p

a)
```
SELECT  Applicants.name, Programmes.name
FROM    Applicants, Programmes, AppliesFor
WHERE   applicant = appNumber
        AND code = programme
        AND department = 'Physics'
        AND choiceNumber = 1
```

b)
```
WITH R AS ( SELECT  programme, COUNT(applicant) AS numApplicants
            FROM    AppliesFor
            WHERE   choiceNumber = 1 )
SELECT Programme
FROM   R
WHERE  numApplicants = ( SELECT  MAX(numApplicants)
                         FROM    R )
```

c)
```
WITH R1 AS
    ( SELECT  A.applicant AS appNumber
      FROM    AppliesFor A JOIN AppliesFor B ON A.applicant = B.applicant
      WHERE   A.programme = 'MPALG'
              AND B.programme = 'MPCSN'
              AND A.choiceNumber < B.choiceNumber )
WITH R2 AS
    ( SELECT  appNumber
      FROM    Applicants
      WHERE   'MPALG' IN (
                  SELECT  programme
                  FROM    AppliesFor
                  WHERE   applicant = appNumber )
              AND 'MPCSN' NOT IN (
                  SELECT  programme
                  FROM    AppliesFor
                  WHERE   applicant = appNumber )
SELECT  COUNT(name)
FROM    R1 UNION R2
```

Question 6. 5 p

a) See the lecture slides on transactions. In short phantoms can occur when (i) transaction A reads data satisfying some <search conditions>, then (ii) transaction B creates data items satisfying A's <search conditions>, then A repeats a read with the same <search conditions>.

b) In the normal case, T_5 returns a value one larger than T_2 (if place is accepted) or the same as T_2 (if place is declined).

Larger values for T_5 can occur due to phantoms (see part (a)) for transactions run with isoltion levels REPEATABLE READ, READ COMMITTED or READ UNCOMMITTED.

Running transactions with isolation level SERIALIZABLE is the only way to avoid possible problems with phantoms. But step T_4 involves waiting for a reply from the applicant, and it would be unacceptable for other transactions to have to wait.

Question 7. 8 p

a)
```
<!DOCTYPE Question7 [

<!ELEMENT Question7 (Applicants, Choices)>

<!ELEMENT Applicants (Applicant*)>
  <!ELEMENT Applicant EMPTY>
    <!ATTLIST Applicant
      name   CDATA #REQUIRED
      appNum ID    #REQUIRED >

<!ELEMENT Choices (Choice*)>
  <!ELEMENT Choice EMPTY>
    <!ATTLIST Choice
      applicant  IDREF #REQUIRED
      code       CDATA #REQUIRED
      choiceNum  CDATA #REQUIRED
      meritScore CDATA #REQUIRED>

]>
```

b)
```
//Choice[@choiceNum="1" and @meritScore>800]
```

c)
```
<Question7>
  <Applicant appNum="a1" name="Andersson">
    <Choice meritScore="750" choiceNum="1" code="MPSOF"/>
    <Choice meritScore="750" choiceNum="2" code="MPALG"/>
    <Choice meritScore="800" choiceNum="3" code="MPCSN"/>
  </Applicant>
  <Applicant appNum="a2" name="Jonsson">
    <Choice meritScore="700" choiceNum="1" code="MPALG"/>
  </Applicant>
  <Applicant appNum="a3" name="Larsson">
    <Choice meritScore="850" choiceNum="1" code="MPCSN"/>
    <Choice meritScore="850" choiceNum="2" code="MPALG"/>
  </Applicant>
</Question7>
```

d)
```
<Question7>
  {
    let $d := doc("exam.xml")
    for $a in $d//Applicant
    let $choices := (
        for $c in $d//Choices/Choice[@applicant = $a/@appNum]
        return <Choice code="{$c/@code}"
                       choiceNum="{$c/@choiceNum}"
                       meritScore="{$c/@meritScore}" /> )
    return <Applicant name="{$a/@name}" appNum="{$a/@appNum}" >
             {$choices}
           </Applicant>
  }
</Question7>
```