EXAM
Databases (DIT620/TDA355/TDA356/TDA357)

DAY: 14 Mar 2009 TIME: 8:30 – 12:30 PLACE: Hörsalsvägen

Responsible:	Niklas Broberg, Computing Science mobil 0706 49 35 46
Results:	Will be published on the course web page after the exam
Extra aid:	A single, hand-written A4 paper. It is legal to write on both sides. This paper should be handed in with the exam.
Grade intervals:	**U**: 0 – 23p, **3**: 24 – 35p, **4**: 36 – 47p, **5**: 48 – 60p, **G**: 24 – 41p, **VG**: 42 – 60p, **Max.** 60p.

IMPORTANT

The final score on this exam is computed in a non-standard way. The exam is divided into 7 blocks, numbered 1 through 7, and each block consists of 2 or 3 levels, named A, B, and optionally C. A level can contain any number of subproblems numbered using i, ii and so on. In the final score you can only count **ONE** level from each block. For example: if you attempt to solve the problems on all three levels in block 4 and manage to obtain 4 points for 4A (block 4, level A), 1 point for 4B and 8 points for 4C, only problem 4C (where you got your highest score) will count towards your final result, so your score for block 4 will be 8 points.

The score for each problem depends on how difficult it is (more points for harder problems) and how important I think it is (more points for more important problems). It does *not* depend on how much work it takes to answer the problem. There could very well be a 12 point problem that takes 15 seconds to answer (given that you know the right answer, of course).

The problems in each block are ordered by increasing difficulty. Hence the A problems are easy, but aim to cover the full basics of its area. The B and C level problems are more difficult, and aim to test your knowledge of the areas beyond the mere basics. If you only solve A problems your maximum score is 36 points, and if you only solve the B problems where there are also C problems it is 40 points.

Please observe the following:

- Answers can be given in Swedish or English
- Use page numbering on your pages
- Start every assignment on a fresh page
- Write clearly; unreadable = wrong!
- Fewer points are given for unnecessarily complicated solutions
- Indicate clearly if you make assumptions that are not given in the assignment

Good advice

- Most problems have been designed to give short answers. Few problem should require more than one page to answer.
- There are more problems than you are likely to solve in 4 hours. This means that you have to think about which problems you attempt to solve. If you try solve the problems in the order they are given, **you are likely to fail the exam**!

Good Luck!

Block 1 - Entity-Relationship Diagrams **max 12p**

1A (8p)

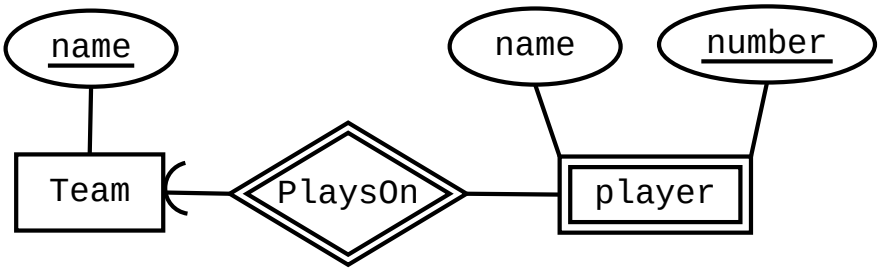
(i) (6p)

A new cinema chain is starting in Göteborg. They want a booking system for their cinemas. Here is a description of this domain:
A cinema has an address and name. It can show several movies, which can have different prices for customers to view. The same movie can cost differently at different cinemas. Each cinema can contain several halls, distinguished by hall numbers. Each hall can show a number of movies, at different times, and we need to know the start and end times of each show. The same movie can be shown in different halls in the same cinema, and several times in the same hall. A hall contains a number of seats, which can be occupied or not during each show. For each seat the location in the hall has to be known, i.e. the row number and seat number. For each movie one needs to know the title, type of movie, length, the legal age for seeing it (in general there will be more information about a movie, but what is given here is sufficient for this question).

Your task is to draw an ER diagram that correctly models this domain and its constraints. To obtain full points for the above domain you should include both weak entities and relation attributes.

(ii) (2p)

The E/R diagram below is part of an E/R diagram needed for keeping track of a football game.



Translate this ER diagram into a set of relations. Mark keys and references clearly in your answer.

