Uitwerking Proeftentamen Databases-1, 4 april, 2006

0. Toetsing van enkele basisnoties

Merk op dat dergelijke vragen *niet* op het tentamen zelf gevraagd zullen worden. Ik wilde slechts nagaan in hoeverre deze elementaire zaken bekend zijn.

1. Gegeven de verzameling R={(a,b), (b,e), (c,d),(d,e),(e,f),(e,g)}, en S={(x,y),(y,e),(e,f)} geef
   a. R+. De transitieve afsluiting van R, dus als (x,y) en (y,z) in R, dan zitten (x,y),(y,z) en (x,z) in R+.
      R+ = {(a,b), (b,e), (c,d),(d,e),(e,f),(e,g), (a,e),(a,f),(a,g),(b,f),(b,g), (e,e),(e,f),(e,g),(d,f),(d,g)}
   b. |R|. Het aantal elementen in R, dat is dus 6.
   c. R – S (ook wel genoteerd als R \ S). Het verschil van R en S, ofwel de elementen van R die niet in S zitten.
      R – S = {(a,b), (b,e), (c,d),(d,e),(e,g)} (alleen (e,f) valt weg).
   d. R ∩ S = {(e,f)}, de doorsnede van R en S
   e. R ∪ S = {={(a,b), (b,e), (c,d),(d,e),(e,f),(e,g),(x,y),(y,e)}, de vereniging van R en S.

2. Herschrijf de volgende logische expressies:
   a. f => g , zodat je ¬ (NOT) en ∨ (OR) gebruikt
      f => g is equivelent met ¬ f ∨ g, dit kan je controleren met een *waarheidstabel*:

f	g	f => g	¬ f ∨ g
0	0	1	1
0	1	1	1
1	0	0	0
1	1	1	1

   b. ¬∀x ¬ f(x), zodat je alleen ∃ (ER-IS) en ¬ (NOT) gebruikt
      ∀x f(x) = ¬ ∃ ¬ f(x) en dus ¬¬ A = A, dus
      ¬∀x ¬ f(x) = ¬¬ ∃ x¬¬ f(x) = ∃ x f(x)

1. Ontwerp in EER (Extended ER!) de database van een museum, die voldoet aan de volgende eisen:

- Het museum heeft een collectie van ART_OBJECTs, Elke ART_OBJECT heeft een unieke IdNo, een ARTIST (indien bekend), een jaartal (indien bekend), een titel en een beschrijving;
- De collectie van ART_OBJECTs wordt onderverdeeld in drie typen, PAINTINGs, STATUEs plus nog een rest-type OTHER;
- Een PAINTING heeft een PaintType (zoals “olie”, “waterverf”, etc.), Material (zoals “papier”, “doek”, “hout”, etc.) en Style (“modern”, “abstract”, etc.);

- Een STATUE kent alleen Material (“hout”, “steen”, etc.) en Style;
- Een ART_OBJECT in de OTHER kent als eigenschappen Material, Type (zoals “print”, “photo”, etc.) en Style.
- ART_OBJECTs zijn daarnaast ook onderverdeeld in een PERMANENT_COLLECTION als ze eigendom zijn van het museum (in welk geval ze drie eigenschappen hebben, nl. de datum van acquisitie, de betaalde prijs en of ze uitgestald zijn dan wel in het archief staan opgeslagen) en BORROWED (in welk geval de COLLECTION is aangegeven waar het object deel van uitmaakt, de datum van lening, en de datum waarop het geretourneerd moet worden) en tot slotte ON-LOAN waarin de uitgeleende objecten zitten. ON-LOAN heeft dezelfde eigenschappen als BORROWED.
- Van elke ARTIST worden de volgende eigenschappen bijgehouden: Name (bestaande uit een FirstName en een LastName), DateBorn (indien bekend), DateDied (indien de artiest is overleden en de datum is bekend), CountryOfOrigin, Description en Style. De naam wordt geacht uniek te zijn.
- Een COLLECTION staat voor een ander museum waarmee objecten worden uitgewisseld (die dan zijn opgenomen als BORROWED of ON-LOAN). Een COLLECTION heeft als eigenschappen: Name (uniek), Description, Address, Phone. Tot slot, elke COLLECTION heeft haar eigen CONTACT-PERSON.
- Een CONTACT-PERSON heeft een Name (FirstName, LastName) en een Adress. Er worden alleen CONTACT-PERSON’s bijgehouden als er een COLLECTION bijhoort.
- Teneinde (potentiële) bezoekers te kunnen informeren wordt er ook een collectie SUBSCRIBERS bijgehouden met een SubscriptionNumber (uniek), Name (FirstName, LastName), Address, Phone en de favorite ARTIST’s.

