EXAM
Databases (DIT620/TDA355/TDA356/TDA357)

DAY: 9 Mar 2012 TIME: 14:00 – 18:00 PLACE: V

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 35 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)

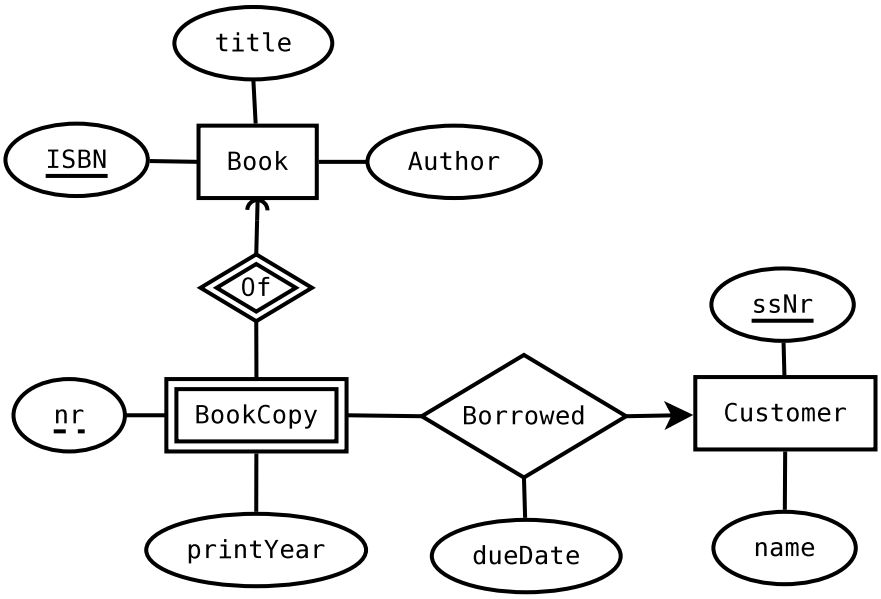
(i) (5p)

An online user-centric news site needs a database to keep track of its users and articles. Each user that registers with the site picks a unique login/username, and submits their name and email address. Having registered, a user can comment on articles posted by other users. To be allowed to publish articles themselves, a user must upgrade to author status. They do this by providing an image, and a title for themselves e.g. "M.Sc. in Computer Science". These will then appear along their name in the signature beneath their articles, to give readers a good idea of who the author is.
Articles can be published by one or more authors together. For each article, the database needs to store its title and its contents, and also the date it was published.
Further, each article can be tagged by users (possibly, but not necessarily, the authors) as belonging to certain topics, e.g. "Politics" or "Sports". This allows readers on the site to browse articles by the topics they are interested in. Each time an article is tagged, the database should store who tagged it and when. An article can be tagged any number of times, each time with a new tag.
Finally, users may comment on articles. Comments are treated as a flat list, where each comment gets a running count number (i.e. 1,2,3...) for that article. For each comment, the database should store its number and contents, as well as the user who wrote it.

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

(ii) (3p)

The E/R diagram below depicts a rudimentary database used for a library.



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

1B (12p)

(i) (6p)

A small board game store needs a database to store information about their products. The store sells not only board games but also related books and movies. Products thus fall into one of four different categories: *games*, *movies*, *books*, or *other*. For each game, the database should store its name, a description text, the minimum and maximum number of players and the approximate playing time. For each movie, the database should store its name, length, director and producer, as well as a description. For each book, the database should store its name and author, the number of pages, and a teaser text. For other products, only the name and a description should be stored.

Every product also has a unique identifier, and a base price. The store sometimes runs campaigns during which the prices of some products is lowered. The database should store the lowered price for products currently part of a campaign.

Finally, for each product the database needs to store how many items of that product the store has in stock, and the name, phone number and organisation number of its supplier.

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

(ii) (6p)

Translate your ER diagram for the board game store into relations, such that your translation makes use of two different translation schemes for different sub-entities.