1B (12p)

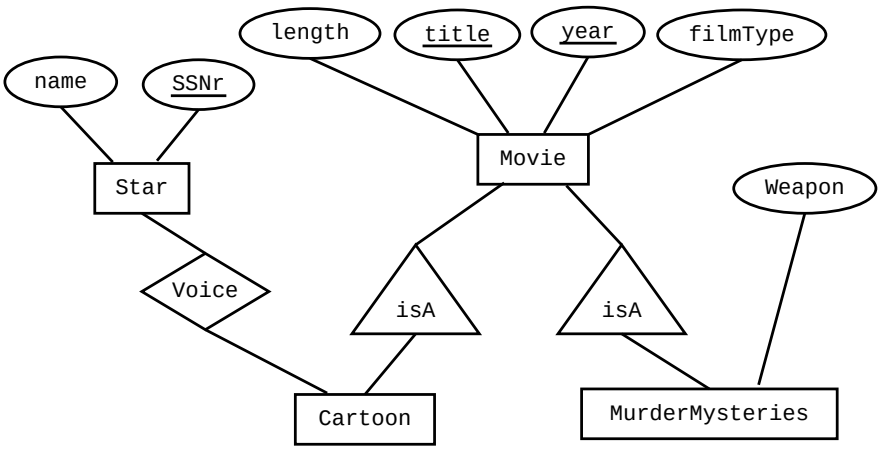
(i) (6p)

Below are schemas which could be part of an airline database:

Airport(<u>*code*</u>, *name*, *city*, *state*)
FlightLeg(<u>*legNo*</u>, <u>*number*</u>)
    number → *Flight.number*
Flight(<u>*number*</u>, *airline*, *weekdays*)
DepartureAirport(<u>*airportCode*</u>, <u>*legNo*</u>, <u>*number*</u>, *scheduledDepTime*)
    airportCode → *Airport.code*
    (*legNo*, *number*) → *FlightLeg.*(*legNo*, *number*)
ArrivalAirport(<u>*airportCode*</u>, <u>*legNo*</u>, <u>*number*</u>, *scheduledArrTime*)
    airportCode → *Airport.code*
    (*legNo*, *number*) → *FlightLeg.*(*legNo*, *number*)

All of this should be self-explanatory (or part of the problem), but if there is something that you don't understand, don't hesitate to ask.
Reconstruct the ER diagram that led to these relations and constraints.

(ii) (6p)



Translate this ER diagram into a set of relations using

- E/R-style conversion
- An object-oriented approach
- null values to combine relations

Furthermore, give the advantages and disadvantages for each approach. Mark keys and references clearly in your answer.

Block 2 - Dependencies and Normal Forms **max 12p**

2A (8p)

A boat rental company specializes in renting motor boats to people for one-day trips. They intend to use a database to store information about boats, customers and rentals.

They offer a wide variety of boats for rent, and to simplify for their customers they have divided their boats into various classes depending on size and horse power. These classes are denoted with letters A, B and so on, where A is the smallest class.

Boats are identified by a unique identification number. They are categorized by their model and year of make, and all boats of the same model made the same year have the same size and horse power.

Customers should be stored in the database using a unique identification number. For each customer the name and a contact phone number should also be stored. Customers may rent boats on a per day basis, where for each rental the class should be stored to determine the price, and the phone number of a customer for easy contact if something goes wrong.

They propose the following relations, and have asked you to verify their design:

Classes(<u>*letter*</u>)
Boats(<u>*regNr*</u>, *model*, *yearOfMake*, *size*, *horsePower*, *class*)
    class $\rightarrow$ *Classes.letter*
Customers(<u>*idNr*</u>, *name*, *phoneNumber*)
Rentals(<u>*boat*</u>, <u>*date*</u>, *class*, *customer*, *phoneNumber*)
    boat $\rightarrow$ *Boats.regNr*
    class $\rightarrow$ *Classes.letter*
    customer $\rightarrow$ *Customers.idNr*

This schema is not fully normalized, and thus suffers from a number of problems. It is your task to solve these by normalization of the schema.

(i) (4p)
For the given domain, identify all functional dependencies that you expect to hold.

(ii) (1p)
With the dependencies you have found, identify all BCNF violations in the relations of the database. For each violation, also specify what kinds of problems that could arise if it was not resolved.

(iii) (3p)
Do a complete normalization of the schema, so that all relations are in BCNF.

2B (12p)

A lone enthusiast wants to create an online database for death metal bands and records. While being very knowledgeable in the domain of death metal, he knows less about databases, so he asks your help.

