Second set of mandatory exercises in INF3100/INF4100 Spring 2007

Formalities

All students are to hand in an individual assignment. If two students want to hand in an assignment together, they have to apply to the group teacher and get an acception for this in advance. Students following the course Inf3190/4190 and that have a home exam with deadline May 11 may apply to the group teacher for a week's postponement.

The assignment is to be delivered by email to the group teacher.

The email is to have the following subject field:
Subject: Oblig 2 inf3100 (<username student >)

Notice: Assignments delivered after the deadline will be rejected. You must answer all questions in order to get your assignment approved.

Students that have had their assignment accepted and still choose to withdraw from the exam, have to deliver a paper copy of the assignment to their group teacher to get a written statement from the group teacher that it has been accepted. This only concerns students that withdraw before the 14-day limit.

This mandatory exercise is due: Friday 11th of May at 12:00 am.

It is a good idea to start working on the assginment as early as possible.

Exercise 1

Given the following data structure for a relational database (primary keys are printed in **bold**, other candidate keys are printed in *italics*, foreign keys are given by the attribute names):

PERSON (**Ssn**, Surname, Forename, Address)
MARRIAGE (**Ssn-w**, *Ssn-m*, ***Date***, Surname-w, Surname -m)
NAME-CHANGE (**Ssn**, **Date**, Surname, Forename)

The current name is stored in PERSON. Hence, in NAME-CHANGE we store the name a person had immediately prior to changing his/her name. The names in MARRIAGE are the surnames after the wedding.

Solve the following tasks both using relational algebra and SQL:

1. Find name and address of all women which in the years 1968-72 on their wedding day changed their last name to a name different from their groom's last name.

2. Find the name and address of all women who by a wedding in 2003 have swapped surname with their bridegroom. Wedded couples having the same surname should not be listed.

Exercise 2

Using Exercise 1 as basis, make an OO-data structure. It is obvious that the table PERSON have to be replaced with a class Person. It is also fairly obvious that the table NAME-CHANGE is to be replaced with a local list in each person object. (We could very well have chosen a set or bag instead of a list, but the name changes for a person are ordered in time.)

For the table MARRIAGE, it is not such a clear cut case. We can choose to replace it with a separate class Marriage, or to replace it with local data in each person object.

Exercise 2 a

Write a complete ODL definition for each of the alternatives above (having respectively two and one classes). Do not forget the methods, and remember that it is only the method signature that is part of ODL!

Exercise 2 b

Use OQL to answer both questions in Exercise 1 for each of the ODL definitions in 2 a.

Exercise 3

Assume that we have a storage system using (modified) Seagate Cheetah X15 disks. Some of the specifications are as follows:

```
Disk platters:                      8
Tracks:                             20000
Number of sectors per track:        700 (not zoned disk)
Bytes per sector:                   512
Bytes per gap/checksum/...:         64
Average seek time:                  3.6 ms
Track-to-track seek:                0.3 ms
Rotation speed:                     15.000 RPM
```

Further, assume that the storage system uses blocks of 4 KB, and that we may neglect delays like processing queries, transfer over buses, waiting for turn, etc.

Exercise 3 a

1. What is the total capacity of the disk if the platters are used on both sides?
2. What is the average rotation delay?
3. What is the average access time for an arbitrary block?

We create a customer database on the storage system above. Each block has a header of 12 B, and each record has a header of 20 B. We have 1.000.000 customers, where each record has the following fields (number of bytes):

```
navn    (30)
fnr     (9)
adresse (30)
tlf     (8)
email   (30)
```

Exercise 3 b

1. What is the block factor?
2. How much time does it take to read the whole table (continuously) if we assume unspanned storage and arbitrary placing of blocks on the disk?
3. Instead of arbitrary placement, we try continuous placing on the disk – we fill one track first, if full, we take the next track in the same cylinder, if full again, the next cylinder. Assume that the table starts in the first sector in the first track in a cylinder. How much time does it take now to read the whole table (continuously), if we still assume unspanned storage and that change of disk head does not take any time (we can start reading directly)?

Assume again that the blocks are arbitrarily placed on the disk, and that the time it takes to process a block in memory is close to zero.

Exercise 3 c

1. What is the average time for accessing an arbitrary record without indexes?
2. We add a dense index with binary search. Unique search key is fnr, and the pointer takes 8 B. How much space does the index take if each block in average is 80% full?
3. What is the average time (assume a maximum number of lookups in the index) needed to access an arbitrary record using an index like this?
4. Now, we replace our index with a B^+-tree. How much space does the B^+-tree need if each node uses a disk block of its own which in average is 80 % full?
5. What is the time now needed to access an arbitrary record?

