EXAM
Databases (DIT620/TDA355/TDA356/TDA357)

DAY: 14 Mar 2014 TIME: 8:30 – 12:30 PLACE: H

Responsible:	Niklas Broberg, Computer Science and Engineering 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)

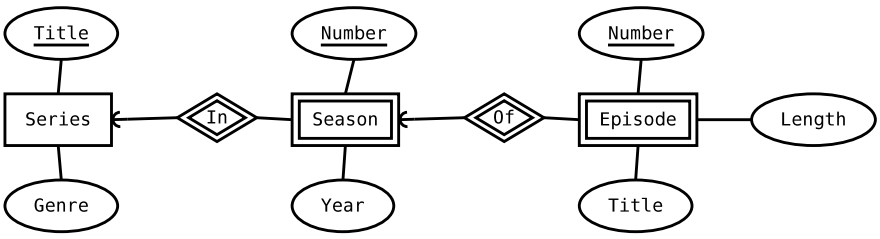
(i) (5p)

A small hardware store wants a database to keep track of its business.
Each item for sale is identified by a unique bar code. For each item, the store needs to track its manufacturer and purchase price (what the store pays for it from the manufacturer), as well as its base retail price (what the store normally sells it for) and how many copies the store currently has in stock.
For manufacturers, only their name and a contact phone number are needed.
The store often runs campaigns, either by selling one or more items at a fixed discount price, or by giving a percent discount on a range of items. Campaigns are identified through an artificial key string on a special format. Apart from this identifier, the database also needs to store the start and end dates for the campaign, and of course the discount in question, fixed or percent.
Every purchase done in the store should be logged in the database. A purchase is identified by the unique receipt number. For each purchase, the database should store the date and time, what items are included in the purchase, how many of each item are bought and at what price (could be the base retail price or a discounted price).
If the purchase is paid by credit card, the database should store the last four digits of the credit card number.

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 an online service streaming TV series.



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

1B (12p)

Below is part of a database schema used for an assignment submission system (akin to Fire):

Students(<u>*ssNr*</u>, *email*, *name*)
Groups(<u>*nr*</u>)
InGroup(<u>*student*</u>, <u>*group*</u>)
    student → *Students.ssNr*
    group → *Groups.nr*
Graders(<u>*email*</u>, *name*)
Assignments(<u>*nr*</u>, *title*)
IsSoloAssignment(<u>*assignment*</u>)
    assignment → *Assignments.nr*
IsGroupAssignment(<u>*assignment*</u>)
    assignment → *Assignments.nr*
Submissions(<u>*id*</u>, <u>*grader*</u>)
    grader → *Graders.email*
Files(<u>*submission*</u>, <u>*file*</u>)
    submission → *Submissions.id*
SoloSubmissions(<u>*submission*</u>, *assignment*, *student*)
    submission → *Submissions.id*
    assignment → *IsSoloAssignment.assignment*
    student → *Students.ssNr*
GroupSubmissions(<u>*submission*</u>, *assignment*, *group*)
    submission → *Submissions.id*
    assignment → *IsGroupAssignment.assignment*
    group → *Groups.nr*

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.

(i) (8p)
Reconstruct the ER diagram that led to these relations and constraints.

(ii) (4p)
During the translation that led to the above schema, the ER approach will have been used consistently. For each case where applicable in the diagram you have reconstructed, state what constraints you would lose by instead using the NULL approach.

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

2A (8p)

A website maintains a searchable database over academic publications.

A *publication* can be either a *journal*, or the printed proceedings following a *conference*. Journals publish *editions* with regular intervals, which could range from several times per year to once every few years. These editions are simply referred to by their ordinal number, e.g. the 14th edition.
Each edition contains a set of *articles*, each of which is written by one or more authors. An article has a title, unique within the edition, as well as an artificial identification number, unique within the database.
A conference – e.g. the International Conference on Databases in Education (ICDE) – is held once per year, in a different location each time.
For each time the conference is held, the printed proceedings contains all the articles presented at the conference that year. For simplicity, these proceedings are, like with journals, also called editions and referred to by ordinal numbers.
Articles published in conference proceedings are typically fairly short, while those published in journals are often significantly longer.
Every author is affiliated with a university or institute. It is not uncommon for an article to be written collaboratively by authors with different affiliations.
Articles typically refer to other articles through *citations*. The influence of an article or a whole publication can be measured by how many times they are cited.

You are given the following schema of their intended database:

Affiliations(<u>*name*</u>, *city*, *country*)
Authors(<u>*id*</u>, *name*, *affiliation*, *country*)
    affiliation → *Affiliations.name*
Publications(<u>*id*</u>, *name*)
Editions(<u>*publication*</u>, <u>*nr*</u>, *pubName*, <u>*articleId*</u>, *articleTitle*, <u>*author*</u>)
    publication → *Publications.pubId*
    author → *Authors.id*
Conferences(<u>*conference*</u>, <u>*nr*</u>, *year*, *country*, *city*)
    (*conference*, *nr*) → *Editions.*(*publication*, *nr*)
Citations(<u>*article*</u>, <u>*citedArticle*</u>, <u>*citedPublication*</u>)
    article → *Editions.id*
    (*citedArticle*, *citedPublication*) → *Editions.*(*articleId*, *publication*)

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 between attributes.

