UNIVERSITY OF TWENTE.

Pearls of Computer Science

Module 1.1, Study *Technical Computer Science*
Academic year 2016/2017

DATE
September 12, 2016
PARTIAL EDITION

Chapter 3: Pearl 010 — Databases

MODULE COORDINATOR
Maurice van Keulen

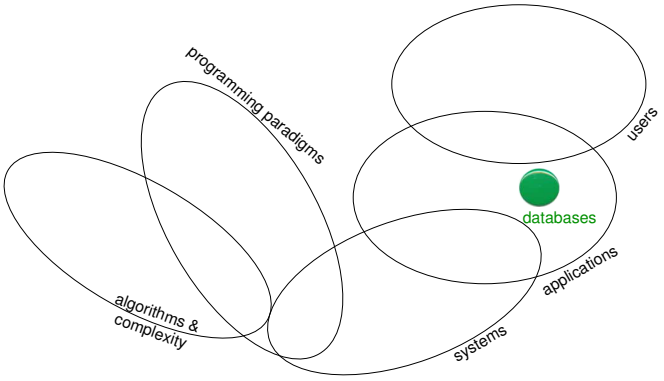
DESIGN TEAM
Maurice van Keulen
Arend Rensink
Pieter-Tjerk de Boer

INVOLVED TEACHERS
Harry Aarts
Rieks op den Akker
Pieter-Tjerk de Boer
Ansgar Fehnker
Marco Gerards
Celine Heijnen
Dirk Heylen
Maurice van Keulen
Frans de Kogel
André Kokkeler
Jan Kuper
Arend Rensink
Andreas Peter
Mannes Poel
Klaas Sikkel
Karen Slotman

FORMER TEACHERS
Pascal van Eck
Anne Remke
Dolf Trieschnigg

TEACHING ASSISTANTS
Kevin Alberts
Dex Bleeker
Tim Blok
Jeroen Bos
René Boschma
Dennis Cai
Sharbel Chmoun
Twan Coenraad
Frans van Dijk
Vincent Dunning
Thijs van Essen
Paula Felix
Noah Goldsmid
Aron van Harten
Joran Honig
Wim Kamerman
Lindsay Kempen
Etienne Khan
Dirk Koelewijn
Mark Kok
Jan-Jaap Korpershoek
Yoep Kortekaas
Remco de Man
Sander Meinderts
Ömer Sakar
Jasper Sustronk
Wouter Timmermans
Jeroen Weener

The green pearl seemed to corrupt with greed and selfishness all those who touched it
Jack Vance — The Green Pearl



Week 3

Pearl 010: Databases

3.1 Introduction

The third pearl of computer science we study is "Databases". The perspective we take is the question "How do you effectively and robustly manage large amounts of data that have to be queried, supplemented, corrected, shared and should not be lost or corrupted?". In the pearl we pay attention to data modelling and the managing, querying and manipulation of data in a relational database using SQL. The ambition is not to go in depth, but to be able to tackle a simple scenario.

Module 1.4 "Data and Information" is shared by BIT and TCS, and requires some knowledge beforehand. BIT students are taught basic knowledge on databases in the BIT module 1.3. This includes database schema design with ER-diagrams, database creation and SQL. This pearl offers similar foreknowledge for the "Data and Information" module for TCS-students.

3.1.1 Global description of the pearl assignment

The practical part of the pearl consists of two parts:

- The *SQL-tutorial*. In this tutorial, the database language SQL is practiced. We use a database with information about movies.
- The *Pearl assignment*. In the real world, much information is managed and shared in "office documents". You will get a spreadsheet with information from a student registration system. For conveniency, you get the same data from the spreadsheets' three tabs as three tables in a database. This structure with three tables is in many aspects a bad data structure, your final product will be much better. The assignment is to design a table structure for this data model, to create these tables in your own database, to convert the current data to the new model and to fill the accompanying tables to finally answer some questions about the data and to execute some manipulations on the data.

You work in groups of two on the assignment; the groups are pre-assigned by the module coordinator.

U	Ma			Di			Wo			Do			Vr		
1	**M**	**tst**	**T∞**	**P**	**lec**	**T∞**	**M**	**self**	**A∞**	**P**	**tst**	**T∞**	M	self	N
2	**M**	**tst**	**T∞**	**P**	**lec**	**T∞**	**M**	**self**	**A∞**	**P**	**tst**	**T∞**	M	self	N
3	**M**	**lec**	**T∞**	**S**	**tut**	**T24**	**M**	**tut**	**T∞**	**P**	**ass2**	**A∞**	M	self	N
4	**M**	**lec**	**T∞**	**S**	**tut**	**T24**	**M**	**tut**	**T∞**	**P**	**ass2**	**A∞**	M	self	N
6	**P**	**lec**	**T∞**	**P**	**ass2**	**A∞**	**P**	**qa**	**T∞**	**M**	**tut**	**T∞**	P	ass2	N
7	**P**	**lec**	**T∞**	**P**	**ass2**	**A∞**	**P**	**ass2**	**T∞**	**M**	**tut**	**T∞**	P	ass2	N
8	**P**	**prac**	**A∞**	P	ass2	N	P	ass2	N	M	self	N	**M**	**tst**	**T∞**
9	**P**	**prac**	**A∞**	P	ass2	N	P	ass2	N	M	self	N	**M**	**tst**	**T∞**

