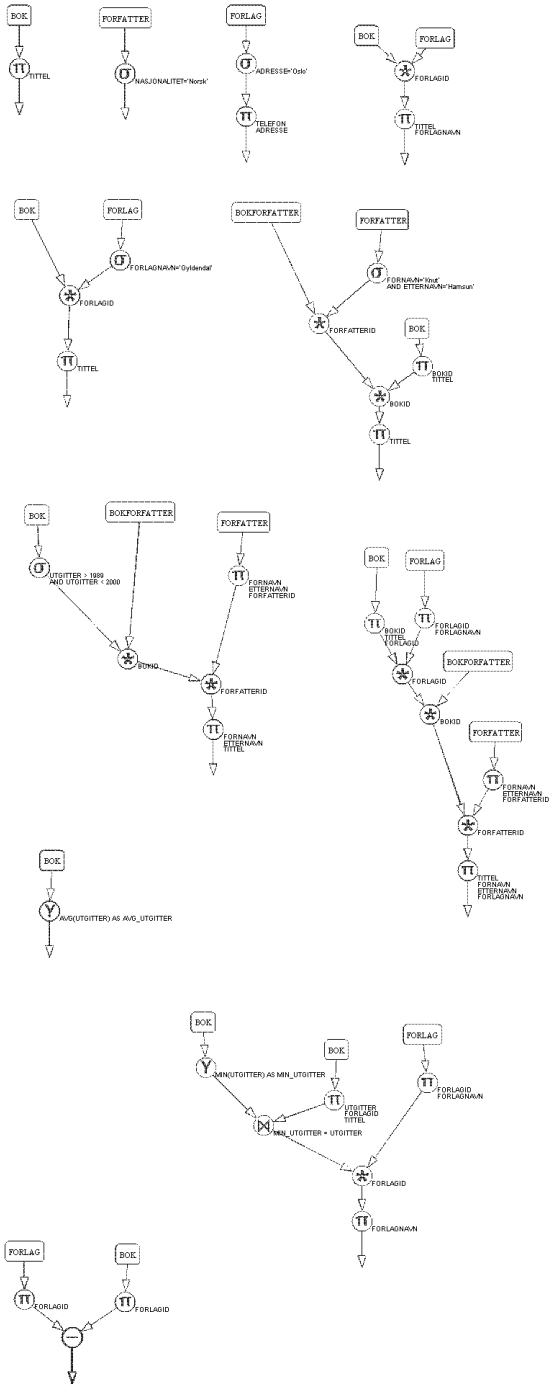
# **TDT4145**: Data Modelling and Database Systems

# Exercise 3: “Relational algebra and SQL”

David Rozas Domingo
Miguel Bono Tur

# Task1


BOK
TITTEL
FORFATTER
NASJONALITET='Norsk'
FORLAG
ADRESSE='Oslo'
TELEFON
ADRESSE
FORLAGID
FORLAGNAVN
FORLAGNAVN='Gyldendal'
BOKFORFATTER
FORNAVN='Knut'
AND ETTERNAVN='Hamsun'
FORFATTERID
BOKID
UTGITTER > 1989
AND UTGITTER < 2000
FORNAVN
ETTERNAVN
FORFATTERID
AVG(UTGITTER) AS AVG_UTGITTER
MIN(UTGITTER) AS MIN_UTGITTER
UTGITTER
MIN_UTGITTER = UTGITTER

## Task 2

a)

```
CREATE TABLE Customer(customerId INTEGER,
                      name       CHAR(30),
                      creditLimit FLOAT,
                      zipCode    INTEGER,
                      PRIMARY KEY (customerId),
                      FOREIGN KEY (zipCode) REFERENCES PostCode)

CREATE TABLE PostCode (zipCode   INTEGER,
                      place      CHAR(20),
                      PRIMARY KEY (zipCode)

CREATE TABLE Item     (itemId    INTEGER,
                      name       CHAR(20),
                      quantity   INTEGER,
                      price      FLOAT
                      PRIMARY KEY (itemId))

CREATE TABLE Order    (itemId    INTEGER,
                      customerId INTEGER,
                      quantity   INTEGER,
                      PRIMARY KEY (itemId, customerId),
                      FOREIGN KEY (itemId) REFERENCES Item,
                      FOREING KEY (customerId) REFERENCES Customer)
```

b)

For performing the changes in Order if a Customer or Item is deleted or updated, we have to change the definition of Order in this way:

```
CREATE TABLE Order    (itemId    INTEGER,
                      customerId INTEGER,
                      quantity   INTEGER,
                      PRIMARY KEY (itemId, customerId),
                      FOREIGN KEY (itemId) REFERENCES Item,
                      FOREING KEY (customerId) REFERENCES Customer
                            ON DELETE CASCADE
                            ON UPDATE CASCADE)
```

Regarding to PostCode, it seems natural that if we delete an instance of PostCode, instead of deleting the customer, his attribute ZipCode which references to PostCode takes null as value. In the case of Update, the natural would be to update also the ZipCode in Customer. So the necessary changes to perform this behaviour are:

```
CREATE TABLE Customer (customerId      INTEGER,
                      name       CHAR(30),
                      creditLimit FLOAT,
```

```
zipCode       INTEGER,
PRIMARY KEY (customerId),
FOREIGN KEY (zipCode) REFERENCES PostCode,
ON UPDATE CASCADE,
ON DELETE NULL)
```

c)