(Deze uitwerking volgt nog)

2. Bekijk het bijgevoegde plaatje van deze vraag

a. Geef het relationele model, in abstracte zin, dus alleen de tabelnamen en de bijbehorende attributen

Alternatief 1

EMPLOYEE(NAME,SSN,BDATE,ADDRESS)
SECRETARY(SSN,TSPEED)
SALESMAN(SSN,LIMIT)
ENGINEER(SSN,SPECIALITY)

Alternatief 2

SECRETARY(NAME,SSN,BDATE,ADDRESS,TSPEED)

SALESMAN(NAME,SSN,BDATE,ADDRESS,LIMIT)
ENGINEER(NAME,SSN,BDATE,ADDRESS,SPECIALITY)

Alternatief 3

EMPLOYEE(NAME,SSN,BDATE,ADDRESS,EMP_TYPE,
TSPEED,LIMIT,SPECIALITY)

Alternatief 3

EMPLOYEE(NAME,SSN,BDATE,ADDRESS,EMP_TYPE,
SEC_FLAG,TSPEED,
SALE_FLAG,LIMIT,
ENG_FLAG,SPECIALITY)

b. Geef het physieke model in SQL

Alternatief 1

```
CREATE TABLE EMPLOYEE(
        NAME        VARCHAR(15) NOT NULL,
        SSN         CHAR(2) NOT NULL,
        BDATE       DATE
        ADDRESS     VARCHAR(32),
        PRIMARY KEY SSN);

CREATE TABLE SECRETARY(
        SSN         CHAR(2) NOT NULL,
        TSPEED      INT,
        PRIMARY KEY SSN,
        FOREIGN KEY SSN REFERENCES
                ON DELETE CASCADE
                ON UPDATE CASCADE);

CREATE TABLE SALESMAN(
        SSN         CHAR(2) NOT NULL,
        LIMIT       INT,
        PRIMARY KEY SSN,
        FOREIGN KEY SSN REFERENCES
                ON DELETE CASCADE
                ON UPDATE CASCADE);

CREATE TABLE ENGINEER(
        SSN         CHAR(2) NOT NULL,
        SPECIALITY INT,
```

```
        PRIMARY KEY SSN,
        FOREIGN KEY SSN REFERENCES
                ON DELETE CASCADE
                ON UPDATE CASCADE);
```

3. Gegeven het volgende model, geef een aantal queries in Tupel-calculus, Domain-calculus en SQL.

(Zie ook de opgaven van 2 maart)

EMPLOYEE
(FNAME, MINIT, LNAME,SSN, BDATE, ADDRESS, SEX, SALARY, SUPERSSN, DNO)

DEPARTMENT
(DNAME, DNUMBER, MGRSSN, MGRSTARTDATE)

DEPT_LOCATIONS
(DNUMBER, DLOCATION)

PROJECT
(PNAME, PNUMBER, PLOCATION, DNUM)

WORKS_ON
(ESSN, PNO, HOURS)

DEPENDENT
(ESSN, DEPENDENT_NAME, SEX, BDATE, RELATIONSHIP)

a. Geef de namen en hun SSN van de werknemers die aan geen enkel project werken (in Tupel-calculus, Domain-calculus en SQL).