Abbr	Form
ass	Assignment
prac	Practical
lec	Lecture
pfb	Peer feedback
pj	Project
qa	Q&A
self	Self study
tst	Test
tut	Tutorial

Abbr.	Part
P	Pearl
F	Final project
S	Academic skills
M	Math

Abbr	Supervision
T	Teacher
A	Teaching assistent
N	None

Mon 1–4 Math

6–7 The first lecture introduces databases: what is a database, what are they meant for, where they are used, what is special about a database, a bit of history, and a preview of what's to come. Additionally, the most imporant concepts will be explained. Also introduces SQL, the standard computer language for defining, querying and managing data in a relational database.

8–9 Lab session SQL; refer to Section 3.2.2. Make sure the assignment is finished within these two hours.

Tue 1–2 The second lecture continues with the introduction of databases, specifically data modelling and how to design a table structure for a data model.

3–4 Acadamic skills

6–9 Start with working on the pearl assignment; a description of the assignment is located in Section 3.2.3. You get a datamodel, and the pearl assignment is to design a table structure for the saved data, in such a way that all the necessary information (including all exceptional cases) can be represented in all information systems and can be used by end-users. We follow the standard *Entity-Relationship* model (ER). You are converting 'old' data to the new structure and answer information-based questions with it.

Wed 1–4 Math

6 We begin with an hour dedicated to questions. You can ask questions about theory, the study material and the pearl assignment.

7 The 'question hour' will likely not take more than 1 $\frac{1}{2}$ hours. You can use the remaining time to prepare for the exam or to work on the pearl assignment.

8–9 Self study to prepare for the exam and/or work on the pearl assignment.

Thu 1–2 Exam

3–4 You should finish the pearl assignment during this time, as the rest of the week is mostly math.

6–9 Math

Fri 1–4 Self study to prepare for the math exam

6–7 Completing pearl assignment

8–9 Math exam

3.1.2 Study material and tools

The study material for this week can be found on blackboard:

- A general introduction to Databases:
  Modified version of: M. van Keulen, "Gegevensbanken". Chapter VI.4 in Prof.Dr. T.M.A. Bemelmans, Dr.Ir. M. van Keulen, Prof.Dr. R.J. Kusters, Prof.Dr.Ir M. Looijen (eds), "ICT-Zakboek". 3rd edition, 2007. Reed Business. ISBN 978-90-6228-671-3.
- (Multiple SQL tutorials).
- (Documentation on PostgreSQL).

The tools we use:

- PostgreSQL version 9.1.9
- phpPgAdmin version 5.0.3

You don't have to install anything. Refer to Section 3.2.1 for details on how to use both.

3.1.3 Obligatory sessions and deliverables

Mendatory elements of this week:

- Signing off Monday's SQL tutorial.
- Participation in the exam.
- Handing in the pearl assignment.