Warning: Since this sub-task depends on the solution to the previous sub-task, any errors in your ER diagram produced for that task may cause further errors here.

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

2A (8p)

A small banking enterprise uses a database to store customer information as well as account information. The bank is organized into local branches, that all have a branch code, a name and an address, and all accounts belong to a particular branch. Account numbers are unique within each branch, but different branches may use the same account numbers. There are two kinds of accounts; savings and credit accounts. Savings accounts must have a positive balance, whereas credit accounts may have a negative balance down to some lower limit. Each account belongs to a specific customer, i.e. no accounts are shared. The credit rating of a customer denotes how much credit they may at most have among all their credit accounts.

You are given the following schema of their intended database:

Accounts(<u>*branchCode*</u>, *name*, *address*, <u>*accountNr*</u>)
Customers(<u>*ssNr*</u>, *name*, *creditRating*, <u>*account*</u>, <u>*branch*</u>)
(*branch*, *account*) → *Accounts*.(*branchCode*, *accountNr*)
SavingsAccounts(<u>*branch*</u>, <u>*account*</u>, *balance*)
(*branch*, *account*) → *Accounts*.(*branchCode*, *accountNr*)
CreditAccounts(<u>*branch*</u>, <u>*account*</u>, *balance*, *limit*)
(*branch*, *account*) → *Accounts*.(*branchCode*, *accountNr*)

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 are expected to hold.

(ii) (1p)
With the dependencies you have found, identify all BCNF violations in the relations of the database.

(iii) (3p)
Do a complete normalization of the schema, so that all relations are in BCNF. (It's the end product that's important, not the steps you take to get there.)

2B (12p)

A clinic wants to use a database to store patient journals. Apart from basic information about name and year of birth, each patient's journal should contain entries written by doctors for each visit, as well as information about what medicines the patient is taking, and what substances the patient is allergic to. For each medicine, the prescribed dosage should be stored. For each journal entry, the entry itself should be stored and which doctor that wrote it. Doctors are identified with a unique ID number. Each doctor has one or more fields of specialty, e.g. cardiology, osteopathy etc, which should also be stored.

Journals(*patientId*, *name*, *yearOfBirth*, *medicine*, *dosage*,
allergy, *entry*, *doctorId*, *doctorName*, *doctorSpecialty*)

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

(i) (8p)
Find all non-trivial independencies (multi-valued dependencies) that are expected to hold for this domain, given the domain description above. Remember that functional dependencies are also independencies.

(ii) (4p)
Show a decomposition of *Journals* that satisfies 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 an online site for finding apartments in Gothenburg. The database keeps track of apartments and landlords, as well as clients and their interests.

You are given the following schema of their intended database:

Areas(<u>*name*</u>)
Landlords(<u>*orgNr*</u>, *name*)
Properties(<u>*code*</u>, *streetName*, *streetNr*, *postalCode*, *area*, *yearBuilt*, *nrFloors*, *owner*)
    owner → *Landlords.orgNr*
    area → *Areas.name*
    (*streetName*, *streetNr*, *postalCode*)*unique*
Apartments(<u>*property*</u>, <u>*nr*</u>, *size*, *nrRooms*, *rent*)
    property → *Properties.code*
    size >= 0
    rent >= 0
    nrRooms >= 1
Clients(<u>*ssNr*</u>, *name*, *income*, *minSize*, *maxSize*, *minRooms*, *maxRooms*, *minRent*, *maxRent*)
    minSize, *maxSize*, *minRooms*, *maxRooms*, *minRent*, *maxRent* >= 0
    minSize <= *maxSize*
    minRooms <= *maxRooms*
    minRent <= *maxRent*
ClientAreas(<u>*client*</u>, <u>*area*</u>)
    client → *Clients.ssNr*
    area → *Areas.name*
ClientInterests(<u>*client*</u>, <u>*property*</u>, <u>*apartment*</u>)
    client → *Clients.ssNr*
    (*property*, *apartment*) → *Apartments.*(*property*, *nr*)

