UPPSALA UNIVERSITET

Department of Information Technology

INSTRUCTIONS

Please check that you have the correct exam!

This sheet should always be turned in, even if you haven't solved any of the exam questions.

Each solution should be written on a new paper.

Write your exam code on each new paper.

Please use only *one side* of the papers and *do not* use a pencil with red colour.

Sort the solutions in question order, with question 1 first, before you turn them in.

FRONT SHEET FOR EXAMS

DATE: 2014-01-13

Course name (incl. group)
Database Design I (1DL300)

Your exam code

Semester and year when you were first registered for the course[1]	Programme (or similar)
Time for turning in the exam:	Table number

Nr.	Solved questions (mark with X)	Points earned
1.		
2.		
3.		
4.		
5.		
6.		
7.		
8.		
9.		
10.		
Σ		
Exam grade		

Comment from the teacher:

☐ Exam with bonus points: Grade is not shown[2].

Grade limits:
VG≥ G≥

5≥ 4≥ 3≥

[1] Please note: If you are NOT registered for the course your exam will NOT be graded.
[2] The final result (points including bonus points and grade) will appear at the student portal when the result has been added to Uppdok.

Uppsala University
Department of Information Technology

Exam 2014-01-13

Database Design I (1DL300)

Date	Monday, January 13, 2014
Time	08:00 – 13:00

Teachers on duty:

- Matteo Magnani

Instructions:

Read through the complete exam and note any unclear directives before you start solving the questions.

Make sure you observe the following guidelines:

- You are allowed to use dictionaries to and from English and a calculator, but **no other material**.
- Assumptions outside of what is stated in the question must be explained. Any assumptions made should not alter the given question.
- Write legibly and clearly. Solutions failing to follow this guideline will be given zero points.

1) Questions on theory (12 points)

Indicate the correct answer. Every correct answer gives 3 points, every wrong answer gives -1. If you do not answer you do not get any points and you do not lose any. (The negative points have the effect that random answers will bring you an expected score of 0).

(a) In SQL we can check if a value is null using the predicate:
   1) IS NULL.
   2) = NULL.
   3) ALL.
   4) None of the previous answers.

(b) If A is the primary key of table T, and B is another attribute of T, which of the following SQL queries may return a value which is higher than the value returned by the two other queries?
   1) SELECT COUNT(A, B) FROM T.
   2) SELECT COUNT(distinct A, B) FROM T.
   3) SELECT COUNT(A) from T.
   4) None: they all return the same value.

(c) In the relational model, if a set of attributes K is a superkey of a relation r then:
   1) R contains at least two different tuples t_1 and t_2 with $t_1[K] = t_2[K]$.
   2) R cannot contain two different tuples t_1 and t_2 with $t_1[K] = t_2[K]$.
   3) R contains exactly two different tuples t_1 and t_2 with $t_1[K] = t_2[K]$.
   4) R contains at least two different tuples t_1 and t_2 with $t_1[K] \neq t_2[K]$.

   (with t[K] we notate the projection of t on the attributes in K)

(d) The cardinality of a join between two relations r1 and r2 is:
   1) Always in [min(|r1|, |r2|), |r1| x |r2|].
   2) Always in [0, min(|r1|, |r2|)].
   3) Always in [0, max(|r1|, |r2|)].
   4) None of the previous answers.

   (with |r| we indicate the cardinality or r, i.e., the number of tuples in r)

2) ER modeling (16 points)

A database for a Car Rental company should be designed with the following requirements:

1) For all the customers we want to store the name, surname, telephone number, national identification code (e.g., SSN in the States or Personnummer in Sweden) and an internal identifier assigned by the company.

2) Every car has a model and a code identifying the car among the ones of the same model. For example, there can be two cars with code “Car #1”, one of model “Volvo XC60” and the other of model “Volvo V40”. However, there cannot be two different cars with the same code and of the same model.

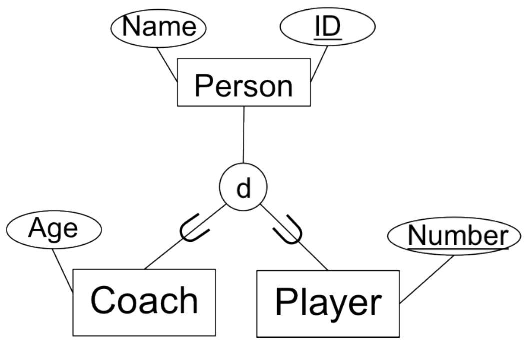
3) For each model we want to represent the number of kilometers per liter and its capacity.
4) For each car we want to represent the year when it was purchased.
5) Customers can book cars from a starting date and time to an ending date and time.
6) A single booking cannot last longer than 3 days. (but the same car can be booked by the same customer multiple times)
7) The same car cannot be booked more than once during the same time.
8) For each booking we want to store the fuel before and after the car has been used by the customer. Both values are not always available.