3.2 Description of the sessions and lab assignments

3.2.1 Tools: Database server and front-end

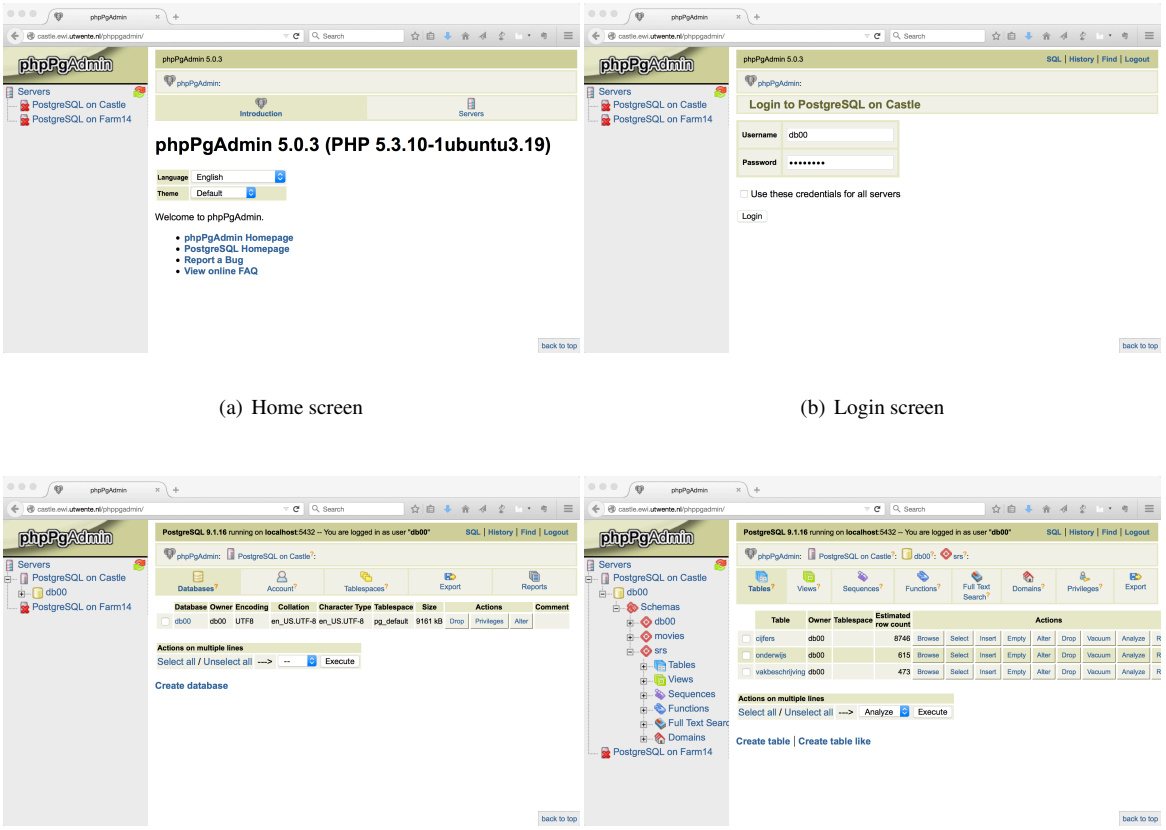
For the SQL practical and the pearl assignment, each group of two students will receive their own database on a database server. You will receive a username and password from the teaching assistants. On Blackboard is a URL to PhpPgAdmin. This is a front-end for the database management system. PhpPgAdmin makes it easier to execute queries and various other tasks on your database. Blackboard also shows what server the database is running on. This is all web-based, so no installation is required.

See Figure 3.1 for some of the screens of PhpPgAdmin. PhpPgAdmin contains more features than you will need, as usual. Below are some instructions on how to reach the two screens that should allow you to finish the entire assignment.

On the home screen, click "PostgreSQL on Castle" in the left column to login. After logging in you will see all databases you can access: a database named 'db*NN*' (*NN* being a number), this is *your* database in which you will need to insert and fill the resulting tables of the pearl assignment, a database named 'movies' that we use for the SQL practical, and 'srs' (meaning "StudentRegistrySystem") which contains the same data as the spreadsheets.

If you click on your database or on "Schema" and then the "SQL" tab, you will reach a screen where you can execute arbitrary SQL queries. You can enter SQL queries by copy pasting from a text editor or by running an entire SQL file. Another useful screen is reached by clicking the '+' button to the left of your database to expand it, then clicking 'Tables'. From this screen you can easily Browse the contents of your tables, Empty them, or Drop (remove) them.

A good way of doing things is to write all SQL queries you use in an SQL script in a text editor and saving them as one or multiple files. These need to be handed in in the end, but it also allows you to easily empty or remove tables and then creating and filling them again with a few copy paste actions.



(a) Home screen

(b) Login screen

(c) Logged in

(d) Database 'Test' expanded and 'Tables' clicked

Figure 3.1: phpPgAdmin screens

If you accidentally messed up the SRS or movies database, these can be restored similarly. On Blackboard you can find a "dump" (SQL file) for both databases. Go to the SQL tab, click "Browse", select the correct dump and execute it. This will restore the database.

3.2.2 SQL practical (Monday 8–9)

Introduction

During the first pearl lecture you received an introduction to SQL. It is important to practice SQL well before starting the pearl assignment. To help with the SQL practical, you will first go through a small SQL tutorial. Students who have experience with SQL may choose whether they will follow this tutorial or skip to Question 3.2 "SQL practical exercises".

SQL-tutorial

☞ **3.1** This tutorial is based on "Learn SQL The Hard Way"[1] by Zed A. Shaw. All individual exercises refer to this tutorial, though some comments have been added. This is because the tutorial uses SQLite while we use PostgreSQL, which means some of the SQL examples from the tutorial will not immediately work and need to be edited first. For example, PostgreSQL uses single quotes ('...') for strings, but the tutorial examples all use double quotes ("...").

[1] `http://sql.learncodethehardway.org`

The tutorial has a "What you should see" section for each Exercise. This will be different for PostgreSQL. Usually this is concerning the results of a query or the contents of a table. In PhpPgAdmin you will immediately see the result of a query in the SQL tab, and tables can be inspected with 'Browse'.

