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.

End of task 1

Task 2 is to be found on the next page

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.