The database should contain information about bands and their members, as well as their records. Bands can (and do) obviously have many members. But also, since bands come and go throughout history, and old band members form new bands, a particular person could be the member of more than one band. For bands, the name is stored together with the year that the band started playing. For members, name (assumed unique here) is stored together with year of birth, and also what instrument(s) the member plays (possibly more than one). It is assumed that a person who plays in several bands plays the same instrument(s) in all of them. Each album is recorded by a single band, and has a globally unique id number together with a title.

The following relation sums up all the attributes that should be stored in the database:

DeathMetal(*bandName*, *playedSince*, *recordId*, *title*, *memberName*, *yearOfBirth*, *instrument*)

Your task is to use normalization techniques to find a suitable schema for this database.

(i) (8p)
Find all dependencies and independencies (aka multi-valued dependencies) that you expect should hold for this domain given the domain description above.

(ii) (4p)
Do a complete decomposition of *DeathMetal* so that the resulting schema fulfills 4NF.

Block 3 - SQL Data Definition Language **max 8p**

The domain for this block, and for several following blocks as well, is that of a database for a small banking enterprise. The database models customer information as well as account information. The bank is organized into local branches, and all accounts belong to a particular branch. There are two kinds of accounts, savings and credit accounts. Savings accounts must have a positive balance, whereas credits accounts may have a negative balance down to some lower limit. The credit rating of a customer is given as a number, denoting how much credit they may at most have. The database also has a table with information regarding interest rates.

You are given the following schema of their intended database:

$Customers(\underline{ssNr}, name, adress, creditRating)$
$Branches(\underline{branchCode}, name, adress)$
$Accounts(\underline{branch}, \underline{accountNr}, customer)$
$\quad branch \rightarrow Branches.branchCode$
$\quad customer \rightarrow Customers.ssNr$
$SavingsAccounts(\underline{branch}, \underline{account}, balance)$
$\quad (branch, account) \rightarrow Accounts.(branch, accountNr)$
$\quad balance > 0$
$CreditAccounts(\underline{branch}, \underline{account}, balance, limit)$
$\quad (branch, account) \rightarrow Accounts.(branch, accountNr)$
$\quad balance > limit$
$\quad limit < 0$
$InterestRates(\underline{threshold}, rate)$

3A (4p)

Write SQL DDL code that correctly implements these relations as tables in a relational DBMS. Make sure that you implement all given constraints correctly. Do not spend too much time on deciding what types to use for the various columns. We will accept any types that are not obviously wrong. Don't forget to implement all specified constraints, including checks.

3B (8p)

The bank wants to implement a special scheme for how to handle withdrawal of money from credit accounts. The idea is that if a withdrawal would cause the balance to fall below the current limit, instead of rejecting the withdrawal the system should check if the credit can be extended. The way this should be done is to calculate the total amount of credit the customer has been granted on all their credit accounts (sum of all limit values), and compare this value to the customer's credit rating. If there is enough credit left to give, the credit of the account should be extended instead of rejecting the withdrawal.

Your task is to sketch what elements - views, assertions and triggers - the database needs to be extended with to handle this scheme automatically. You do not need to (read should not!) write the full code for these elements, but sketch enough of them so that it is clear that they would implement the scheme as intended.

Block 4 - SQL Queries **max 8p**

Use the relations for the bank enterprise from the previous block when answering the following problems.

4A (4p)

(i) (2p)
Write an SQL query that lists all accounts - both savings and credit - with a balance of one million or above.

(ii) (2p)
Write a query that lists all customers along with the number of accounts they hold in the bank.

4B (6p)

Write a query that for each account lists the current interest rate. The interest rate depends on the balance - it is the rate associated with the highest threshold in the table over interest rates that the balance *exceeds*. For example, if the balance is between 0 and 100000, the interest rate would be the rate associated with the threshold 0.

4C (8p)

Write a query that finds the branch(es) that has the richest customers on average, i.e. the average of the sum of all balances for a customer. A customer that has accounts in more than one branch should be counted for all of them for the purpose of finding the averages. If more than one branch has the same highest average, list all of those that do.

Block 5 - Relational Algebra **max 6p**

Use the relations for the banking enterprise from the previous blocks when answering the following problems.

5A (4p)

(i) (2p)
What does the following relational-algebraic expression compute (answer in plain text):

$$\tau_{-nr}(\gamma_{customer,COUNT(*)\rightarrow nr}(\sigma_{S.account=A.accountNr\, S.branch=A.branch}(\rho_S(SavingsAccounts) \times \rho_A(Accounts)))) \quad (1)$$

(ii) (2p)
The following relational algebra expression returns all customers that have accounts in more than one branch. Translate it to a corresponding SQL query:

$$\delta(\pi_{C.ssNr,C.name}(\sigma_{C.ssNr=AA.customer\, C.ssNr=AB.customer\, AA.branch!=AB.branch}(\rho_C(Customers) \times \rho_{AA}(Accounts) \times \rho_{AB}(Accounts)))) \quad (2)$$

5B (6p)

Write a relational algebra expression that lists all branches together with the number of customers they have, i.e. the number of customers that have accounts in that branch.