(a) Read the introduction of the SQL tutorial "Introduction: Haters Gonna Hate, Or Why You Still Need SQL" (`http://sql.learncodethehardway.org/book/introduction.html`).
(b) Make Exercise 1: Creating Tables (`http://sql.learncodethehardway.org/book/ex1.html`). The 'What you should see' section here is useless. Check if your table was created with the correct attributes in the "Tables" menu of your db*NN* database.
(c) Make Exercise 2: Creating A Multi-Table Database (`http://sql.learncodethehardway.org/book/ex2.html`).
(d) Make Exercise 3: Inserting Data (`http://sql.learncodethehardway.org/book/ex3.html`). Note: replace the double quotes in this example with single quotes.
(e) Make Exercise 4: Insert Referential Data (`http://sql.learncodethehardway.org/book/ex4.html`).
(f) Make Exercise 5: Selecting Data (`http://sql.learncodethehardway.org/book/ex5.html`). Note: replace the double quotes in this example with single quotes.
(g) Make Exercise 6: Select Across Many Tables (`http://sql.learncodethehardway.org/book/ex6.html`). Note: replace the double quotes in this example with single quotes.
(h) Make Exercise 7: Deleting Data (`http://sql.learncodethehardway.org/book/ex7.html`). Note: replace the double quotes in this example with single quotes.
(i) Make Exercise 8: Deleting Using Other Tables (`http://sql.learncodethehardway.org/book/ex8.html`). Note: replace the double quotes in this example with single quotes.
(j) Make Exercise 9: Updating Data (`http://sql.learncodethehardway.org/book/ex9.html`). Note: replace the double quotes in this example with single quotes.
(k) Make Exercise 10: Updating Complex Data (`http://sql.learncodethehardway.org/book/ex10.html`). Note: replace the double quotes in this example with single quotes.
(l) Make Exercise 12: Destroying And Altering Tables[2] (`http://sql.learncodethehardway.org/book/ex12.html`). Note: ".schema" is SQLite-specific. You can view the structure of a table by clicking the table name in the left column of phpPgAdmin or by expanding the table.
(m) *[Optional]* Exercise 13: Migrating and Evolving Data (`http://sql.learncodethehardway.org/book/ex13.html`). This is a useful exercise to make, because it is about changing the structure of a table while it still contains data. This is an important functionality, however this exercise requires a lot of effort. Only make it if you have enough time left.

It is a good idea to remove the tables created in this tutorial from your db*NN* database before starting the pearl assignment. □

SQL practical exercises

☞ **3.2** The actual SQL practical consists of writing some SQL queries about movies. As preparation it is a good idea to look through the tables, attributes and data to get an idea of how the database is structured. The names are pretty straightforward. The database is named 'movies'. Click on the database and click "Tables" to see all tables. You can view the data in a table with "Browse", and by clicking a table name you can see its attributes.

When you have familiarized yourself with the database, you can start with the exercises below. **Note**: do not forget to make an SQL script containing all solutions as described in Section 3.2.1. The exercises also indicate how many rows the solution should produce. This can be used to check if the solution is roughly correct.

Note: The db*NN* database is the default database, but the tables for these exercises are in the "movies" schema. Because of this, table names should be preceded by "`movies.`", for example "`movies.genre`" for the `genre` table.

(a) Return all movies from the year 2000 (8 movies).

[2] We will skip Exercise 11 because it is about something specific to SQLite and is not part of basic SQL.

(b) Return the name and year of all movies with the genre 'Drama' (143 movies).
(c) Return the name and year of all movies that have both 'Drama' and 'Crime' as their genre (18 movies).
(d) Return the name, role, movie name and year for all actors who have played a role containing 'Tom', sorted by actor name, then year (26 actor-movie combinations). **Tip**: you can use '`LIKE`' to do substring matching; just look up how this works.
(e) Return for all directors the amount of movies they have directed, sorted from many to few movies (165 directors).
(f) Return for each language the amount of movies in that language and the amount of actors that played in these movies (20 languages, so 20 rows. For 'Japanese' there are 6 movies and 64 actors. If you get 78 actors, keep in mind that you want the number of *unique* actors).

□

Finalization

Ask a teaching assistant to sign off your SQL script containing the solutions of the SQL practical. If you are not finished by the end of Monday afternoon, show your work to the teaching assistants anyway. They can advise you on what to do: continue for a bit longer or stop working on the exercises and start with the pearl assignment.

3.2.3 Pearl assignment (Tuesday through Friday)

Introduction

In practice, a lot of information is managed and shared through "office documents". You will be analysing how effective this is by working with it yourself through a fictive case. On Blackboard you will find a spreadsheet named 'DBparel-SRS.xls' containing information on a student registry system. This information was received from an 'educational management office'. On Blackboard is also a second spreadsheet containing the grades of students ('DBparel-SRS-Grades.xls'). These were received from a 'grading administration office'. All information in these files is fictitious. While the names of the courses, students and teachers are real, the rest of the data has been made up.