Tuple relational Calculus:

```
{ e.LNAME, e.FNAME |
        EMPLOYEE(e)
        AND NOT(EXISTS w) ( WORKS_ON(w) AND w.ESSN=e.SSN ) }
```

Domain relational Calculus (book notation):

```
{ qs | (EXISTS t) ( EMPLOYEE(qrstuvwxyz)
                        AND NOT(EXISTS a) (WORKS_ON(abc) AND a=t ) ) }
```

Domain relational Calculus (onze notatie met expliciete attribuutnamen):

```
{ (fname,lname) | (EXISTS ssn) (
                        EMPLOYEE(FNAME:fname, LNAME:lname, SSN:ssn)
                        AND NOT(EXISTS essn) (WORKS_ON(ESSN:essn)
```

```
                                                AND essn = ssn ) ) }
```

or more direct

```
{ (fname,lname) | (EXISTS ssn) (
                        EMPLOYEE(FNAME:fname, LNAME:lname, SSN:ssn)
                        AND NOT (WORKS_ON(ESSN:ssn) ) }
```

In SQL:

```
SELECT FNAME, LNAME FROM EMPLOYEE WHERE
        (NOT EXISTS WORKS_ON WHERE ESSN = SSN)
```

of met aliases

```
SELECT EMP.FNAME, EMP.LNAME FROM EMPLOYEE AS EMP WHERE
        (NOT EXISTS WORKS_ON AS WO WHERE WO.ESSN = EMP.SSN)
```

b. Geef de namen en addressen van de EMPLOYEEs die aan minstens 1 project in Houston werken, zonder dat hun eigen afdeling in Houston zit (in Tupel-calculus, Domain-calculus en SQL).

Tuple relational Calculus:

```
{ e.LNAME, e.FNAME, e.ADDRESS |
        EMPLOYEE(e)
        AND (EXISTS p) (EXISTS w) (
                WORKS_ON(w)
                AND PROJECT(p)
                AND e.SSN=w.ESSN
                AND w.PNO=p.PNUMBER
                AND p.PLOCATION='Houston'
                AND NOT(EXISTS l) ( DEPT_LOCATIONS(l)
                                        AND e.DNO=l.DNUMBER
                                        AND l.DLOCATION='Houston' ) ) }
```

Merk op dat de NOT voor de exists moet komen, en bij de locatie, dus niet

```
                AND (EXISTS l) ( DEPT_LOCATIONS(l)
                                        AND e.DNO=l.DNUMBER
                                        AND l.DLOCATION != 'Houston' ) ) }
```

Want dit zou ook afdelingen opleveren die 1 vestiging buiten Houston hebben.

Domain relational Calculus (book notation):

```
{ qsv | (EXISTS t) (EXISTS z) (
        EMPLOYEE(qrstuvwxyz)
        AND (EXISTS b) (EXISTS c) (EXISTS e) (EXISTS f) (
            WORKS_ON(efg)
            AND PROJECT(abcd)
            AND t=e
            AND f=b
            AND c='Houston'
            AND NOT(EXISTS h) NOT(EXISTS i) (
                DEPT_LOCATIONS(hi)
                AND z=h
                AND i='Houston' ) ) }
```

Domain relational Calculus (onze notatie met expliciete attribuutnamen):

```
{ (fname, lname, address) | (EXISTS ssn) (EXISTS D#) (
    EMPLOYEE(FNAME:fname, LNAME:lname, SSN:ssn, DNO:D#)
    AND (EXISTS P#) (EXISTS ploc) (EXISTS wo_ssn) (EXISTS wo_P#) (
            WORKS_ON(ESSN:wo_ssn, PNO:wo_P#)
            AND PROJECT(PNUMBER:P#, PLOCATION:ploc)
            AND ssn=wo_ssn
            AND wo_P#=P#
            AND ploc='Houston'
            AND NOT(EXISTS D2#) NOT(EXISTS dloc) (
                DEPT_LOCATIONS(D2#,dloc)
                AND D2#=D#
                AND dloc='Houston' ) ) }
```