(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. Also ensure that all key constraints are properly captured. (It's the end product that's important, not the steps you take to get there.)

2B (12p)

Consider the domain presented for the publications site above, and then consider the hypothetical functional dependencies and independencies listed below. For each of these, state why it does *not* hold for this domain, and give an example of what could go wrong in the database if the respective functional dependency or independency was enforced:

i. $authorName \rightarrow country$

ii. $articleId \rightarrow authorId$

iii. $conference \twoheadrightarrow country$

iv. $publicationId, nr \twoheadrightarrow articleId, articleTitle$

(Note for students having taken the course several years ago: Back then we referred to independencies as "multi-valued dependencies (MVDs)". The two terms are equivalent.)

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 hotel.

You are given the following schema of their intended database:

Floors(*floorNr*)
    $floorNr > 0$
Rooms(*floor*, *nr*, *pricePerNight*)
    $floor \rightarrow Floors.floorNr$
    $nr > 0$
    $pricePerNight > 0$
Guests(*id*, *name*, *ccNumber*)
StaysIn(*guest*, *room*, *floor*, *checkInDate*, *checkOutDate*)
    $guest \rightarrow Guests.id$
    $(room, floor) \rightarrow Rooms.(nr, floor)$
    $checkInDate < checkOutDate$

KeyCards(*cardId*)
MasterKeys(*card*)
    $card \rightarrow KeyCards.cardId$
GuestKeys(*card*, *guest*)
    $card \rightarrow KeyCards.cardId$
    $guest \rightarrow Guests.id$
Opens(*card*, *room*, *floor*)
    $card \rightarrow KeyCards.cardId$
    $(room, floor) \rightarrow Rooms.(nr, floor)$

Items(*code*, *description*, *price*)
    $price > 0$
Purchases(*receiptNr*, *room*, *floor*)
    $(room, floor) \rightarrow Rooms.(nr, floor)$
Includes(*receipt*, *item*, *qty*)
    $receipt \rightarrow Purchases.receiptNr$
    $item \rightarrow Items.code$
    $qty > 0$

The hotel is a multi-story building, with rooms on every floor. The rooms on each floor are numbered consecutively from 01, so to identify a particular room, both the floor number and the room's number on that floor are needed.
Once a guest has stayed at the hotel, their information is saved for future visits. When a guest checks in, the room they stay in is stored together with the date of check-in. The check-out date is set to NULL at check-in, and filled in when the guest checks out.
The hotel uses electronic cards as keys. Each card can be programmed to open the door to any rooms in the hotel, and the same key can be programmed open multiple rooms. A key can also be programmed to be a master key, which makes it able to unlock any door.
When a guest checks in, they are given one or more keys to their room(s), one for each person in the group (e.g. a family) sharing the room.
Further, the hotel has a wide range of items for sale. Anything the hotel offers for sale is considered an item, e.g. breakfast or meals in the hotel restaurant, drinks in the bar, massage in the spa, etc. A purchase states all items purchased at a single time, and how many of each (*qty*).
When making a purchase, the guest has the option to put the cost "on the room", for paying upon check-out. Once the purchase is paid for, either right away or upon check-out, the room and floor are set to NULL for the relevant entry.

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)

For various reasons, every so often, a guest may need a new key card. Sometimes keys are wiped by magnetic disturbance, sometimes keys are forgotten inside the room, sometimes keys are dropped or lost. As a precaution, whenever the hotel hands out a replacement key card to a guest who has lost their card, all (non-master) cards that previously opened that guest's room are reset to no longer do so. This is to avoid that a lost card is found by someone else and used to gain entry to a room they should not have access to. This measure is not necessary if the card is merely wiped.

Sketch an overview of how to implement this operation of resetting keys, through the use of views, and/or triggers, and/or privileges. For any views you want to use, give the schema and explain its intended contents. For any trigger you might include, list the trigger head ([BEFORE/AFTER/INSTEAD OF] [INSERT/DELETE/UPDATE] ON which element), and describe its intended operation in broad terms (a simple overview in plain English would be fine, you don't have to write any code). Also specify what privileges the front-end should be granted in order to handle its use cases properly.

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

Use the relations for the hotel 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 unless explicitly instructed otherwise.

4A (4p)

(i) (2p)
Write an SQL query that lists, for each room, all key cards that can open that room. This includes both master keys and cards explicitly programmed to open the room.

(ii) (2p)
Write an SQL query that lists all rooms that are currently occupied.

4B (6p)

Write an SQL query that lists, for each item, its code and description, together with the total amount of money the hotel has received by selling it.

4C (8p)

Write an SQL query that finds the guest(s) who has stayed at the hotel the most times. (In case of ties, list all those tied for first place.)