You can probably think of many reasons why a spreadsheet is not the best option for a data management tool. This does not mean spreadsheets are a bad choice by definition; simply that they are not always a good one either. A carpenter might want to do everything with his trusty hammer, but getting a screw into wood is much easier with a screwdriver.

For this exercise you will design and realize a better system for the student registry system (SRS) based on relational database technology. This will be done in three steps:

1. Design a new database schema by designing and realizing a suitable table structure for a given ER model.
2. Fill the new tables with data.
3. Execute several information processing tasks with the new schema.

The exercise is set up in a way that lets you become more familiar with SQL in addition to the SQL practical, as this will be needed to execute these three steps. The exercise is also split up into two parts: the above three steps are first executed on (a) a simplified version of the system that only considers courses, students and teachers. Afterwards they are executed again on (b) a more complex version of the system that also considers teaching assistants, exams and grades. This second part is optional (bonus exercise).

What to deliver?

What should be handed in on Blackboard for this pearl assignment is:

- Your newly designed table structure (list of tables with for each table the attributes and key(s); this may also be an image).
- *SQL script* that creates and fills your database and executes the information processing tasks.

An SQL script is simply a text file containing all SQL commands in the correct order, meaning they can be executed on an empty database in that order without any errors occurring.

The spreadsheet

There are two spreadsheet files with a total of three sheets.

Education This sheet contains information on what courses are given. It contains the course code, the course name and what study this course primarily belongs to. Each course can occur in multiple rows: one per quarter of a year. For each quarter, the head teacher for that quarter is listed (with teacher id and name).

Course description This sheet provides extra information for the courses. It contains a description of the course as a large piece of text. For ease of use, it lists both the course code and course name.

Grades This sheet is found in the 'DBparel-SRS-Grades.xls' file. Each row contains a student (both student id and name), a course (both course code and name), a quarter and a year.

Note: Course names and descriptions are partly in Dutch and partly in English. Since it doesn't matter for the assignments that you can understand these textfields, we simply left them as is.

Designing a new database schema

To save you some effort, the data from the spreadsheets has been inserted into the 3 tables of the "srs" schema as accurately as possible. The table structure is as follows:

courses course_code, course, description.
For each course, this table contains the course code, the course name and a long description.

education course_code, course, study, teacher_id, teacher, year, quarter.
For each time a course is given, this table contains the course code, course name, quarter and year, what study it belongs to, and the id and name of the head teacher.

grades student_id, student, course_code, course, grade, year, quarter.
For each grade that a student has received, this table contains the student id, student name, course code, course name, quarter and year, and the achieved grade.

This table structure is by far not the best structure for a student registry system. Certain data is duplicated (redundancy) and there are simplifications and incompletenesses (some situations that occur in practice cannot be represented). There are some more deficiencies than the ones shown here.

We will realize a new design for the SRS database. The goal of this new data model is that the student registry system can represent all necessary information (including exceptional cases) for all information systems and end users that need to make use of it. Some requirements for this model are:

- All information needs to be retained, meaning all existing data from the "srs" database needs to be in the new database as well.
- Several useful informational queries and modifications, including exceptional cases, need to be executable (e.g. it needs to know when a teacher has retired).
- It will need to support the assignment of multiple teachers and teaching assistants to a single course.
- With the introduction of the Twente Educational Model (TEM, formerly known as 'TOM'), there are besides 'courses' also 'modules'; these also need to be supported. Modules resemble courses but are larger (more ECTS[3]). Aside from a final grade, a module may also have partial grades for each module part, each with its own name.

[3]From the ECTS, "European Credit Transfer System". One ECTS equals 28 hours of effort. This module gives 15 ECTS. See http://en.wikipedia.org/wiki/European_Credit_Transfer_and_Accumulation_System

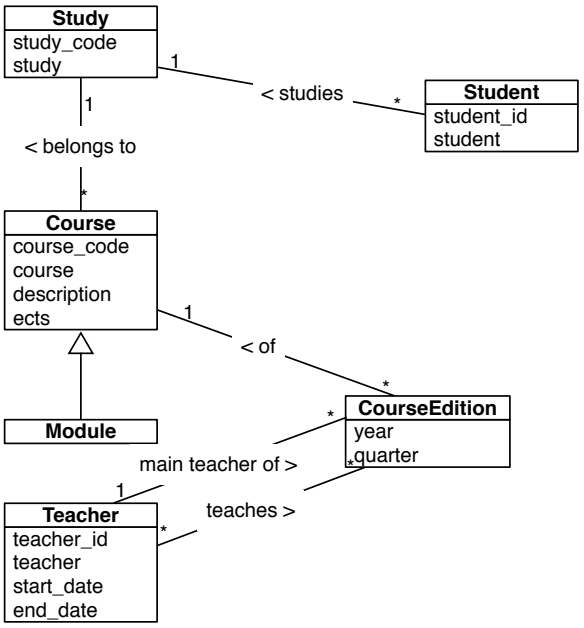


Figure 3.2: New ER model for 'SRS' (only courses, teachers, students and studies)

We will realize the new SRS system in two steps: first we will focus on courses, teachers, students and studies. After this, we will (as a bonus assignment) focus on teaching assistants, exams and grades. Figure 3.2 contains the first part of the exercise as an ER model. Of course, there can be many variations on this model. These are not necessarily wrong, but may have different pros and cons.