Block 6 - Transactions **max 6p**

Use the relations for the banking enterprise from the previous blocks when answering the following problems.

6A (4p)

Consider the following two operations. One transfers a given amount from one account to another, the other adds interest to an account.

```
Program 1:
... get values for accountA/B, branchA/B and amount from the user ...
1 balance := SELECT balance FROM ...
             WHERE account = accountA AND branch = branchA;
2 IF (balance >= amount) THEN
3  UPDATE ... SET balance = balance - amount
     WHERE account = accountA AND branch = branchA;
4  UPDATE ... SET balance = balance + amount
     WHERE account = accountB AND branch = branchB;
5 END IF;
```

```
Program 2:
... get values for accountA and branchA from the system ...
1 rate := SELECT rate FROM InterestRates WHERE ...;
2 UPDATE ... SET balance = balance * rate
    WHERE account = accountA and branch = branchA;
```

(i) (1p)
For the Program 1 specified above, what atomicity problems could arise if if was not run as a transaction?

(ii) (1p)
For the two programs specified above, what isolation problems could arise if they were run in parallel without being run as transactions?

(iii) (2p)
Which of the four possible isolation levels would solve both of these problems (more than one answer possible)?

6B (6p)

Give a realistic example from the banking domain of two transactions that would interfere with each other when run in parallel with no transaction management, but that would no longer interfere when run with isolation level Read Committed.

Block 7 - Semi-structured Data and XML **max 8p**

A web-based company, HomeFinder.se, specializes in finding free apartments all over Sweden. They use a database based on a semi-structured model, with XML as the interface language to the database. Their database can be described by the following document type definition:

```
<!DOCTYPE HomeFinder [

<!ELEMENT Town (Area+)>
<!ELEMENT Area  (Appartment*)>
<!ELEMENT Appartment   (#EMPTY)>
<!ELEMENT Customer (#EMPTY)>

<!ATTLIST Town
    name ID #REQUIRED>
<!ATTLIST Area
    name ID #REQUIRED>
<!ATTLIST Appartment
    id        ID #REQUIRED
    address   CDATA #REQUIRED
    size      CDATA #REQUIRED
    nrOfRooms CDATA #REQUIRED
    rent      CDATA #REQUIRED>
<!ATTLIST Customer
    email   ID #REQUIRED
    name    CDATA #REQUIRED
    phoneNr CDATA #IMPLIED
    address CDATA #IMPLIED
    areasOfInterest IDREFS #IMPLIED>

]>
```

7A (4p)

(i) (2p)

The following XML document is (part of) an instance of the HomeFinder database:

```
<HomeFinder>
 <Town name="Göteborg">
  <Area name="Johanneberg">
   <Appartment id="03482"
     address="Gibraltargatan 12"
     size="43.8" nrOfRooms="2"
     rent="7642"/>
   <Appartment id="03647"
     address="Rännvägen 3"
     size="22.3" rent="4227"/>
  </Area>
 </Town>
 <Town name="Borås">
 </Town>
 <Customer
   email="emil@chalmers.se"
   phoneNr="031-112233"
   areasOfInterest="Johanneberg, Borås">
   Emil Johnsson
 </Customer>
 ...
</HomeFinder>
```

However, this document does not fully conform to the DTD given above. State in what way(s) the document fails to match the specification.

(ii) (2p)

For a document conforming to the schema given above, what would the following XPath expression compute? Answer in plain text:

```
//Area/*[@nrOfRooms>=3]/..
```

7B (6p)

(i) (3p)

For a document conforming to the schema given above, what would the following XQuery expression compute? Answer in plain text:

```
FOR $a IN //Area
LET $x := (FOR $p IN $a/*
           ORDER BY $p/@size
           RETURN $p)
RETURN <Result name=($a/@name)/>({$x})</Result>
```

(ii) (3p)

Write an XQuery expression that returns all appartments in areas that are listed as interesting by at least 100 customers.

7C (8p)

For the schema given above, give an example of a query that would be impossible to write using XPath. Explain why, and show the XQuery expression that implements the query.