Seminar 1 Report

Data Storage Paradigms, IV1351

Viktor Danielsson, Armin Eghtesadi

2025-11-14

Project members:
[Viktor Danielsson, vdani@kth.se]
[Armin Eghtesadi, armineg@kth.se]

Declaration:

By submitting this assignment, it is hereby declared that all group members listed above have contributed to the solution. It is also declared that all project members fully understand all parts of the final solution and can explain it upon request.

It is furthermore declared that the solution below is a contribution by the project members only, and specifically that no part of the solution has been copied from any other source (except for lecture slides at the course IV1351), no part of the solution has been provided by someone not listed as a project member above, and no part of the solution has been generated by a system.

Tips for Report Writing

REMOVE THIS SECTION BEFORE SUBMITTING THE REPORT.

The target audience has exactly the same skills as the author, except they do not know anything at all about the specific application described in the report.

Consider the following:

- **The report must be *centered around the requirements*. Which are they (Introduction), how did you work to meet them (Method), what is the solution that meets them (Result), and how can you be sure they are met (Discussion). This is the IMRaD method.** The requirements on the Introduction, Method, Result and Discussion chapters are described below under each chapter.

- Is spelling and grammar correct? Is spoken language avoided?
- Does the report have a good structure with sections, subsections and paragraphs?
- Is the text clarified with images and/or other figures, and with links to the code in your Git repository? Remember that all figures (images, tables, graphs, code listings, etc) shall be numbered and have a short explaining text.

1 Introduction

This chapter tells *what* are you going to do.

Explain the task and the requirements on the solution. It's important to clearly state the requirements. *Also specify which other student you worked with when solving the tasks, or if you worked alone.*

The task at hand is to design and create a database capable of describing a universites course system. There has to exist a course layout that describes the course code, course name, how many points the course is worth, the minimum amount of students and the maximum amount. A course instance is needded to track the total amount of students registered and in which period the course is given in ex. P1, P2, P3, P4. Each course has many different teaching activites such as lectures, labs, tutorials and seminars. Every teaching activity requires a diffrerent amount of time for the teacher to prepare (multiplication factor) which has to be included in the database. Every course has a varying amount of time for each teaching activity which also has to be considered in the database, aswell as derived data for examination hours and admin hours dependening on the amount of students. The last things are the different departments with their manager (who is also an employee) and the teachers/employees themselves. An employee or teacher works at a department and must have contact details, salary, teaching activites based on skill set, job title and their supervisor/manager. Each teacher can be involved in many teaching activites for multiple different course instances and each course instance can have multiple teachers involved at once. However, a teacher can only be in up to four course instances in a single period.

The final database has to be able to handle all the affermentioned details from the university and be flexible enough to add more teaching activites later on. It must also be possible to add different course versions of the same course layout to to have a 15hp course aswell as a 7.5hp for the same course.

2 Literature Study

This chapter must prove that you collected sufficient knowledge before starting development, instead of just hacking away without knowing how to complete a task. State what you have read and briefly summarize what you have learned.

Some of the knowledge gathered to complete the task was:

- Relational data model: What we are making is a relational database built on the relational data model. It stores its data in a collection of tables all of which contain columns to describe what the table contains. The rows describe an instance of the table. The relationship between tables describes how they are related and what type of cardinality they have.

- Cardinality: A relationship between two tables must express how many of such tables can be associated with the other table. One-to-one would indicate that the their only exists a single table related to another single table. However Many-to-many would instead indicate multiple tables can have a single relationship to multiple other tables. Cardinality is also necessary to define how many of a certain attribute is allowed in a signle instance, can we have a null amount, one to many, zero to many and so on.

- Attribute Domain: For each column or attribute there has to be an assigned domain to define all the values that would be allowed to fill that column for each instance of the table.

- Atomic values: Our database strives to use atomic values in our columns. That means that each value is in its smallest part and not able to be divided into any more smaller parts. So our domain can be atomic if it only allows for a single integer in its column.

- Normalization: A database should not store loads of redundant data, it is space inefficient and requires more computation when you perform operations on the database. So normalization is design decisition that formally defines how normalized the data is in the database. The first normal form 1NF is when every single attribute is atomic. NF2 is the state of the model where every attribute is fully dependent on the entire primary key and 3NF has removed all the transitive functional dependencies in a lossless manner (no information was lost).

- Primary Key: A primary key is a specific attribute choosen specifically to identify each tuple in the relation, or alternativly each row in a table.

- Foreign key: The foreign key is a value in a different table that has to match one of the values in the primary key of that other table. This allows for references between tables.

There is of course more, but this was a short summary of what has been taught and what is needed to understand the task.

3 Method

This chapter tells *how* you solved the task.