The ER model is structured as follows. Each course has, as mentioned earlier, a course code, a course name and a description. It also has an attribute 'ects'. You may assume that all old courses are 5 ECTS and that modules are 15 ETCS. A 'module' is a special type of course. There is also a 'CourseEdition' entity. This represents one edition of a course, i.e., one occurrence of a course, and contains the year and quarter in which this happened. For example, there is one module "Pearls of Computer Science" that has multiple editions: one for the academic year 2013/2014, one for 2014/2015, and so on. Different teachers may teach in each edition (and teachers can teach in multiple courses). For each edition, there is one main teacher (these are called "module coordinators" for modules). Finally, each course belongs to a study (e.g. "Technical Computer Science") and each student is enrolled in one study (this is a simplification).

☞ **3.3** Design a table structure (all tables with their attributes and keys) for the ER model as seen in Figure 3.2. This design needs to be handed in separately as a list of tables with the attributes and key(s) for each table. This may also be handed in as an image.

The `CREATE TABLE` statements that create the tables in your database should be at the start of your SQL script. On phpPgAdmin, under the 'SQL' tab, there is a button labelled "Upload an SQL script" below the text field. This can be used to run your entire SQL script at once, which needs to be able to create your database without errors.

Tip: It is useful to use 'DROP TABLE IF EXISTS' statements before your 'CREATE TABLE' statements. If any of the tables already exist in the database, these will then first be removed. This allows your script to run regardless of the state of your database. □

Filling the new database

Now that we have a better table structure, it can be filled with data. We will begin by transforming the currently existing data. This is also called 'data migration'.

The data from the 'old' student registry system is located in the tables of the "srs" schema. Not all tables, attributes and situations in the new schema will have data available in the old system. Normally, how to fill in the missing data properly is a complex problem. We circumvent this issue here: You can just make up some data of your own.

☞ **3.4** Expand your SQL script with `INSERT` and `UPDATE` statements that take all data from the old tables, transforms it, and inserts it into the new tables. All old data needs to be given a place. Check this by using 'Browse' to see if the number of courses, students, etc. in your new database matches those from the old database.

Tips: Transforming is just a matter of using the correct 'SELECT' queries. The 'INSERT' statement can be used as 'INSERT INTO *table query*'. To avoid duplicate rows, 'DISTINCT' can be used: 'SELECT DISTINCT ...' only produces unique rows. □

☞ **3.5** As said before, not all tables, attributes and situations have data available in the 'old' student registry system. Expand your SQL script with `INSERT` and `UPDATE` statements that fill the empty tables and attributes in the new database with fictional data. Make sure each table contains at least 5 rows. Also make sure that they contain most of the situations that can occur in practice. These are situations such as a module with multiple grades and teaching assistants, a masters course with more than one teacher, an internship, a student that's enrolled in two studies, etc. □

Information manipulation with SQL

To test the new system, we will retrieve and modify some of the information in your new database.

☞ **3.6** Let's start with some basic informational queries. Expand your SQL script with SQL statements for all of these queries. The answer to the first three can be checked by looking at the data in the spreadsheets.

(a) Which students study Technical Computer Science?
(b) Of which courses is "van Keulen" a teacher?
(c) Which teachers teach courses about "databases" (meaning courses with the word "databases" in the description)?
(d) For each quarter, how many courses are given?

□

☞ **3.7** A final, important bit of functionality is the modification and manipulation of the data in the database. Expand your SQL script with SQL statements for all of these modifications.

(a) Choose a random teacher. We will assume that this teacher has retired at the end of 2003. Change the data to reflect this.
(b) Courses are often given again in the following year. Make it so that all courses that were given in quarter 2 of 2003 (69 courses) are copied so that these have a 2004 edition as well. Check whether it indeed added 69 rows.
(c) Now that professor Brinksma is the rector, he won't be giving many courses anymore. In honour of TEM, he still teaches some math lectures, but that is all. Because of this, he was still a teacher in 2003 according to the SRS. He will not be teaching anymore in 2004 though, so edit the resulting data from the previous query so that professor Brinksma's courses are now given by a new teacher: Dr. M. Huisman.[4]

□

[4]Module coordinator of the second module; see `http://wwwhome.ewi.utwente.nl/~marieke/`

