Database Management Systems

(COP 5725)

Dr. Markus Schneider

Department of Computer & Information Science & Engineering (CISE)

Group Project Specification – Spring 2018

Last update: December 27, 2017

Contents

1 Overview of the Database Software Project

The class is accompanied by a semester-long database software group project. Its main goal is to apply and practice the theoretically learned concepts in class in a professional and commercial database environment by means of the design and implementation of a *web-based database application program*. Section 1.1 discusses the main project objectives. Section 1.2 provides a general description of the project and explains the expectations.

1.1 Project Objectives

An important objective of the database project is to transfer the database concepts learned in class, such as the conceptual database design with the Entity-Relationship (ER) Model, the transformation of an ER diagram into a relational database schema with a provided algorithm, and the application of the synthesis algorithm for 3NF normalization, step by step into practice by deploying the professional and commercial database system Oracle. While in homework assignments students learn the formulation of SQL queries in an *ad hoc* mode, that is, by entering SQL queries with the keyboard of a computer, in this project students learn how to embed SQL queries into database application programs. In particular, in this project, the database application will include a web-based user interface as its front-end and a supporting Oracle database as its back end. In this sense, students will obtain a real hands-on database experience that will enable them to work later in industry in the database field. Further, by working in groups, students will learn to argue, discuss, compromise, write technical documents, and solve arising social conflict situations in the group at a professional level.

1.2 General Description and Project Activities in Detail

The project includes the following chronological activities spread over the entire semester:

1. Identify an application area for which a DBMS may prove beneficial to store the data it processes. Consider factors such as the need to store and query large data volumes, support multiple users, provide concurrent access, maintain consistency, etc.
2. Find appropriate real world data that match your application and can be used to populate your database with *at least 100,000 records*. Understand and analyze the available data. Find problems in the data such as missing data and inconsistent data.
3. Determine the main functionalities and operations of the database application. Think about the various requirements of the user of your application and the various data attributes that need to be stored and later queried.
4. Application development:
   i. Database development
      - Model the data to be stored in the database, i.e., identify the various entities, relationships, constraints, etc. by creating an ER diagram.
      - Transform the ER diagram into a relational database schema.
      - Design, normalize, and perfect the relational database schema.
      - Transform and upload the real word data according to the database schema into the database by using SQL queries, spreadsheets, CSV files, and/or stand-alone programs written in Java, C, C++, etc.

- Design the SQL queries that will be embedded into the application program.

ii. User-Interface (UI) development

- Design the web interface for the application by considering the various "screens" and the "flow of control" of your application. For example, in a photo manager application you might start with a user login screen, then a web page to display the user's photo in a gallery, then another one to display a specific picture with additional descriptive information, a search page, logout page, etc. The whole user interface can be considered a directed graph in which the vertices are the web pages and the edges represent links (URLs) to other web pages.
- Select web-based technologies for designing web-based user interfaces, e.g., PHP and Ruby on Rails.
- Implement the web interface and write supporting code to embed the designed SQL queries to retrieve data from the DBMS.

5. Test your database application software and check if it works as desired.

2 Organizational Issues

2.1 Topic of the Group Project

Each group selects an application topic on its own. The instructor will *not* provide a topic. This allows a group to be creative and follow its interests. Note that it is not sufficient to identify an interesting topic that is worthwhile to be supported by a database system. A group will also have to find real world data that support the selected application.

2.2 Project Group Size and Formation

Four students will form a group. If the total number of students in class should not be divisible by four, we will have one or two groups with three students or one group with five students. If the class size should be low, three students per group are possible. Each group will select tools from the Internet (e.g., email, Skype) for group communication.

Since we can assume that the students in class do not know each other, the instructor will randomly form groups.

2.3 Grading

Each group will have to submit four specific project deliverables. Detailed instructions for each deliverable will be provided in Section 4. Each group has to turn in a common, single solution document as a deliverable. All members of a group will get the same grade. In exceptional cases, for example, if it turns out that a group member has not adequately contributed to the group's efforts and therefore harmed the group, the instructor will take the right to assign a different and adequate grade to such a group member. General information about the grading of the four project deliverables can be found in the syllabus.

2.4 “100,000 Tuple” Rule

A group’s database must store at least 100,000 tuples (records) as the sum of all records stored in all database tables.