A property represents a building at a particular address. The Swedish Land Survey Bureau ("Lantmäteriet") has an identification system for all property, and the codes used in that system are used as identifiers for properties in the database. There can be many apartments available at the same time within the same property, each identified using a standardized number system (exactly how is irrelevant for this task).
When a clients registers at the site they provide information about their preferences, to help filter out uninteresting apartments. Apart from preferences in size, number of rooms and rent levels (which should be self-explanatory), the user may specify in what areas they are interested in finding an apartment, e.g. Johanneberg, Guldheden, Landala etc.
When a client sees an available apartment that they are interested in, they should register their interest for the benefit of the landlord that may then send offers (handled outside the scope of this database).

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.
Hint: Use the $<= ALL$ operator for the constraint in *Clients*.

3B (6p)

The most common query over the database is to find all available apartments that suit a client's preferences. The following view lists all such apartments for each client:

```
CREATE VIEW Available AS
SELECT C.ssNr, A.property, A.nr AS apartment
FROM Clients C, ClientAreas CA, Properties P, Apartments A
WHERE C.ssNr = CA.client
  AND CA.area = P.area
  AND A.property = P.code AND A.nr = P.nr
  AND A.size BETWEEN C.minSize AND C.maxSize
  AND A.nrRooms BETWEEN C.minRooms AND C.maxRooms
  AND A.rent BETWEEN C.minRent AND C.maxRent;
```

(i) (3p)
To ask for information from this view, what is the minimal set of privileges needed by the application?

(ii) (3p)
Apart from the indexes already induced by the primary keys of the respective tables, what indexes could help speed this query up? Note that you do not need to do any calculations to answer this question, only look at the SQL code.

3C (8p)

When a new client registers on the site, they do so through an interface in which they specify their basic information – social security number and name – and up to three areas of interest. All other information about income and preferences can be added later in the client's settings, but should at the point of creation be set to some default values.

Sketch a solution to this task, involving whatever you find necessary of views, triggers, table element properties and privileges. I want to know what components your solution uses and how they interact, but no implementation details.

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

Use the relations for the apartment finder site from the previous block when answering the following problems.

When you are asked to list all X, you need only return the key attributes of X.

4A (4p)

(i) (1p)
Write an SQL query that lists all apartments with at least 4 rooms.

(ii) (1p)
Write an SQL query that lists all clients in order of income, highest first.

(iii) (2p)
Write an SQL query that finds the largest apartment(s), i.e. with the highest size among all available apartments. If more than one apartment qualifies, list all of those that do.

4B (6p)

Write a query that lists, for each area, the number of available apartments in that area. Areas with no available apartments should be listed as having 0 apartments available.

4C (8p)

Write a query that lists, for each apartment, all clients that have *only* registered interest for that particular apartment. If an apartment has no clients with registered interest yet, list the apartment with `NULL` as client.

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

Use the relations for the apartment finder site from the previous blocks when answering the following problems.

5A (3p)

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

$$\tau_x(\gamma_{area,nrRooms,AVG(rent)\rightarrow x}(\sigma_{property=code}(Apartments \times Properties)))$$

(ii) (2p)
Translate the following relational algebra expression to corresponding SQL:

$$\gamma_{ssNr,AVG(rent)}(Apartments \bowtie (\pi_{ssNr,property,nr\rightarrow apartment}(\sigma_{client=ssNr}(Clients \times ClientInterests))))$$

5B (4p)

Translate the following SQL query to relational algebra:

```
SELECT orgNr
FROM Landlords, Properties
WHERE orgNr = owner
GROUP BY orgNr
HAVING MIN(yearBuilt) < 1800;
```

5C (6p)

Write a relational algebra expression that lists the client who has registered interest for most apartments. If more than one client are tied for first place, list all of those.

Block 6 - Transactions **max 6p**

Use the relations for the apartment finder site from the previous blocks when answering the following problems.