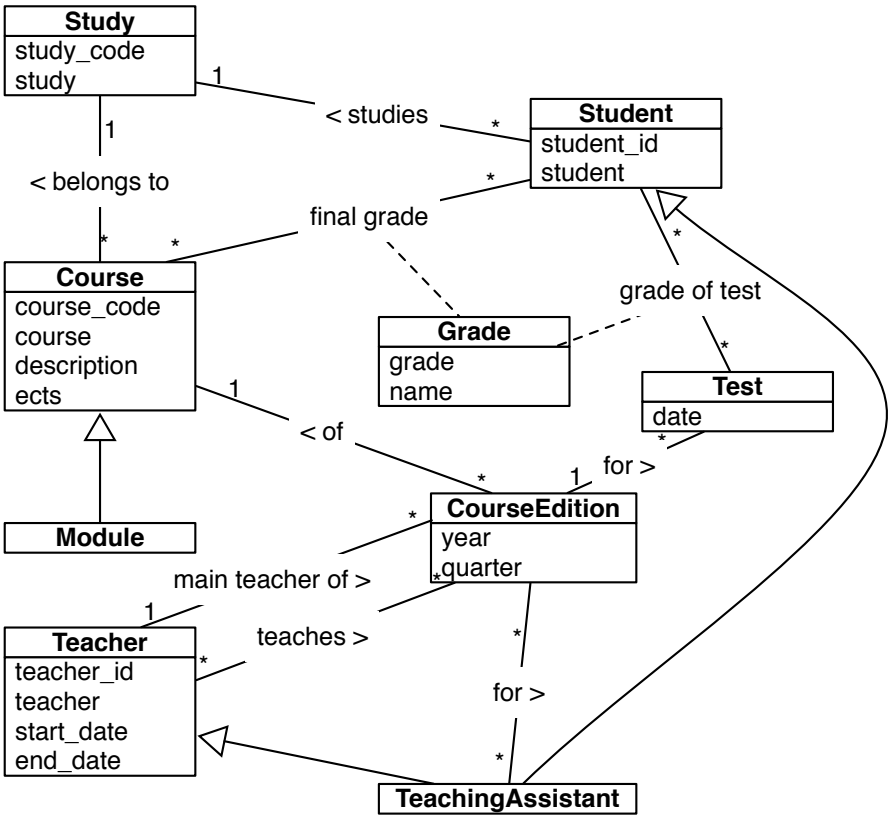


Figure 3.3: New ER model for 'SRS' (with grades and teaching assistants)

Finalization

This is the end of the pearl assignment. This is what needs to be handed in on Blackboard:

- Your newly designed table structure (list of tables with for each table the attributes and key(s); this may also be in an image).
- *SQL script* that creates and fills your database and executes the information processing tasks.

Do you have some time left and want to score bonus points, or would you like to go more in-depth about databases? Try some of the following bonus assignments.

3.2.4 Bonus assignments

The idea behind the bonus assignments is to provide a more in-depth look into the different aspects of the subject. Each of the bonus questions indicates the amount of bonus points that can be scored. You can only get a maximum bonus of 1, so choose the questions that seem most interesting of you. You are allowed to make all of them if you want, but you cannot get more than 1 bonus point in total.

☞ **3.8 Subject**: More complex table design. **Max. Bonus**: 0,5.

Figure 3.3 shows an expansion of the ER model with exams, grades and teaching assistants. The structure is as follows. Each course edition has multiple exams. For a normal course there are two: the exam and the resit; for a module there generally are more: exams for the different module parts and their resits. Students can participate in multiple tests and will receive an exam grade for each of those (we store both the grade and a name for the grade, e.g. "pearl exam databases"). Eventually, these grades together form a final grade

for that course for each student. Finally, the teaching assistants. A teaching assistant is basically a student (hence the IS-A relation with student) who acts as a special kind of teacher (hence the IS-A relation with teacher). A student can be a teaching assistant for multiple course editions.

(a) Design a table structure according to the ER model. Make SQL statements that create these tables.
(b) Fill these tables with data from the 'old' SRS database supplemented with made up data for a minimum of 5 teaching assistantships.

Now that we have tables with data on teaching assistants, exams and grades, we can answer information queries about this data.

(a) How many students have failed a course in quarter 2 of 2003?
(b) Make an SQL query that returns an overview of each course with the course name, year/quarter and average grade of that course for that year/quarter.
(c) Make an SQL query that returns a list of all students, with their name and student id, that have been a teaching assistant in a certain quarter (choose a quarter that actually had any teaching assistants according to your data, otherwise the result will be empty).

□

☞ **3.9 Onderwerp**: SQL puzzles. **Max. Bonus**: 0,5.