2.5 Programming Environment

As for the user interface, it is each group’s choice to use any high-level web programming language. Some options are Ruby on Rails, PHP, .NET, JSP, and Javascript. If group members have not designed a webpage before, now is the right time to get started and learn how to perform this task. Please note the teaching of web-based programming technologies is beyond the scope of this class. A student who is not familiar with web programming languages will have to get knowledge of at least one them by self-study.

As for the database interface, we will use the CISE Oracle database server. In order to be able to connect to the Oracle server, each student needs to have a current CISE account as well as an Oracle account (they are two different accounts with different passwords). Information how to obtain a new Oracle account or how to renew an existing Oracle account can be found on the CISE Oracle Database Help web page. Oracle clients for connecting to the server are available on Linux workstations and Windows PCs in the CISE computer labs. One can use ODBC, JDBC, or other connectivity protocols to establish the database connection between the web-based user interface and the database. Additional information on how to achieve this can be found on the CISE Oracle Database Help web page. Simple coding examples for Java using JDBC, PHP using OCI8, Perl using DBD::Oracle, and Ruby using DBI are provided. The CISE help pages also contain information how to remotely access CISE Oracle. The project database must run on Oracle and use the *orcl* instance running on the CISE Oracle database server. To test queries on the database before their integration into the webpages, one can use a client such as SQL*Plus (command line) or Oracle SQL Developer (graphical).

2.6 Working as a Group

Working as a group is not always easy. Miscommunication, lack of experience, and social conflicts can impede a group’s success. It is up to each individual group member to take a professional attitude and contribute to the group’s success. This requires reliability as well as the willingness and ability to communicate and work hard. Social problems such as conflicts between group members should be discussed with the instructor who will then take the role of an intermediator and aim to bring the group back to work.

Often a group distributes a task into subtasks, one for each group member. The idea is then that each group member works on the assigned subtask and produces the desired sub-result. All sub-results are then merged into the final result. However, experience shows that sometimes group members do not perform their assign task, do not inform the other group members about it, and do not deliver the expected sub-result. This behavior can put the whole group effort into question. Often the remaining group members have then to compensate the detected gap in a very short amount of time at the end of the semester. In the worst case, the group can fail. The recommendation is therefore not to assign only one group member to a task but at least two members. For example, let us assume a group has four members and identified four tasks for the implementation phase. Then an assignment of group members to tasks could, for example, be as follows:

	Task 1	Task 2	Task 3	Task 4
Group member	1, 2	2, 3	3, 4	4, 1

The red numbers indicate those group members who are *mainly* responsible for a particular task. The blue numbers indicate group members who support the group members with red numbers. The two group members assigned to a task talk to each other, solve problems together, and implement the task together. This strategy avoids that a group member has to work alone and does not make progress. Similar other strategies are, of course, conceivable too.

3 Examples of Project Ideas

This section provides a few proposals of project ideas.

1. *Coursework Management Tool*: Develop a system to store course related information in a university. An example for this is UF's eLearning or Canvas interface. This system should store data about users (students, faculty) and coursework (course, semester, year, student grades, etc.). Users should be able to login and enter and view grades, manage the course material, perform updates on profile or grades, import and export data, get an analysis of their grade data, see their overall performance in comparison to the average class performance, etc.

   Examples: UF eLearning, CourseWork@Wikipedia

2. *Crime Analysis*: Use real world crime data to find out which kind of crime is committed most and in which area of a city and in which period of time one should not be present.

   Examples: -

3. *Genealogy database*: Develop a database of information about people and their ancestors, with additional information such as birth place, education, etc., and construct the family tree.

   Examples: FamilySearch.org, Ancestory.com

4. *Shopping and eCommerce application*: Develop a website to help merchants sell products and customers buy them. The system could provide functionality to store user and product information, offer user roles, and provide a shopping cart. Additional features possible could be, for example, ranking and reviews of users and products as well as trend analysis to find the most popular products over time.

   Examples: OpenCart, eBay, Amazon

5. Others: Of course, there are a whole amount of other ideas such as a census database, bank accounts, NBA data, car rentals, auto insurance, or "map driven" applications such as a hurricane database, a real-time location tracker ("spatial-tweets"!), and a campus map.