6A (3p)

Consider the situation where a landlord employee wants to put up a new available apartment on the site. If the property in which the apartment is located is already in the database, it's a simple matter of adding the required information for the apartment itself. Otherwise information about the property must be added as well. Consider the following program (partly in pseudo-code), for inserting new apartments. In the code I prefix program variables with : just to distingush them from attributes (i.e. you don't need to worry about any connection to PSM or the like).

```
1 ... user submits :property, :nr, :size, :nrRooms and :rent ...
2 INSERT INTO Apartments VALUES
    (:property, :nr, :size, :nrRooms, :rent)
3 if (NOT EXISTS (SELECT * FROM Properties WHERE code = :property)) {
4   ... ask user to submit information about property; :streetName etc ...
5   INSERT INTO Properties VALUES
      (:property, :streetName, ...)
  }
```

(i) (1p)

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

(ii) (2p)

For the program as specified above, what isolation problems could arise if it was not run as a *serializable* transaction?

6B (4p)

If the program above was not run as a transaction, the insertion on line 2 would fail. Why would it fail, and why would it not fail if a transaction was used?

6C (6p)

Compare what would happen if the program above was run as a transaction with isolation level SERIALIZABLE compared to if it was run with isolation level READ COMMITTED. Point out benefits and drawbacks of the two choices for this particular problem.

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

The following DTD attempts to model a small part of the data kept by the Swedish parliament, restricted to the parts pertaining to its members, committees and discussions held therein.

```
<!DOCTYPE Parliament [

<!ELEMENT Parliament (Politician+,Committee+)>
<!ELEMENT Politician (CDATA)>
<!ELEMENT Committee  (Discussion*)>
<!ELEMENT Discussion (Entry+)>
<!ELEMENT Entry      (CDATA)>

<!ATTLIST Politician
    idNr      ID    #REQUIRED
    party     CDATA #REQUIRED
    committee CDATA #IMPLIED >
<!ATTLIST Committee
    name ID #REQUIRED>
<!ATTLIST Discussion
    title  CDATA #REQUIRED
    date   CDATA #REQUIRED>
<!ATTLIST Entry
    speaker IDREF #REQUIRED>
]>
```

The name of a politician is stored as a `CDATA` child. The `CDATA` child of a discussion entry is a textual representation of what the speaker said. Everything else should be self-explanatory (and not necessarily relevant).

7A (5p)

(i) (2p)
Give an example XML document that is valid with respect to the DTD above, and that contains information about at least one discussion.

(ii) (3p)
Write an XPath expression that lists all politicians who are members of the "Folkpartiet" party.

7B (8p)

(i) (3p)

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

```
<Result>
{ for $d in ( doc("parliament.xml") )
  for $p in $d//Politician
  let $x := (for $e in $d//Entry
             where $e/@speaker = $p/@idNr
             return $e)
  let $c := count($x)
  order by $c
  return <Speaker name="{data($p)}">{$c}</Speaker> }
</Result>
```

(Slightly simplified, the `data` function in XQuery returns any non-element (`CDATA`) child of its element argument.)

(ii) (5p)

The following DTD is an alternate representation of the domain:

```
<!DOCTYPE Parliament [
<!ELEMENT Parliament (Committee+)>
<!ELEMENT Committee  (Politician+,Discussion*)>
<!ELEMENT Politician (EMPTY)>
<!ELEMENT Discussion (Entry+)>
<!ELEMENT Entry      (EMPTY)>
<!ATTLIST Committee
    name ID #REQUIRED>
<!ATTLIST Politician
    idNr      ID    #REQUIRED
    party     CDATA #REQUIRED
    name      CDATA #REQUIRED>
<!ATTLIST Discussion
    title  CDATA #REQUIRED
    date   CDATA #REQUIRED>
<!ATTLIST Entry
    speaker IDREF #REQUIRED
    content CDATA #REQUIRED>
]>
```

Write an XQuery expression that takes the document `parliament2.xml`, valid with respect to this alternate DTD, and transforms it so that it is instead valid with respect to the original DTD, without losing any information.