To avoid that a customer order total does not exceed the credit limit we can create an assertion:

```
CREATE ASSERTION CreditLimit
CHECK(
SELECT (SUM(I.Price)
FROM Customer C, Order O, Item, I
WHERE C.CustomerId = O.CustomerId AND O.ItemId = I.ItemID) <
                                                    (SELECT C2.CreditLimit
                                                     FROM Customers C2") )
```

## Task 3

a) SELECT Tittel FROM Bok.

b) SELECT * FROM Forfatter
WHERE Nasjonalitet = Norsk

c) SELECT Forlagnavn, Telefon FROM Forlag
WHERE Adresse = Oslo
ORDER BY Forlagnavn ASC

d) SELECT a.Tittel, b.Forlagnavn FROM Bok a, Forlag b
WHERE a.Forlagid = b.Forlagid

e) SELECT a.Tittel,a.Utgittar FROM,.Bok a, Forfatter b, Bokforfatter c
WHERE c.Forfatterid = b.Forfaterid AND b.Fornavn = Knut
AND b.Etternavn = Hamsun AND a.Bokid = c.Bokid

f) SELECT Fornavn, Etternavn, Fodear FROM Forfatter
WHERE Etternavn > Gz AND Etternavn < Ia

g) SELECT COUNT(*) FROM Forlag

h) SELECT a.Tittel, b.Fornavn, b.Etternavn, c.Forlagnavn
FROM Bok a, Forfatter b, Forlag c, Bokforfatter d
WHERE b.Nasjonalitet = Britisk AND d.Forfatterid = b.Forfatterid
AND d.Bokid = a.Bokid AND a.Forlagid = a.c.Forlagid

i) SELECT COUNT (ab.Bokid) AS total, a.Forfatterid
FROM Bokforfatter ab, Forfatter a
WHERE a.Forfatterid = ab.Forfatterid
GROUP BY a.Forfatterid
ORDER BY total DESC

j) SELECT Tittel, Utgittar FROM Bok
WHERE Utgittar = min(Utgittar)

k) SELECT COUNT(b.bokid)n AS total, p.Forlagid
FROM Bok b, Forlag P
WHERE b.Forlagid = p.Forlagid
GROUP BY p. Forlagid

HAVING total > 2

l) SELECT a.Forlagid FROM Forlag a
WHERE a.Forlagid NOT IN ( SELECT b.Forlagid FROM Bok b)

## Task 4

a)

View mechanism provides:
- Support for logical data independence, which can be used to define relations in the external schema that mask changes in the conceptual schema.
- Support for security, because we can define views that give a group of users access to just the information they are allow to see.

But some problems can arise on updating, because we can have side effects. So we have to handle the updating through views carefully, or even disallow it.

b)

```
CREATE VIEW (ProjectName, DepartmentDependency, NumberOfEmployees, TotalHours)
AS
SELECT P.name, DL.Dlocation, COUNT (E.ssn), SUM(W.hours)
FROM Employees E, Project P, Works_on W, Department D, Dept_locations DL
WHERE E.ssn=W.ssn AND P.Pnumber = W.Pno
        AND P.Dno = D.Dnumber AND DL.Dnumber = D.Dnumber
GROUP BY P.Dno )
```

c)

1) It is valid, it returns all the fields of the summary of the the department, who has been performed by grouping the employees by his department number.
2) It is valid, it returns the number of department and the number of employees where the total salary of all the employees in this department exceed 10000.
3) It is not valid, you cannot modify data in views that use GROUP BY.
4) It is not valid (same reason as 3)

## Task 5

The index are for find more quickly what we want to find. They streamline the search.

ADVANTAGES:

- They avoid a completely scanning of the table. When we have a huge amount of data, the improvement is very high.
- How we avoid the completely scanning of the table, we avoid too the CPU overload, hard disk overload and concurrency.
- They avoid the sequential lectures.
- They allow more speed in the queries.
- Get an advantage when there aren  t duplicate data.

DISADVANTAGES:

- The index is a disadvantage when there are operations that can modify the table (INSERT, DELETE, UPDATE) because the index will be change when a column is modified.
- When the tables are so small, the improvement time is like zero.
- Is not advisable when we want that the table where is applied returns high amount of data.
- The index occupy space, sometimes more space than de data.

## Task 6

a)

```
SELECT *
FROM Supplier S
WHERE S.status>15
```

b)

```
SELECT S.sname, S.city
FROM Supplier S, Part P, SuppliesPart SP
WHERE S.sno = SP.sno AND P.pno = SP.pno AND P.pname = “Screw”
```

c)

```
SELECT P.pno, P.name
FROM Supplier S, Part P, SuppliesPart SP
WHERE S.sno = SP.sno AND P.pno = SP.pno
                AND   (SELECT COUNT(S2.sno)
                        FROM Suppliers S2, Part P2, SuppliesPart SP2
                        WHERE S2.sno = SP2.sno
                        AND P2.pno = SP2.pno
                        GROUP BY P2.pno)>=2
```

d)

```
SELECT COUNT(S.sno)
FROM Suppliers S
```

e)

```
SELECT DISTINCT(S.city)
FROM Supplier S, Part P, SuppliesPart SP
WHERE S.sno = SP.sno AND P.pno = SP.pno AND P.weight>10.0
```

f)

```
SELECT S.sname, S.city
FROM Supplier S, Part P, SuppliesPart SP
WHERE S.sno = SP.sno AND P.pno = SP.pno
        AND S.sno NOT IN (
                        SELECT S2.sno
                        FROM Suppliers S2, Part P2, SuppliesPart SP2
                        WHERE S2.sno = SP2.sno AND P2.pno = SP2.pno
                                AND P2.pname = “Screws”)
```