4 Project Deliverables and Final Project Demonstration

The database group project consists of five phases. Each of the first three phases leads to a project deliverable that summarizes the group's result of each such phase in a PDF document that has to be submitted at a given deadline for grading. The fourth phase results in a web-based database application software. The fifth phase is the project software demonstration that will also be graded.

4.1 Phase I: Requirements Analysis

In the first phase, the group's task as application developer and database designer is to propose and understand an appropriate project topic, identify its main data management needs, explore and motivate its potential for interesting queries, and analyze the needed user functionality. The group members should ask themselves questions such as

- What are the main functions that the web-based user interface should provide?
- How do the different functions work together? Sometimes there are dependencies between different functions.
- Which real world data are needed to support the functions identified before?
- Can such real world data be found in the Internet?
- What (colloquial) queries are important for the application?
- Which public domain and/or proprietary software is needed to perform the task? (The database system used must be CISE Oracle.)

The first project deliverable is supposed to be a detailed document (PDF file) that presents a clear and structured description and motivation of the selected project topic and its requirements that the group thinks the software solution should later fulfil. This means that a group has to carefully deliberate on the requirements and functions and precisely describe them in their document.

The focus of this project is supposed to be on the database part and not so much on the application part. This means that a group should not design and implement highly sophisticated main memory algorithms but focus on database queries that evaluate large volumes of stored data. Of course, the application part must be highly functional, and the different user functions must cooperate nicely together. However, a fancy layout design of the user interface is not required but appreciated.

It is important that each group demonstrates in their deliverable that their application would really benefit from database support and that *new information* (such as *trends*) *can be derived from the stored data*. A simple retrieval of data from the database (that is, search) or the pure connection of different tables (that is, joins) are not sufficient. As an example, let us assume that a group selects a sales application as their project topic and stores many sales numbers in their database. Of course, one can search for sales data of interest in the database and display them in the user interface. But searching only identifies an interesting subset of all data stored in the database. DBMS are specialized for search tasks, and the respective SQL queries are relatively simply structured. This project aims at more interesting queries that derive new information which is not explicitly stored in the database but can be derived from the data in the database by computations. In the sales application, examples of more interesting queries are:

- What were the total monthly sales in 2012?
- Which item was sold most (so that we have to pre-order more of it)?
- Which item was a slow seller (and should therefore be removed from the inventory since it

only wastes storage space)?

- How many items does the store have in stock?
- How have the total sales (in general, of product X, of products X, Y, and Z) developed in the last *n* months? Can a trend be recognized? For example, it could be that the sales are low in the summer months so that advertising efforts could be put in place in these months.
- When were the most successful or most lossy *m* months in the last *n* months?
- Provide a ranking of all customers based on the decreasing amount of money they have spent in the store in last three months.

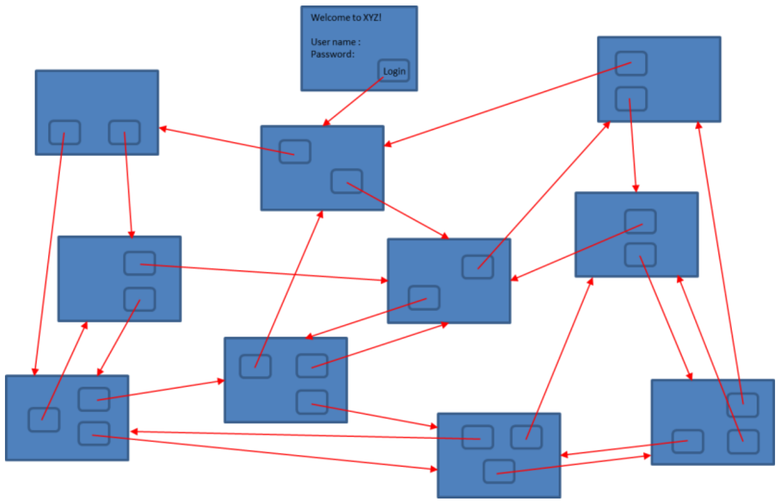
The answers to all these queries are not directly stored in the sales data. But they can be derived, that is, computed by (complex) database queries. Therefore, *each group has to list a number of (at least five) database queries in their deliverable in order to show that their application has the potential to have such interesting database queries*.