In SQL

```
SELECT FNAME, LNAME, ADDRESS FROM EMPLOYEE WHERE
    EXISTS WORKS_ON, PROJECT WHERE
        SSN = ESSN
        AND PNO = PNUMBER
        AND PLOCATION = ‘Houston’
        AND NOT EXISTS DEPT_LOCATIONS WHERE
            DNUMBER = DNO
            AND DLOCATION = ‘Houston’
```

or with aliases

```
SELECT EMP.FNAME, EMP.LNAME, EMP.ADDRESS FROM
    EMPLOYEE as EMP WHERE
    EXISTS WORKS_ON AS WO, PROJECT AS P WHERE
        EMP.SSN = WO.ESSN
        AND WO.PNO = P.PNUMBER
```

```
        AND P.PLOCATION = 'Houston'
        AND NOT EXISTS DEPT_LOCATIONS AS LOC WHERE
                LOC.DNUMBER = EMP.DNO
                AND LOC.DLOCATION = 'Houston'
```

c. Geef voor elk project de projectnaam en het totaal aantal uren dat eraan gewerkt wordt (alleen in SQL)

```
SELECT PNAME, SUM (HOURS)
        FROM PROJECT, WORKS_ON
        WHERE PNUMBER = PNO
        GROUP BY PNAME
```

4. Gegeven het volgende model, geef een aantal SQL queries

(Zie de opgaven van 16 maart 2006)

BOOK
(BookId, Title, PublisherName)

PUBLISHER
(Name, Address, Phone)

BOOK_AUTHORS
(BookId, AuthorName)

BOOK_COPIES
(BookId, BranchId, No_of_Copies)

BOOK_LOANS
(BookId, BranchId, CardNo, DateOut, DueDate)

LIBRARY_BRANCH
(BranchId, BranchName, Address)

BORROWER
(CardNo, Name, Address, Phone)

a. Geef de namen van alle BORROWERs die momenteel geen boeken geleend hebben.

```
SELECT Name
        FROM BORROWER B
        WHERE NOT EXIST ( SELECT *
                                        FROM BOOK_LOANS L
                                        WHERE B.CardNo = L.CardNo )
```

or

```
SELECT Name
        FROM BORROWER
                EXCEPT
                (SELECT BORR.CardNo, BORR.NAME, BORR.Address, BORR.phone
                        FROM BORROWER AS BORR, BOOK_LOANS AS BL
                        WHERE BORR.CardNo = BL.CardNo)
```

Note in this last example that:
- The EXCEPT requires two tables with the same sequence of attributes.
- This formulation has the disadvantage that if the table definition of BORROWER changes (an attribute is added or deleted), then the innermost query has to be updated as well.

b. Geef van elke BRANCH de naam en het totaal aantal aldaar uitgeleende boeken.

```
SELECT BRANCH.BranchName, COUNT(*)
        FROM BOOK_LOANS LOAN,
                LIBRARY_BRANCH BRANCH
        WHERE LOAN.BranchId = BRANCH.BranchId
        GROUP BY BRANCH.BranchName
```

or

```
SELECT BranchName, COUNT(*)
        FROM BOOK_LOANS NATURAL JOIN LIBRARY_BRANCH
        GROUP BY BranchName
```

c. Geef de namen en hun CardNo's van alle BORROWERS die meer dan 5 boeken hebben uitgeleend

```
SELECT B.CardNo, B.Name, B.Address, COUNT(*)
        FROM BORROWER B,
                BOOK_LOANS L
        WHERE B.CardNo = L.CardNo
        GROUP BY B.CardNo
        HAVING COUNT(*) > 5
```

or

```
SELECT B.CardNo, B.Name, B.Address, COUNT(*) as count
        FROM BORROWER B,
                BOOK_LOANS L
        WHERE B.CardNo = L.CardNo
        GROUP BY B.CardNo
        HAVING count > 5
```

5. Herschrijven

a. Herschrijf de volgende SQL-query, zodat er geen nesting meer in voorkomt.

