EXAM
Databases (INN120/TDA355/TDA356/TDA357)
Informationssystem (TDA426)

DAY: 18 Dec 2006 TIME: 14:00 – 18:00 PLACE: V

Responsible:	Niklas Broberg, Computing Science tel. 772 10 88, mobil 0706 49 35 46
Results:	Will be published by the middle of January at the latest
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 35 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
- Write your personal number on every page
- 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

- The problems have been designed to give short answers. No 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) (4p)

A small association of ornithologists wants a database that will make it easier for them to keep track of the sightings of rare (and common) birds in their general area. In particular they wish to tie bird species to where and when they have been sighted. To keep track of the various species, they wish to use the scientific names for classification, but slightly simplified.

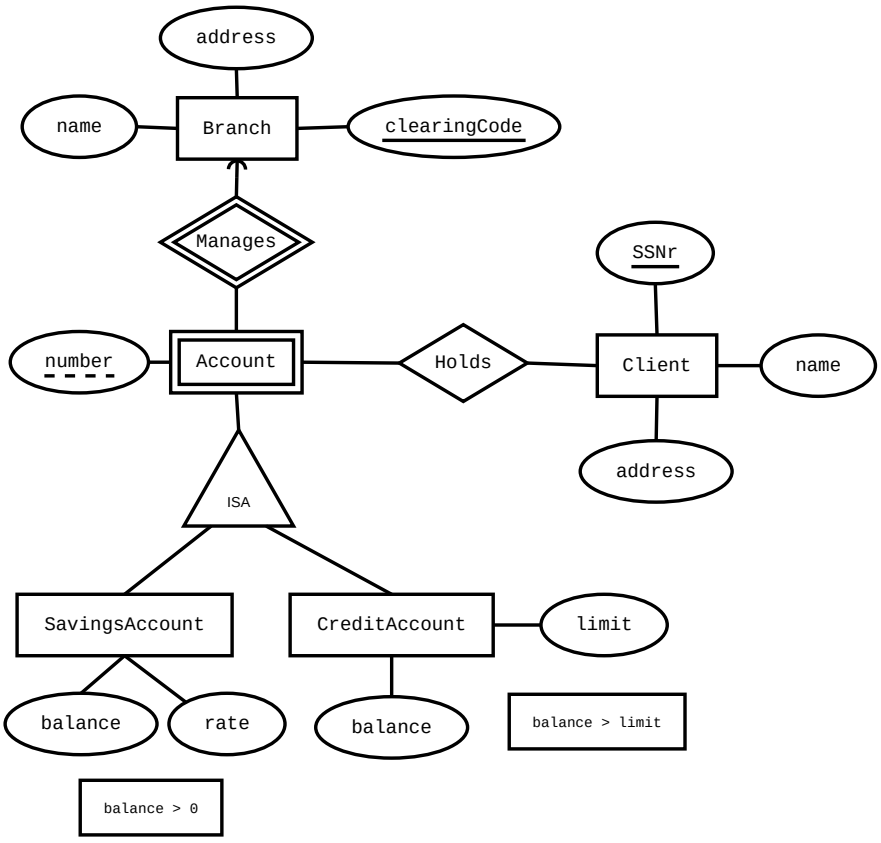
All species of birds belong to some *family* of species, identified by its latin name. As an example, all crows, magpies, ravens and other related birds belong to the family *Corvidae* (Crow-like birds). Each such family is comprised of a number of *genus* (or genera in plural), for instance crows and ravens both belong to the genus *Corvus* (Crows), while magpies belong to the genus *Pica* (Magpies). Finally each genus contains a number of separate *species*, for instance the common Hooded Crow (*Corvus cornix*) and the Raven (*Corvus corax*). Note that only the species identifier (e.g. corax) is not enough to identify a species, since different species within different genera could have the same identifier. The association wants all this information covered in their database so that they can easily see trends in families or genera, as well as species.

For the sightings of these birds, they have divided their area of interest into a number of smaller regions. For each time a bird is sighted, they want to store what species of bird it was, within which region it was sighted, the date and time of the sighting, and the number of birds sighted. In the very rare instance that two different sightings of the same species were to be reported for the exact same date, time and region, then the number of birds from the two would be added together and stored only once.

Your task is to draw an ER diagram that correctly models this domain and its constraints.

(ii) (4p)

Here is an ER diagram describing the domain of a banking enterprise. The diagram should be self-explanatory, except note that both the subclasses of Account have a balance attribute. This is by design, and is done in order to enable the use of separate checks on the balances in the database.



