CIS 4301 UF Online
Information and Database Management Systems I

Department of Computer & Information Science & Engineering (CISE)

Group Project Specification

Contents

1 Overview of the Database Software Project

The database course 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*. This skill is nowadays asked for in industry. Most websites we use in everyday life have the structure of running a web-based application as a frontend in a web browser and being supported by a database as its backend for storing and retrieving user-specific application data and answering user-specific database queries. Section 1.1 discusses the main project objectives. Section 1.2 describes the overall and application-independent topic of the group project. Section 1.3 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 this project, the database application will particularly 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 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 Topic of the Group Project

This group project has the *overall* and *application-independent* topic of *trend analysis and visualization*. But the actual application and the real-world data set supporting the application must be selected by each group on their own. What does this mean? Databases can store large amounts of data. But this does not mean that it is easy to make sense out of them and to make them valuable for an application. For example, we can store and then retrieve millions of records of customers' sales purchases. But this does not mean that we can directly derive consumer behavior and other interesting information from them as it is important for many companies that sell products. Disciplines such as data science, data mining, machine learning, and statistics provide sophisticated tools for this purpose. In our project, we aim to go into the direction of data analysis by deploying pure database technology. *We focus on the identification, analysis, interpretation, and visualization of past trends in historical real-world data sets by sophisticated database queries*. This explicitly excludes the prediction of future trends. For example, we can group products by selected categories such as clothes, electrical goods, food and analyze how customers subdivided into age groups have made purchases over time regarding the different product categories. This leads to several trends that can be visualized as overlaid linear graphs *over time*. That is, continuous time always forms the *x*-axis of a trend visualization. The reason is that a *trend describes the evolution or change of data over time*. For example, we could be interested in the purchase trends of electronic goods from 2002 and 2018 in the context of different customer age groups. The goal could be, for example, to determine which age groups are more open-minded to electronics and whether the assumption that these are the younger age groups can be confirmed or not. Trends are *not* stored in the database but can be *computed* from the data stored in it. Section 3 will provide more details about trends.

The actual application for trend analysis and visualization and the actual data sets that support the application must be selected by each group on its own. The instructor will neither provide an application nor a data set. This allows each group to be creative and follow its interests. Note that it is insufficient 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 sets that support the selected application.

1.3 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 the project topic of *trend analysis and visualization* (see Sections 2.1 and 3) might be useful and 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 500,000*[1] *records*. Understand and analyze the available data. Find problems in the data such as missing data and inconsistent data and provide solutions to fix these problems.
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

[1] More precisely, the lower bound of 500,000 records means that in the final database in Oracle, the sum of the tuples (records) of all tables must be equal to or exceed 500,000.

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 (see below).

- Select web-based technologies for designing web-based user interfaces, e.g., PHP or 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

Section 2.1 defines the project group size and explains how project groups are formed. Section 2.2 describes the grading process around the project. Section 2.3 specifies the rule that a project database in Oracle must have at least 500,000 tuples. Section 2.4 emphasizes that each group has the free choice to select their programming environment. The only constant is the mandatory use of the CISE Oracle database server. Section 2.5 gives pieces of advice on how to work in and be successful as a group.

2.1 Project Group Size and Formation

Four students will, in general, form a group. If the total number of students in our course should not be divisible by four, some groups with five students are possible. Each group will select tools from the Internet (e.g., email, Skype, Zoom) for group communication. Extensive and regular group communication is the key to a group's success in the project.

You are free to choose your own project members. You are also allowed to form incomplete groups that only contain 2 or 3 students. The instructor will then fill them up randomly. If you should be unable to find a group, do nothing. The instructor will then assign you randomly to a group. The instructor will communicate further details about the group formation process in emails.

2.2 Grading

Each group will have to submit four specific project deliverables. Detailed instructions for each deliverable will be provided in Section 4. Each group must 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. In the worst case, this grade can be 0. General information about the grading of the four project deliverables can be found in the syllabus.

2.3 "500,000 Tuple" Rule

A group's database must store at least 500,000 tuples (records) as the sum of all records stored in all database tables. The tuples must be derived from real-world data sources exclusively and may not be constructed artificially.

2.4 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 of them by self-study.

As for the database interface, we will *exclusively* use the CISE Oracle database server. 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). Note that off-campus access to CISE Oracle is only possible if you have installed and activated the Gatorlink VPN client on your computer as the first step.

2.5 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 assigned 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 for 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 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 are supposed to talk to each other and conceptually solve the problems of the task together. But only the red group member implements the solution to the task. This strategy prevents a red group member from working alone and not making progress. It implements a control function for the group. Other strategies are, of course, conceivable too.