Exercise 4

Consider the following schedule where we use binary locks:

	Trans T1	Trans T2	Trans T3
1	Lock(X)		
2	Read(X)		
3		Lock(Y)	
4		Read(Y)	
5	X=f(X)		
6			Lock(Z)
7			Read(Z)
8		Y=g(Y)	
9		Write(Y)	
10		Unlock(Y)	
11			Z=h(Z)
12	Write(X)		
13	Unlock(X)		
14			Write(Z)
15			Unlock(Z)
16			Lock(Y)
17			Read(Y)
18		Lock(X)	
19		Read(X)	
20	Lock(Z)		
21	Read(Z)		
22		X=g(X)	
23			Y=h(Y)
24	Z=f(Z)		
25		Write(X)	
26		Unlock(X)	
27			Write(Y)
28			Unlock(Y)
29	Write(Z)		
30	Unlock(Z)		

Exercise 4 a

Draw the precedence graph (serializability graph), and decide if the plan is conflict-serializable.

Exercise 4 b

Explain/show why the plan does not obey the two-phase locking protocol (2PL).

Exercise 4 c

Make a schedule for the three transactions that satisfy 2PL, draw the precedence graph for the new schedule, and show that it is conflict-serializable.

Exercise 5

Solve exercises 2, 3, and 5 from Exam in INF3100 given spring 2004 (you will find a link to the exam on the INF3100 semester page).

Exercise 6

Solve the Exam in INF3100 given spring 2006 (you will find a link to the exam on the INF3100 semester page).

End of mandatory exercise set 2.

UNIVERSITY OF OSLO

Faculty for Mathematics and Natural Sciences

Exam in : INF3100/INF4100 — Database Systems
Date of Examination : Tuesday, June 8, 2004
Examination hours : 09.00 am – 12.00 am
This examination set consists of 5 pages
Appendices : None
Permitted aids : Calculator

Make sure that your copy of the examination set is complete before attempting to answer anything.

There are six tasks, all having the same weight

Thus you should spend approximately 30 minutes solving each task

Task 1 SQL and relational algebra

Given the following data structure for a relational database (Primary keys are printed in **bold** face, other candidate keyes are printed in *italics*, foreign keys are given by the attribute names):

PERSON (**Ssn**, Surname, Forename, Address)
MARRIAGE (**Ssn-w**, *Ssn-m*, ***Date***, Surname-w, Surname -m)
NAME-CHANGE (**Ssn**, **Date**, Surname, Forename)

The current name is stored in PERSON. Hence, in NAME-CHANGE we store the name a person had immediately prior to changing his/her name. The names in MARRIAGE are the surnames after the wedding. Date is a text string with format `'2004-06-08'`, the order is year, month, day.

Solve the following task both using relational algebra and SQL:

> Find the name and address of all women who by a wedding in 2003 have swapped surname with their bridegroom. Wedded couples having the same surname should not be listed.

Task 2 Normalization

A small youth club offers their members (all have telephones at home and cellular phones) a number of courses. To keep track of who participates in what, they have made a table (in Excel) with the following 8 columns:

MembershipNo, Name, Address, HomePhone, CellularPhone, CourseCode, CourseName and Instructor. We have the following functional dependencies:

MembershipNo determines Name, Address, HomePhone, and CellularPhone
CellularPhone determines MembershipNo, Name, Address, and HomePhone
Address and HomePhone determine each other
CourseCode determines CourseName and Instructor

Task 2 a Which candidate keys and which normal form does this table have?

Task 2 b Normalize the table to BCNF

Task 3 Parsing, query plan, and optimization

I this task you shall show your understanding of how to parse a query, how to make a logical query plan, how to optimize such a query plan. We shal use the following relations in this task:

```
CUSTOMER(custID, sex, forename, surname, ssn)
ACCOUNT(accountNO, accounttype, interest,
openingdate)
OWNERSHIP(ownershipID, custID, ownership, accountNO)
```

Note that primary keys are in a **double-underscored bold** font. Other candidate keys are just underscored.

The relation `ACCOUNT` contains bank accounts of several types and the dates these accounts were opened. There may be several customers connected to each account. This informaton is stored in `OWNERSHIP` where the field `ownership` is either `'O'`, meanig this customer is the owner, or `'D'`, meanig this customer may dispose of the account. The owner is the one who opened the account. The attribute `openingdate` in `ACCOUNT` is a standard SQL `DATE`, i.e. a text string with format `'2003-06-08'`, the order is year, month, day. Sex in `CUSTOMER` may be `'F'` or `'M'`. The attribute `accounttype` in `ACCOUNT` may be `'C'` for current account, `'S'` for savings account, or `'M'` for mortage account.

We shall use the following query to find the forename, surname and ssn for female customers that have opened a savings account in 2003:

Task 3 continues at the next page

```
SELECT   CUSTOMER.forename, CUSTOMER.surname,
         CUSTOMER.ssn
FROM     CUSTOMER, ACCOUNT, OWNERSHIP
WHERE    CUSTOMER.custID = OWNERSHIP.custID
     AND
         ACCOUNT.accountNO = OWNERSHIP.accountNO
         AND
         ACCOUNT.openingdate LIKE '2003%'      AND
         ACCOUNT.accounttype = 'S'             AND
         CUSTOMER.sex = 'F'                    AND
         OWNERSHIP.ownership = 'O'
```

The data base has clustered indices for its primary keys. In addition there are indices on the attribute `openingdate` in `ACCOUNT` and on the attributes `custID` and `accountNO` in `OWNERSHIP`.