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

1B (12p)

Relationships involving three or more entities are generally referred to as multi-way relationships. These are much less common than relationships involving only two entities, but in the cases where they are needed they are crucial. Using multi-way relationships may lead to problems though, specifically when it comes to marking multiplicity (i.e. many-to-many, many-to-one etc).

Give an example of your own of a multi-way relationship that cannot be modelled using standard ER modelling techniques. Describe it in any way you like, using words and diagrams to convey your meaning.

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 → *Classes.letter*
Customers(<u>*idNr*</u>, *name*, *phoneNumber*)
Rentals(<u>*boat*</u>, <u>*date*</u>, *class*, *customer*, *phoneNumber*)
    boat → *Boats.regNr*
    class → *Classes.letter*
    customer → *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, functional and multi-valued, 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 used to keep track of games in a small women's football league. The league consists of a number of teams, each identified by its name. Each team has a home arena where it plays (most of) its home games.

A team consists of a number of players, each identified by the number on the shirt they wear during play. For each player, her name and social security number is stored, the latter being unique to ensure that the same player is not registered for more than one team.

A game is played between two teams, a home team and an away team. If possible the game will be played at the home arena of the home team, but for various reasons this is not always possible, so for each game the arena that the game is played on is also stored. All goals are stored, numbered from 1 in the order they are scored. Information on goals include who scored, and at what time. Also for each goal, a number of so called assists will be stored, which is those players who were the ones to pass the ball to the person scoring. For each score, from none up to three assists may be recorded.

You are given the following schema of a database for a web forum:

Arenas(*address*, *name*)
Teams(*name*, *coachName*, *homeArena*)
    homeArena → *Arenas.address*
Player(*team*, *number*, *name*, *ssnr*)
    team → *Teams.name*
    ssnr unique
Games(*gid*, *date*, *time*, *at*, *home*, *away*)
    at → *Arenas.address*
    home → *Teams.name*
    away → *Teams.name*
    home != away
Goals(*game*, *nr*, *scorer*, *team*)
    game → *Games.gid*
    (*scorer*, *team*) → *Players.*(*number*, *team*)
    team must be either home or away team for the game
    $nr > 0$
    (*game*, *nr*, *team*)*unique*
Assists(*game*, *goal*, *player*, *team*)
    (*game*, *goal*, *team*) → *Goals.*(*game*, *nr*, *team*)
    (*player*, *team*) → *Players.*(*number*, *team*)

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 the specified checks.

3B (8p)

The special check that the team of a scorer is actually one of the teams in the game, (*team must be either home or away team for the game*), could be written as an assertion instead of a check. Discuss the pros and cons of each of these two approaches, and show what problems that could arise in either, if any.

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

Use the relations for the football league from the previous block when answering the following problems.

4A (4p)

(i) (2p)
Write an SQL query that for each team lists all arenas that the team has played on as home team, that are *not* its home arena.

(ii) (2p)
Write an SQL query that for each player lists the number of goals she has scored. Players who have not scored any goals need not be listed (but it is ok to do so).

4B (6p)

Write a query that lists the number of games each team has played, either as home or away team.

4C (8p)

Write a query that computes, for each player, the total number of goals and the total number of assists she has made. The result should contain these two numbers as separate columns for each player. A player that has scored no goals, but has made some assists should be listed as having made 0 goals and the proper number of assists, and the same for the other way around. A player that has scored neither goals nor assists need not be listed at all (but it is ok to do so). The result should be listed so that the player with the highest sum of goals plus assists should be listed first. If two players have the same sum, the one with the more goals should be listed first.

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

Use the relations for the football league from the previous blocks when answering the following problems.

5A (3p)

(i) (1p)
What does the following relational-algebraic expression compute (answer in plain text):

$$\sigma_{x>1}(\gamma_{A.address,count(*)\rightarrow x}(\sigma_{T.homeArena=A.address}(\rho_A(Arenas) \times \rho_T(Teams))))$$

(ii) (2p)
The following relational-algebraic expression returns the list of all teams that have played two or more games on the same day at some point. Translate it to a corresponding SQL query:

$$\sigma_{x>=2}(\gamma_{team,date,count(gid)\rightarrow x}(\pi_{gid,date,home\rightarrow team}(Games) \cup \pi_{gid,date,away\rightarrow team}(Games)))$$

5B (4p)

Write a relational-algebraic expression that returns all players who have scored at least five goals.