(Zie opgave 8.13.a van 16 maart 2006)

```
SELECT LNAME, FNAME
        FROM EMPLOYEE
        WHERE DNO=5
                AND SSN IN ( SELECT ESSN
                                FROM WORKS_ON
                                WHERE HOURS>10
                                        AND PNO IN ( SELECT PNUMBER
                                                        FROM PROJECT
                                                        WHERE PNAME = 'ProductX' ) )
```

Antwoord:

```
SELECT LNAME, FNAME
        FROM EMPLOYEE, WORKS_ON, PROJECT
        WHERE DNO=5
                AND SSN         = ESSN
                AND PNO         = PNUMBER
                AND PNAME       = 'ProductX'
                AND HOURS       > 10
```

b. Gegeven de bijgevoegde tabellen DRAAIT, FAN_VAN en LUISTERT_NAAR. Beschrijf de query "*Geef de personen die tenminste naar ieder radiostation luisteren waar Piet ook naar luistert",* eerst in Tupelcalculus en dan via herschrijving in SQL.

(Zie ook de slides van Hoofdstuk 8 en 9 van 14 en 21 maart, i.h.b. slide 60 voor het datamodel en 63-65 voor de uitwerking.)

We beginnen met het opschrijven van de query in de Tupelcalculus:

$$\{ (ln1.persoon) \mid LUISTERT\_NAAR(ln1) \wedge \forall lpiet ((LUISTERT\_NAAR(lpiet) \wedge lpiet.persoon = \text{“Piet”}) \Rightarrow \exists ln2 (LUISTERT\_NAAR(ln2) \wedge ln2.persoon = ln1.persoon \wedge ln2.station = lpiet.station)) \}$$

We herschrijven $\forall$ naar $\neg \exists \neg$ en $A \Rightarrow B$ naar $\neg A \vee B$

$$\{ (ln1.persoon) \mid LUISTERT\_NAAR(ln1) \wedge \neg\exists lpiet \neg (\neg((LUISTERT\_NAAR(lpiet) \wedge lpiet.persoon = \text{“Piet”})$$

$\vee \exists$ln2 (LUISTERT_NAAR(ln2)
$\wedge$ ln2.persoon = ln1.persoon
$\wedge$ ln2.station = lpiet.station)
)) }

We herschrijven $\neg$ (A $\vee$ B) naar $\neg$ A $\wedge \neg$ B

{ (ln1.persoon) | LUISTERT_NAAR(ln1) $\wedge$
$\neg\exists$ lpiet (LUISTERT_NAAR(lpiet) $\wedge$ lpiet.persoon = “Piet”)
$\wedge \neg \exists$ln2 (LUISTERT_NAAR(ln2)
$\wedge$ ln2.persoon = ln1.persoon
$\wedge$ ln2.station = lpiet.station)
)}

En dan herschrijven we $\neg\neg$ A naar A.

{ (ln1.persoon) | LUISTERT_NAAR(ln1) $\wedge$
$\neg\exists$ lpiet (LUISTERT_NAAR(lpiet) $\wedge$ lpiet.persoon = “Piet”
$\wedge \neg \exists$ln2 (LUISTERT_NAAR(ln2)
$\wedge$ ln2.persoon = ln1.persoon
$\wedge$ ln2.station = lpiet.station)
)}

Tot slot zetten we de query in de Tupelcalculus om naar SQL

```
SELECT DISTINCT L1.PERSOON
        FROM LUISTERT_NAAR AS L1
        WHERE NOT EXISTS
                (SELECT     LPIET.STATION
                 FROM       LUISTERT_NAAR AS LPIET
                 WHERE      LPIET.PERSOON = “PIET”
                        AND NOT EXISTS
                        (SELECT     *
                         FROM       LUISTERT_NAAR AS L2
                         WHERE      L2.PERSOON = L1.PERSOON
                                    AND L2.STATION = LPIET.STATION));
```