Only the first submission of a group's first deliverable will be graded. If the first submission should so dissatisfactory that the group cannot move on with the next phase, a revision will be requested from the group that will not be considered for (re)grading. This means, the more effort a group puts into the first submission, the lower the probability is that a revision is needed, and the more time the group will have for their next deliverable. It is important to note that it is *not* the task of this phase to determine and describe solutions to the requirements.

4.2 Phase II: Entity-Relationship Diagram Design and User Interface Design

Based on the requirements analysis of Phase I, the goal of the second phase is to describe the overall conceptual design of and solution approach to a group's application. This incorporates the two aspects of user interface design and conceptual database design.

As for the *user interface design*, the task is to devise the set of web pages that is needed to convey the promised functionality of the application to the user at the user interface. This requires a clear description about the flows of action and web pages the user can expect. It starts with the entry or welcome web page and spreads out into a number of successive web pages that are appropriately linked with each other. The whole web site can be regarded as a graph where the nodes are the web pages and the edges are the URLs connecting the web pages. This is shown in the following figure.



The figure does not indicate user actions that are activated and performed on the same web page. An example would be to query sales data from the database by means of a user function in a window and show them as a graph presentation in the same window. At the end of the user interface design, its web pages, components, the control flow, and user functions should be conceptually understood and mature so that the application logic is clear and the implementation of the user interface can be performed in a next step.

As for the *conceptual database design*, the task is to leverage the Entity-Relationship (ER) Model that we learn in class. The task is to identify the important entity sets, relationship sets (including cardinalities), and attributes, which are relevant and have later to be stored in the database, in an ER diagram. But an ER diagram alone will not be sufficient. Each group should provide a motivation for their design and explain the important concepts of their diagram in sufficient detail.

A group's result with respect to the user interface design and the conceptual database design have to be put into the second deliverable. For drawing the ER diagram, any suitable tool can be used. Any kind of programming or HTML web page construction is *not* required. Drawings by hand, as long as they are readable, are allowed. Mixing different tools together is allowed too. The emphasis is *not* on producing a high gloss pamphlet with nice pictures. But it is important that each group provides an overview and introduction of their ideas first before it describes the user interface design and the conceptual database design.

Only the first submission of a group's second deliverable will be graded. If the first submission should so dissatisfactory that the group cannot move on with the next phase, a revision will be requested from the group that will not be considered for (re)grading. This means, the more effort a group puts into the first submission, the lower the probability is that a revision is needed, and the more time the group will have for their next deliverable. It is important to note that it is *not* the task of this phase to determine the database schema of the group's solution.

4.3 Phase III: Database Schema Construction

The third phase applies the transformation algorithm presented in class to the group's ER diagram and results in a relational database schema. Relation schemas are presented in the form $R(A_1 : D_1, \ldots, A_n : D_n)$ in a first step where R is a table name, the A_i's are attribute names, and the D_i's are domains or data types. In a second step, these relation schemas are transferred to SQL table schemas by using the *create table* command. The SQL table schemas should be enhanced by all needed and desired integrity constraints so that they can be directly used in Oracle. The SQL database schema with additional explanations (if needed) represent the third project deliverable. Original screen snapshots from the Oracle DBMS are required that show all *create table* commands and the created empty tables. Each group should not forget to include their (perhaps modified) ER diagram at the beginning of the document that is the input of the transformation algorithm.

An important step of database design, called *normalization*, is missing at this point. The normalization process eliminates possible redundancies, inconsistencies, and anomalies of the database schema and thus improves its quality. But since the normalization process requires a deep understanding of *relational database design theory* and since this theory will not have been taught when this deliverable has to be submitted, the normalization step will be skipped. Groups that are willing to rearrange their database schema at a later time may perform the normalization process. However, this will require a new upload of all data according to the modified database schema.

Only the first submission of a group’s third deliverable will be graded. If the first submission should so dissatisfactory that the group cannot move on with the next phase, a revision will be requested from the group that will not be considered for (re)grading. This means, the more effort a group puts into the first submission, the lower the probability is that a revision is needed, and the more time the group will have for the implementation of their application software. It is important to note that it is *not* the task of this phase to fill the database schema with data.