Tasks:

a) Design an EER diagram representing this scenario.
b) Clearly indicate all requirements that cannot be represented in the diagram or that you prefer not to represent in the diagram to keep it simple, if any.

3) Translation to the Relational Model (15 points)

Consider the following ER diagram:



Tasks:

(a) Translate it to a relational model schema (i.e., indicate the relation name(s) and attributes).
(b) Indicate all the primary keys.
(c) Indicate all the foreign keys, if any.
(d) Indicate all the attributes that allow NULL values, if any.
(e) Indicate all the UNIQUE attributes that are not part of a primary key, if any.

4) SQL (16 points)

Consider the following database schema:

Student(SID, Name, Surname, Age)

Registration(StudentID, CourseID)

Course(CID, Name, Cost)

Tasks:

(a) Write an SQL query to extract the distinct ages of people registered to the course whose name is "Database Design 1".

(b) Write an SQL query to extract the sum of the costs of courses whose identifier starts with "IT".

(c) Define a view in SQL showing for each course the average age of the students registered to it.

(d) Write an SQL query to extract the students registered to courses with an average age lower than their age. If you want, you can use the view defined in (c). Return the distinct student identifiers.

5) Relational Algebra (5 points)

Consider again the following database schema:

Student(SID, Name, Surname, Age)

Registration(StudentID, CourseID)

Course(CID, Name, Cost)

Tasks:

(a) Write a relational algebra expression to extract all the students registered to the course whose name is "Database Design 1". Return the name of the course and the name of the student.

6) Normalization (14 points)

A Car Rental company is using the following *BOOKING* table to store information about its business.

Customer	BookingID	Date	CarID	CarModel
CS001	B001	2014-01-04	Cr001	Volvo CX60
CS002	B002	2014-01-03	Cr001	Volvo CX60
CS002	B003	2014-01-12	Cr001	Volvo CX60
CS002	B004	2014-01-12	Cr002	Volvo V40

Tasks:

a) List all the candidate keys.

b) List all the non-trivial functional dependencies (with *trivial* we mean, e.g., Customer, Date -> Customer).

c) What Normal Form is this table in (1NF, 2NF, 3NF, BCNF)? Why? If some of FFDs violate the criteria for a certain normal form, identify them.

d) If the table is not in BCNF, present an example of a problem that can occur after updating a value in the table.

e) If the table is not in BCNF, present an example of a problem that can occur after removing one or more rows from the table.

f) If the table is not in BCNF, normalize this table such that it fulfills the requirements of BCNF. No extra attributes should be added. List the resulting tables and mark primary keys and foreign keys.

g) Populate the normalized tables with the data in the BOOKING table. You do not need to use SQL: just draw the tables and their content. [Hint: this task helps you to see whether or not your normalization is correct]

7) Transactions (16 points)

In a book database there are two editions of the ”Divine Comedy”, a best seller written by Dante Alighieri. The following program (written in pseudo-code and called T1) has been written to update the price of the two editions in case they differ:

```
SELECT Price INTO $priceEdition1
FROM Book
WHERE TITLE='Divine Comedy' AND Edition='Guelfa';

SELECT Price INTO $priceEdition2
FROM Book
WHERE TITLE='Divine Comedy' AND Edition='Ghibellina';


IF ($priceEdition1 > $priceEdition2)
     UPDATE Book SET Price=$priceEdition1
     WHERE TITLE='Divine Comedy' AND Edition='Ghibellina';
ELSE
     UPDATE Book SET Price=$priceEdition2
     WHERE TITLE='Divine Comedy' AND Edition='Guelfa';
ENDIF
```

Another program, called T2, is available in the system, to apply a discount to the two editions:

```
UPDATE Book SET Price=Price*.9
WHERE TITLE='Divine Comedy' AND Edition='Ghibellina';

UPDATE Book SET Price=Price*.9
WHERE TITLE='Divine Comedy' AND Edition='Guelfa';
```

Tasks:

a) Write the two serial schedules of these transactions and explain what their effect is.

b) Write a non-serial schedule that is serializable, and show why it is serializable.

c) Write a non-serializable schedule, and show why it is not serializable.

d) Explain why the schedule you proposed in the previous task (c) would not be allowed by a strict 2-phase locking protocol (assuming that the programs above are executed as transactions at SERIALIZABLE isolation level).

8) Database terminology (8 points)

a) Briefly explain the concept of physical data independence.

b) Define the concept of candidate key.