5C (6p)

Write a relational-algebraic expression that for each match returns the final score, i.e. the number of goals scored for each team, for instance 3-0 if the home team scored 3 goals and the away team none. The four columns returned should be the names of the two teams, and their respective scores.

Block 6 - Transactions **max 6p**

Use the relations for the football league from the previous blocks when answering the following problems.

6A (4p)

Assume we want a program for scheduling matches. The program should work as follows:

The responsible user supplies the home and away teams for the match, and the day and time that the match is to be played (we assume that this is decided higher up and cannot be changed). The program checks if the home team's home arena is free at that day and time, and if it is the program books the match to be played on that arena. If that arena is not free, the program should instead list all arenas that *are* free, and let the responsible choose one of them to book instead. With pseudocode, and using :var to denote variables, we assume it looks like the following:

```
... user supplies teams and time into :home, :away, :date and :time...
1 arena := SELECT homeArena FROM Teams WHERE name = :home;
2 homeIsTaken := SELECT COUNT(*) FROM Games WHERE (...)
3  if (:homeIsTaken == 1) then
4   freeArenas := SELECT address FROM Arenas WHERE address NOT IT (...)
5   ... list arenas to user and let user choose one into :arena ...
6 INSERT INTO Games VALUES (DEFAULT, :date, :time, :arena, ...)
```

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

(ii) (1p)
For the program specified above, what isolation problems could arise if it was not run as a transaction at a sufficiently restrictive isolation level?

(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 for the football league domain of two transactions that could 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**

The same ornithologist association as described in the first block have grown tired of the rigid relational model and want to try something more flexible. In particular they want to be able to include the sightings of so called *subspecies* in their database. Subspecies are variations within the same species of bird, for instance the common White Wagtail (Motacilla alba) has a number of subspecies such as the Pied Wagtail (Motacilla alba yarelli) and the Masked Wagtail (Motacilla alba personata).

You have come up with the following DTD as a schema for their new database:

```
<!DOCTYPE Birdwatch [

<!ELEMENT Birdwatch  (Family*)>
<!ELEMENT Family     (Genus+)>
<!ELEMENT Genus      (Species+)>
<!ELEMENT Species    (Sighting*, Subspecies*)>
<!ELEMENT Subspecies (Sighting*)>
<!ELEMENT Sighting   EMPTY>

<!ATTLIST Family
    identifier ID #REQUIRED
    commonName CDATA #IMPLIED>
<!ATTLIST Genus
    identifier ID #REQUIRED
    commonName CDATA #IMPLIED>
<!ATTLIST Species
    identifier CDATA #REQUIRED
    commonName CDATA #IMPLIED>
<!ATTLIST Subspecies
    identifier CDATA #REQUIRED
    commonName CDATA #IMPLIED>
<!ATTLIST Sighting
    region CDATA #REQUIRED
    date   CDATA #REQUIRED
    time   CDATA #REQUIRED
    number CDATA #IMPLIED>

]>
```

7A (4p)

(i) (2p)

Give a minimal XML document that contains information about at least one sighting, and is valid with respect to the given DTD. Feel free to use the names of things that I have used in the examples, as I certainly do not expect you to know the latin names of any birds yourselves (although it's fine to make some up). Note that minimal does not refer to the length of strings given, but the number of elements and attributes in the document.

(ii) (2p)

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

```
FOR $f IN //Family
LET $x := count($f/*)
WHERE $x > 10
ORDER BY -($x)
RETURN ({$f})
```

7B (6p)

(i) (2p)

The following XQuery expression computes the total number of sightings for each species of bird, but does not account for subspecies.

```
FOR $s IN //Species
LET $x := count($s/Sightings)
LET $g := $s/../@identifier
LET $p := $s/@identifier
RETURN <Count genus={$g} species={$p} total={$x} />
```

Change one line in the above query so that sightings of subspecies are counted together with their nominal (parent) species, i.e. each sighting of a subspecies should be counted as a sighting of the main species instead.

(ii) (4p)

Write an XQuery expression computes the same as the above, but counts species and subspecies separately. Results should be on the same form as above, with the counts of subspecies having an extra attribute for the subspecies identifier.

7C (8p)

(i) (2p)
Why are the types of the identifiers for species and subspecies not set to ID in the DTD above?

(ii) (6p)
Argue why the semi-structured datamodel handles this situation better than the relational model. Show what the problem would have been with subspecies in the relational model, and why that is not a problem in the semi-structured model. (Hint: Don't write a full essay, it won't increase your chances.)