Databasesystemer, ITU, Forår 2006

Rasmus Pagh

October 4, 2006

Gruppeprojekt – del 2

This is the second part of the **mandatory** group project. The project is to be carried out in the same groups as announced for the first part of the project. It is allowed to merge groups as long as the total number of students in a group does not exceed 4.
This part of the project should be handed in **by e-mail** to the teaching assistant associated with your group, no later than:

Tuesday October 31, 23.59 PM.

Purpose

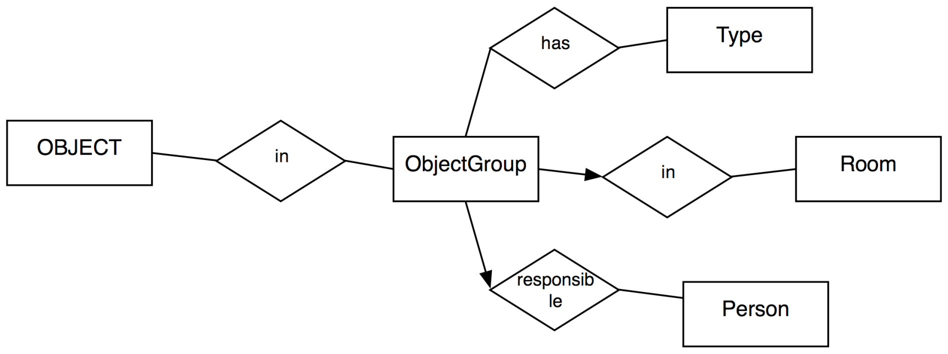
Together with the first part of the project, this part develops competences in database design. It involves the second iteration of the ER design, and further relates to the course goals on:

- Performing normalization to improve a relational data model
- Create database constraints such as referential integrity

Case description – continued

The hand-in concerns the continued development of the new database of IT@. Just after you finished your draft ER diagram, it was decided to integrate an existing database, concerning inventory, with the new system. The aim of the inventory database is to keep track of all objects (e.g., chairs, tables, and computers) that may be moved around IT@. Objects are grouped according to type, according to in what room they are located, and according to who is responsible for the objects in that room. (E.g., all the chairs in a room will typically be of the same type, and hence form a group.)

The database was developed by *Rapid Consulting* (who offered a very advantageous price) and is described in the following. The ER diagram (omitting attributes of entity sets), and the corresponding relation schema are shown below:



```
Object(ObjectID,buyDate,supplier)
ObjectGroup(ObjectGroupID,inRoom,responsible)
Type(TypeID,description)
Room(RoomID,capacity,AVequipment,comments)
Person(PersonID,FirstName,LastName)
ObjectInObjectGroup(ObjectID,ObjectGroupID)
ObjectGroupHasType(ObjectGroupID,TypeID)
```

Having studied *Databasesystemer* you will wonder if this is really in BCNF. If not, you should perform the normalization, and change the ER diagram such that it reflects the normalized design. Also, you may consider whether the ER diagram can be simplified, or improved in some other way.

Redundancy

Since the draft ER diagram was made, you have become aware of some apparent redundancies in the data representation implied by the model:

- Exams of a course are always held exactly two weeks after the last lecture.
- There are a number of competences required by *all* study lines. These are repeated for each study line.
- The exam for a particular course is held in the same room every semester in which the course runs.

You should investigate whether this leads to anomalies in your database schema (i.e., that the relations are not in BCNF).

Your task

For this hand-in you should be able to make a final, complete ER diagram of the whole database. This is to be converted into relations, using the method discussed in the course. You should ensure that the result is in BCNF by changing the ER diagram, if necessary. Furthermore, you should create the relations in Oracle, and write statements for adding all the desired constraints. Optionally, you may choose to do the final modeling in Rational Data Architect, and use its features to generate the DDL statements.

Tip: When entering data, referential integrity constraints can cause problems. A good solution is to defer constraint checking until the end of the transaction performing the insertions. To do this you must declare the referential integrity constraints as `DEFERRABLE`. In particular, you may write `DEFERRABLE INITIALLY DEFERRED` after a referential integrity constaint to obtain this effect. Then constraints need only hold at the end of each transaction, as opposed to after each SQL statement. Note that transactions that violate a constraint will be rolled back, i.e., have no effect. To end a transaction, write `COMMIT` in SQL*Plus.

Creating test data: You may, if you wish, already now create test data for use in connection with the third group hand-in, but this is not a requirement. There is a note on the course home page describing how to create large amounts of test data without having to enter it all manually. We recommend that you use this method, or similar.

What must be handed in

You must hand in a final data model for the system, including:

- **ER diagram**. A graphical data model for the system, using the notation of RG **or** Rational Data Architect. For clarity you may wish to omit attributes from this diagram. You should explain the process behind any changes you made to the ER diagram by *Rapid Consulting*.
- **Description of entities, relationships and their attributes**. A short description in words for all parts of the data model whose meaning is not **completely** obvious.
- **Relational data model**. The relation schemas should be derived from the data model. If you made any choices, e.g. in connection with super/subclasses, you should briefly explain your choice. The relations must be **in BCNF**. If you end up with relations that are not, you should modify the data model such that this is no longer the case. Unless you make special assumptions about the data, you do **not** need to argue that the relations are in BCNF.
- **SQL DDL statements.** The statements for creating the relations and the constraints implied by the ER diagram (or your supplementary documentation of the data model). You are only expected to enforce constraints that are directly supported by SQL – in particular you are **not** expected to write triggers. Primary key constraints, referential integrity constraints, and some check constraints are required.

One the first page, clearly specify the members of your group (if someone dropped the project, don't include him/her). The project should be sent as a **single file in PDF format** to your teaching assistant. Start the subject line of the e-mail with `DBS:`.