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

Use the relations for the hotel 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):

$$let\ R1 = \pi_{room,floor}(\sigma_{checkOutDate\ IS\ NULL}(StaysIn))$$
$$\tau_{pricePerNight}(\sigma_{(nr,floor)\notin R1}(Rooms))$$

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

$$\tau_{-x}(\gamma_{R.floor,COUNT(R.room)\rightarrow x}(\sigma_{guest\ IS\ NULL}(\rho_S(StaysIn) \overset{o}{\bowtie}_{(S.floor=R.floor\ \&\ S.room=R.nr)} \rho_R(Rooms))))$$

5B (4p)

Translate the following SQL query to relational algebra:

```
SELECT guest, name, MAX(checkOutDate - checkInDate) AS timeStayed
FROM StaysIn, Guests
WHERE guest=id
GROUP BY guest, name
ORDER BY timeStayed DESC;
```

5C (6p)

Write a relational algebra expression that lists, for each room, the amount of payment due at check-out from purchases.

Block 6 - Transactions **max 6p**

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

6A (3p)

Consider the situation where a lone guest checks in at the hotel. If the guest has not previously stayed at the hotel, she or he is first registered, with name and credit card number. Second, the guest is presented with a list of available rooms, along with prices, and chooses one of them. The guest is then listed as staying in that room, with the current date as check-in, and NULL as check-out date. Finally, a key card is encoded for the guest, to open the room in question.

Consider the following program (partly in pseudo-code), for handling this situation. 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). For simplicity, we assume the existence of a view *FreeRooms*(*floor*, *room*, *pricePerNight*) that lists all available rooms and their prices.

```
1 IF (guest is new) {
2   ... receptionist inputs :name and :ccNumber ...
3   SET :guestId = NEWGUESTID();
3   INSERT INTO Guests VALUES (:guestId, :name, :ccNumber);
  }
4 :listOfRooms = SELECT * from FreeRooms;
5 ... present available rooms to guest, who picks one into :floor and :room ...
6 INSERT INTO StaysIn VALUES (:guest, :room, :floor, TODAY(), NULL);
7 ... receptionist presents a key card with id :card to the system ...
8 INSERT INTO GuestKeys VALUES (:card, :guest);
```

(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 (6p)

Consider the program above. Give a concrete example of a problem that could arise if the progam was run with isolation level *read uncommitted* that could not arise if the program was run with isolation level *read committed*.

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

The following DTD attempts to as faithfully as possible model the same domain and constraints for the hotel as the relations used in the previous blocks.

```
<!DOCTYPE Hotel [
<!ELEMENT Hotel        (Floor+,Guest*,KeyCard*,Item*,Purchase*)>
<!ELEMENT Floor        (Room+)>
<!ELEMENT Room         (Opens*)>
<!ELEMENT Opens        EMPTY>
<!ELEMENT Guest        (Stay*)>
<!ELEMENT Stay         EMPTY >
<!ELEMENT KeyCard      (IsMaster?,Opens*)>
<!ELEMENT Item         EMPTY >
<!ELEMENT Purchase     (Includes+) >
<!ELEMENT Includes     EMPTY >

<!ATTLIST Floor number ID #REQUIRED>
<!ATTLIST Room  number ID #REQUIRED>
<!ATTLIST Opens
    keycard   IDREF #REQUIRED>
<!ATTLIST Guest
    id           ID    #REQUIRED
    name         CDATA #REQUIRED
    ccNumber     CDATA #REQUIRED>
<!ATTLIST Stay
    checkInDate  CDATA #REQUIRED
    checkOutDate CDATA #REQUIRED
    room         IDREF #REQUIRED>
    floor        IDREF #REQUIRED>
<!ATTLIST KeyCard
    cardId   ID    #REQUIRED
    heldBy   IDREF #IMPLIED>
<!ATTLIST Item
    code        ID    #REQUIRED
    description CDATA #REQUIRED
    price       CDATA #REQUIRED>
<!ATTLIST Purchase
    receiptNr CDATA #REQUIRED
    room      IDREF #IMPLIED
    floor     IDREF #IMPLIED>
<!ATTLIST Includes
    item   IDREF #REQUIRED
    qty    CDATA #REQUIRED>
]>
```

7A (5p)

(i) (3p)
Give an example XML document that is valid with respect to the DTD above, and that contains at least one master key (key cards with an `IsMaster` child element).

(ii) (2p)
The DTD above fails to faithfully represent the same domain and constraints as the relational schema specified in block 3 in a number of ways, due primarily to shortcomings with DTDs. Point out two such problems for this DTD.

7B (8p)

(i) (4p)
Give an example of a query over the DTD above that would be impossible to write using XPath. Explain why, and show the corresponding XQuery expression.

(ii) (4p)
Write an XQuery that lists, for each room, all the keys that open that room. For each room, include both key cards listed as opening that room, and master keys. The format should be as follows:

```
<Result>
 <Floor nr="1">
  <Room nr="05">
   <Opens keycard="a9876fa65be0" />
   ... more cards ...
  </Room>
  ... more rooms ...
 </Floor>
 ... more floors ...
</Result>
```