Task 3 a Parsing

i) Use the simple grammar on page 5 to make a parse tree for the above query.

ii) What are the main task(s) for the preprosessor?

Task 3 b Logical query plan

Convert the parse tree in task 3 a above to a logical query plan in relational algebra (draw the expression tree). NB! This task should be solved without any optimization Optimization belongs to the next task!

Task 3 c Optimization

i) Which rules are often used (usually give a high performance gain) to optimize logical query plans?

ii) Optimize the logical query plan in task 3 b above (draw the new expression tree).

Task 4 Indices (In US English: Indexes)

Task 4 a

Draw and explain what dense, sparse, and multi-level indices are.

Task 4 b

Explain briefly the advantages and disadvantages of these tree types of indices, in which settings they are recommended, and why.

End of task 4

Task 5 and 6 are to be found on the next page

Task 5 Logging

Task 5 a

Describe the two main types of loggs: optimistic (Undo) and pessimistic (Redo)

Task 5 b

Describe what a checkpoint is. Put the main emphasis on the usage of the logg.

Task 6 Transaction handling

The two serializability concepts covered in this course, are based on respectively conflict equivalence and view equivalence of execution plans.

Task 6 a

Define conflict equivalence and view equivalence.

Task 6 b

Are there more conflict serializable than view serializable execution plans?

What additional rule must we have to make conflict equivalence and view equivalence the same?

Prove that this additional rule really ensures that conflict equivalence and view equivalence are the same.

End of examination tasks

Naci Akkøk and Ragnar Normann

Appendix to task 3 — A grammar for parsing queries

```
<query>      ::= <SFW>

<SFW>        ::= SELECT <selList>
                 FROM <fromList>
                 WHERE <condition>

<selList>    ::= <attribute>, <selList> |
                 <attribute>

<fromList>   ::= <relation>, <fromList> |
                 <relation>

<condition> ::= <condition> AND <condition> |
                 <attribute> = <attribute> |
                <attribute> = <pattern> |
                <attribute> LIKE <pattern>
```

Elementary syntactic categories as <attribute>, <relation>, and <pattern> have no rules. They are translated respectively into the name of the attribute, the name of the relation, and a string within double-quotes.

UNIVERSITY OF OSLO

Faculty of mathematics and natural sciences

Examination in	INF3100/INF4100 — Database systems
Day of examination:	June 9, 2006
Examination hours:	14.30 – 17.30
This problem set consists of 3 pages.	
Appendices:	None
Permitted aids:	Calculator and dictionary

Please make sure that your copy of the problem set is complete before you attempt to answer anything.

Read the exercises closely, and good luck!

Problem 1 Relational database languages (60%)

In this exercise you shall make use of four relations:

```
Film(filmID,title,productionyear)
Person(personID,gender,firstname,surname)
Participation(personID,filmID,participationname)
ParticipationRange(participationname)
```

Attributes filmID and personID are primary keys in Film and Person respectively. In Participation all attributes are members of the primary key. Null values are not allowed in any attribute, and 'F' and 'M' (for 'Female' and 'Male') are the only allowed values of gender. Foreign keys are indicated by having the same name as the corresponding primary key. The relation ParticipationRange has only five rows: 'cast', 'director', 'music', 'producer', and 'writing credits'.

1a (10%)

Use SQL to define the two tables Person and Participation. Don't forget the integrity rules!

1b (10%)

Express in relational algebra that participationname in Participation is foreign key to ParticipationRange.

(Continued on page 2.)

1c (10%)

Assume that no film has more than one producer, and use relational algebra to find the title of all films with a female producer and at least two directors.

1d (15%)

Use SQL to find all films directed by Charlie Chaplin. The resulting table shall have five attributes: The title of the film, the production year, the number of female actors, the number of male actors, and the total number of actors. Sort the table wrt. the last attribute, with the largest value first.

Hint: You may use views.

1e (15%)

Make one SQL query that finds out in what year the largest number of films were produced, and how may films were produced that year.

To obtain full score, the query should not make use of views.

Problem 2 Strategy for disk storage (20%)

2a (5%)

Explain briefly the key principles of the RAID 5 strategy and explain what distinguishes RAID 5 from RAID 4.

2b (15%)

Give a more thorough description of RAID 5 by describing what calculations and writes have to be done when a block of data is to be written to disk, and what needs to be done when a disk crashes and must be replaced.

(Continued on page 3.)

Problem 3 Strict 2PL (20%)

Recall that in strict 2PL (strict two-phase locking), in contrast to ordinary 2PL, a transaction does not release any lock before it commits (or aborts).

3a (10%)

We are performing the following two transactions:

$$t_1 = r_1(x)r_1(z)w_1(z) \quad \text{and} \quad t_2 = r_2(x)r_2(z)w_2(x)r_2(y)w_2(y)$$

Make a conflict serializable execution plan for executing t_1 and t_2 that cannot be generated by a strict 2PL-scheduler.

3b (10%)

Does there exist a non-serial execution of t_1 and t_2 that can be generated by a strict 2PL-scheduler? Give the grounds for your answer.