For the following "SQL puzzles", the SQL statements we've seen thus far might not be sufficient. Look in the PostgreSQL documentation for the "HAVING" clause of the "SELECT" statement, the possibility of subqueries (a "SELECT" statement nested inside another query), quantifiers such as "EXISTS", and collection operators such as "UNION", "INTERSECT" and "EXCEPT". Only make use of the three tables of the SRS database for these SQL puzzles, so do not use your own table structure.

(a) Make an SQL query that returns all courses (code and name) that have never had a student and that have never been given by a teacher.
(b) Make an SQL query that determines for each student which courses they need to redo (ignore modules here, so determine per student which courses they were graded insufficiently for and no sufficient grade exists).
(c) Make an SQL query that returns, for each year, the course with the lowest average grade and who the teacher for this course is.
(d) Make an SQL query that returns every teacher who, in each year that they have given a course, only gave one course.

□

☞ **3.10 Onderwerp**: Ensuring data integrity. **Max. Bonus**: 0,5.

SQL supplies functionality to ensure data integrity: you can define *constraints* that the database will always have to respect. If any data is changed in a way that violates these constraints, the database will automatically undo (rollback) the changes.

Read the section on constraints in the PostgreSQL documentation: `http://www.postgresql.org/docs/9.1/static/ddl-constraints.html`. Improve your SQL script by adding useful constraints to ensure the integrity of your data. More constraints means safer data. Bonus is determined by how 'safe' you make your database. □

3.3 Example test questions

Apart from exams from previous years, here are some examples of questions that could be asked, with possible answers. In the real exam you will get questions about the same 3 subjects (SQL, database design, databases (general)).

Courses	course_code, course, description
	For every course this table gives the course code, name of the course, and a longer description of the course.
Education	course_code, course, study, teacher_id, teacher, year, quarter.
	For each time that a course is given, this table gives the course code, course name, quarter and year the course was given, which study (programme, such as Computer Science) the course belongs to, and the number and name of the main teacher.
Grades	student_id, student, course_code, course, year, quarter, grade.
	For each grade given to a student, this table has the student number and name of the student, the course code and name of the concerning course, the quarter and year of the exam attempt, and the given grade.

Figure 3.4: Description of the table structure of the SRS-database

3.3.1 Question 1: SQL (20 points)

We once again use the three tables of the SRS-database from the pearl assignment. The table structure is described in Figure 3.4. The query below returns all the teachers and the courses they give.

```
SELECT DISTINCT teacher, course
FROM education;
```

(a) Explain why the `DISTINCT` is necessary in this query.
(b) Modify the query in such a way that it returns only the teachers who have given a course in 2004.

***Note**: The exam will contain more questions about SQL; as it is worth half of the total points.*

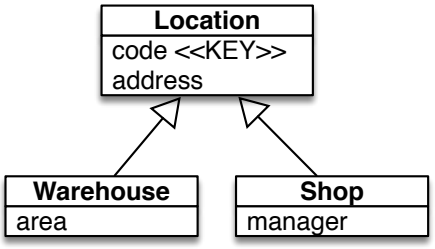
Answers

(a) A teacher can give a course in multiple quarter and years, which could make the query give duplicate results. The 'distinct' makes it so every row is different, i.e. every row is unique.
(b) (every syntactical variation of 'a different study' is accepted)

```
SELECT DISTINCT teacher, course
FROM education
WHERE year <> 2004;
```

3.3.2 Question 2: Database design (10 points)

The data model of an information system of a book store models three entities as below. A database designer has to design a table structure for this ER-model.



(a) Give a possible table structure for this ER-model based on one table per entity. Give your answer in terms of the names of the attribute of these three tables. The types of the attributes may be ignored.

(b) Which tables are used when querying for all adresses of all warehouses? Explain your answer.
(c) Is het possible to store information about these three entities with only one table? Explain your answer.

Answers

(a)
```
Location: code, address, kind --(shop/warehouse)
Store: code, manager
Warehouse: code, area
```

(b) Only 'location' is necessary, as 'kind' could be used to select only the warehouses. If this attribute is not used, both 'location' and 'warehouse' are needed.
(c) Yes. '`Location: code, address, kind, manager, area`', where both manager and area can be NULL.

3.3.3 Question 3: Databases (10 points)

(a) Explain what *concurrency* is.
(b) Which ACID attribute or attributes are connected to the problem of *concurrency*?
(c) The ACID attribute *Durability* means that when the DBMS has labeled a finished transaction (COMMIT), the results will not be lost, *whatever happens*. Give two (as distinctly possible) possibilities a database server should be robust against.

Answers

(a) The possibility/fact/problem that multiple users or applications can read/write/manipulate data in the database at the same time.
(b) Isolation: a transaction cannot be influenced by another, or that one transaction cannot 'see' the effect of another transaction while it is not yet finished. Adding Consistency is not wrong: the integrity of the data can be endangered because of concurrent transactions.
(c) Server crash, harddisk failure, network failures, fire, DoS attack.