Explain how you worked when solving the tasks and how you evaluated that your solution met the requirements. *Do not explain your solution and do not refer to code*,

that belongs to the *Result* chapter. More specific instructions for the content can be found under each task on the Project page in Canvas.

The method of creating a database starts by creating a miniworld of reality that describes how different entities interact and exchange information Then the conceptual model is transformed into a logical and physical model which describes how the miniworld would be represented in a database. The eleven steps necessary to formulate the correct logical and physical model were:

Two trivial steps of creating tables and rows based on the conceptual model.

Then assigning the domain for each attribute.

"UNIQUE" attributes will be marked in the model.

Define primary keys for each table which is definied as a strong entity in the model. Typically surrogate key.

Then define the relationships between tables, one-to-many, one-to-one, many-to-many

Generate pk and fk for multivalued attribute tables

Normalize the model

Final step of making the model is verifying all the operations.

After each step is completed the model is complete and the tool used to make the model Astah professional is able to generate a rough SQL dump. Finalizing the database is done by adding UNIQUE to the correct attributes and adding foreign key constraints ON DELETE.

4 Result

The first two steps generated a very simple model with just the first tables and attributes. Then we assign the domain for each attribute.

Attributes in person: phone number has VARCHAR 13 to allow a phone number with country code that is larger than 2 digits. Adress, last_name and first_name all use VARCHAR 200 to allow for large names but without risks of massive values. personal_number is unique because two people are not allowed to have the same personal number.

Attributes for employee: employment_id is unqiue to an employee and gets the VARCHAR 200 domain. What format the university would use is not known so a variety of formats is allowed by the database. Then an employee has a manager which is also an employee which is why it is allowed to be null. max_allowed_courses is an int, salary is an int. job_title of an employee is a foreign key to the table containing all job_titles.

Attributes for course_layout: We can not not the corse_code so we allow for 200 charachter

VARCHAR so the university can use whatever format it chooses. Course_name gets the same treatment.

attribute for planned_acitivty: int for hours

An employee has a manager_id which is also a employee_id and for the department it is also a employee_id which is the managers_id.

Each attribute such as person_number, employment_id and all the others marked with UNIQUE got labels describing that property.

Each attribute was chosen to be atomic and also relevant to how a university would use that value. most values used as an id is for business logic for the university and is not to be confused by the surrogate id which identifies the tuples.

Finding the strong entities was done by looking at the conceptual model and searching for an entity which is not dependant on another entity to exist. Such entites were found to be person, course layout and department. They were then given a surrogate key to define their primary key. **This chapter explains *the result* of what you did.**

The report must show that you have done the work yourself and that you have understood what you have done, both of these goals are met by carefully explaining your solution here in the result chapter, and proving that it meets the requirements. *State each requirement that is met* and explain *how you met it.* Also include links to your code in your Git repository, and include also diagrams, see Figure , and other figures to illustrate your reasoning. All figures must be referenced in the text. Ask yourself if the solution is clearly explained, and if the reader will understand the application. What would you yourself want to know if you read about the application, is that included in the report? More specific instructions for the content can be found under each task on the Project page in Canvas.

Foreign key constraint

5 Discussion

The employee attribute skill set is tricky because multiple employees can share the same skill and we want to ensure the values are atomic so we also do not want to store an array of multiple skills for a single employee. So multiple employees can have the same skill and a single employee has multiple skills. Then solved with a many-to-many relation.

For our many-to-many relations such as study period and employee skillset we use the cross-reference table. It could be done with less tables with a table with multivalued attributes but it is valuable to decrease the amount of duplicate data. We then force the values to be unique because otherwise they would be able to be duplicate which would remove the benefit of having a cross-reference table.

When finding strong entites the department entity is perculiar because the manager_id is dependent on a employee existing. By allowing for a null manager we can define the department as a strong entity because it is no longer dependant on the employee table. This is allowed because null is not accounted for when checking the column for uniqueness.

When considering the hp for a course there could be the same course with the same course id but with a different amount of hp. Our solution is to divide course and course layout in to two tables. The course only contains code and name with a surrogate key. Then layout has min and max students, hp, a version and the course_id as a foreign key with a surrogate key as primary key. The relation between the two is non-identifying but one could argue that the layout should be identified by its course. Another way of solving it would be to combine the two tables and have the version as a primary key togehter with course_id but that solution was not used because the course_code and course_name are not related to version in the primary key.

This chapter *analysis* the result presented in the previous section.

Evaluate your solution according to the assessment criteria found in the assessment-criteria documents, which are found under the bullet *In the Discussion chapter of your report...*, under each task on the Project page in Canvas. You do not have to cover all specified criteria.

6 Comments About the Course

The time invested to the project were six, five hour sessions so in total 30 hours to complete the project.