4.4 Phase IV: Project Software Implementation

At this point, each group should have a rather clear picture how their software will look like, without having performed any implementation yet, and which tools and programming languages will be deployed in the project. The clearer and more detailed a group has performed the overall design, the simpler the implementation will be. The implementation includes the following main tasks: (i) implementation of the user interface, (2) pre-processing and cleaning of the real-world data found for the application, (iii) upload of the pre-processed real-world data into the database, (iv) establishing the connection between Oracle and the application program, and (v) formulating the SQL queries and embedding them into the application code. Establishing the connection between the database and the user interface as soon as possible is very important since it enables the group to send data from the user interface to the database and retrieve data from the database to the user interface. For this purpose, the user interface could be a single welcome screen that asks for a username and a password. Both data are sent to the database with a correspondingly structured table. A lookup in the database checks whether the username/password combination exists and sends a corresponding result back to the user interface that displays either the message “Username/password is valid!” or “Username/password is not valid!”.

During the implementation phase, two *checkpoints* will be used to determine the progress with respect to each group's project implementation. These checkpoints are an ungraded service to the groups and are supposed to avoid that groups start too late with the implementation in the semester and then get overwhelmed by the implementation task due to its complexity and required time involvement. The instructor will informally meet online with each group and have a relaxed conversation about the group's project implementation. The duration of such a meeting is 15 minutes. All group members have to be present.

The first checkpoint especially checks whether each group has had a successful and promising start. The instructor will especially ask the following questions:

- Has the group managed to established the connection between the user interface and the database for the technologies chosen by the group?
- Has the group been able to find real world data sources that fit to and can be used for the group's project? If not, have "meaningful" data been generated?
- Have these data already been pre-processed so that they can be bulkloaded into the database?
- Has the database schema been created in Oracle?
- Have the pre-processed data been bulkloaded into the database according to the database schema?
- Has the minimum of 100,000 tuples been stored in the database?

- Have parts of the user interface been implemented?

The second checkpoint especially checks whether each group has made considerable progress since the first checkpoint and implemented the main parts of its project. Some main questions are:

- To which extent has the database part been finalized?
- To which extent have the data been loaded into the database?
- To which extent has the user interface been implemented and is functional?
- To which extent have the needed SQL queries been designed and embedded into the code of the user interface?
- To which extent has the whole software system been tested?

An extensive testing of the final software product is, of course, inevitable.

4.5 Phase V: Project Software Demonstration

At the demonstration day(s), which will be at the end of the semester, groups have to present their software system. The groups have to arrange an appointment with the instructor. Demonstrations will last 30 minutes and will be conducted online with a web-based presentation tool that each group selects on its own. The atmosphere during a project demonstration is usually very relaxed and informal. Each group should consider the following aspects:

1. Each project demo lasts 30 minutes. In the first 10 to 15 minutes each group will provide a presentation of their software system. In the next 10 to 15 minutes the instructor will ask questions about the software system. In the last 5 minutes, the instructor will determine the groups' project demo grade.
2. The group presentation in the first 10 to 15 minutes is held by one or several group members and will provide and explain the *highlights* of the group's project functionality and implementation.
   - The following aspects do *not* belong to the highlights: (1) input masks and input procedures, (2) correctness tests for input data, (3) simple search procedures for data. Especially searching can be performed by very simple SQL queries and is therefore not so much of interest.
   - The following aspects belong to the highlights: (1) Interesting functionality at the user interface that allows the user to analyze the data and leads to new conclusions of and insight into the data stored in the database (for example, trends). (2) Complex analytical procedures for data that lead to interesting conclusions of and insight into the data stored in the database.
3. In the question-and-answer period in the next 10 to 15 minutes, the instructor will ask questions to all aspects of the group's software with respect to both its functionality and its implementation. The group has to answer these questions immediately. A typical question is: “Show me and explain the SQL query that does ...”.
4. In the last 5 minutes, the instructor will evaluate the quality of the project software, the presentation, and the answers, and determine the project demo grade. The grade is not negotiable with the group members. Therefore, discussions about the grade are not allowed.

5. Slide presentations are not allowed during the group's presentation.
6. A documentation of the software is not required.