3 A More Detailed Description of the Overall Project Topic

Section 3.1 uses the well-known example of temperature curves to explain the term *trend* and *trend query* in the sense of this database project. Section 3.2 explains the nature of a *complex trend query*.

3.1 What is a Trend? What is a Trend Query?

The overall topic of this database project is to perform *trend analysis by means of database queries*. More specifically, *we focus on the identification, analysis, interpretation, and visualization of past trends in historical real-world data sets by sophisticated database queries*. In general, a *trend describes the evolution or change of data over time*. In this project we are *not* interested in the prediction of future trends.

In the following, very simple application example, we assume a large database that is the result of *daily* sensor measurements over 20 years and that contains triples of the kind (*ti*, *lo*, *te*) where *ti* is a timestamp (e.g., 2020-01-19 15:14:07), *lo* is a location (latitude, longitude, e.g., 40° 26′ 46″ N 79° 58′ 56″ W) and *te* is the measured temperature (e.g., 78.8° F) at the sensor location.

We could now get the idea to fix the location/sensor, call it l_1, and ask for the daily temperatures at l_1 in the years 2010 to 2011. The resulting visualization shows a trend and could look as follows:

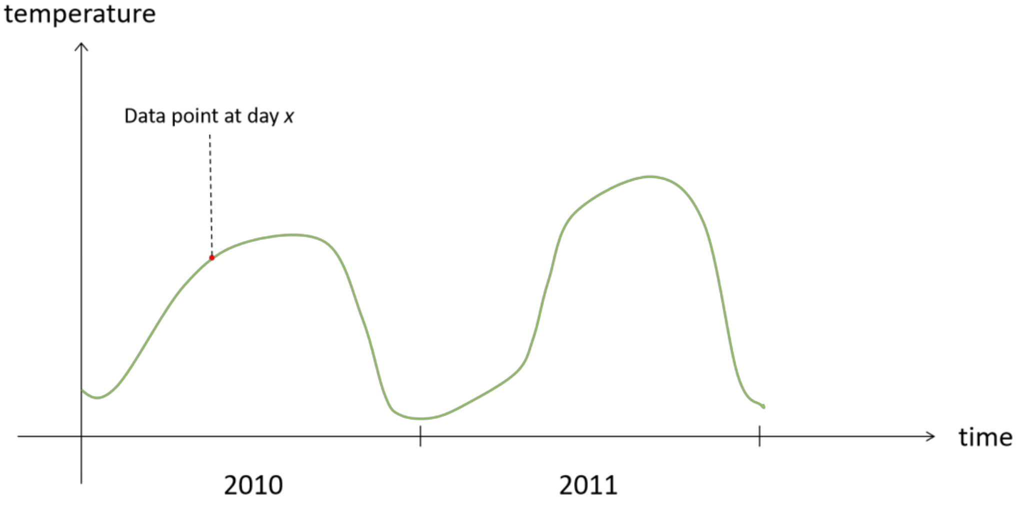


Figure 1

Figure 1 shows the obvious trend that it is warmer in the summer times than in the winter times. We can also see that 2011 had an even warmer but shorter summer than 2010. Is this now a *trend* *query* in the sense of the database project? The answer is: **no**. The reason is that all data points that together form the graph above are already stored in the database so that they only have to be retrieved from it and visualized. In other words, we only need to make a *search* in the database for the right triples (a, b, c) with $2010 \leq a \leq 2011$, $b = l_1$, and any c. This leads to very simplistic database SQL queries in which we are *not* interested in this project since databases are optimized to easily formulate, support, and execute *search* or *retrieval queries*. Instead, we aim at *complex* database queries for trend analysis that are based on data that *cannot* be found in the database but that can be *computed* or *derived* from the data stored in the database. That is, complex database queries involve *computations* that derive non-stored trend data.

The following two examples illustrate *simple trend queries*. But these queries are not complex as required. Their only purpose is to show what a trend query in the sense of this database project is. In our simple example, we could ask for the *monthly* average temperatures at sensor location l_2 between 2010 and 2011. They are *not* stored in the database but can be computed by selecting and grouping all triples by month (that is, all triples with *ti* in Jan 2010, Feb 2010, …, Nov 2011, Dec 2011) and then compute the average temperature for each group (month). This could lead to a graph like in Figure 2.

Frequently, a single graph alone is not expressive enough since it is not arranged in the context of other information or graphs. In this case, an overlay of several graphs can provide an adequate interpretation and comparison. In our simple example, it allows an easy comparison of temperature curves at different locations by drawing one curve per location (Figure 3).

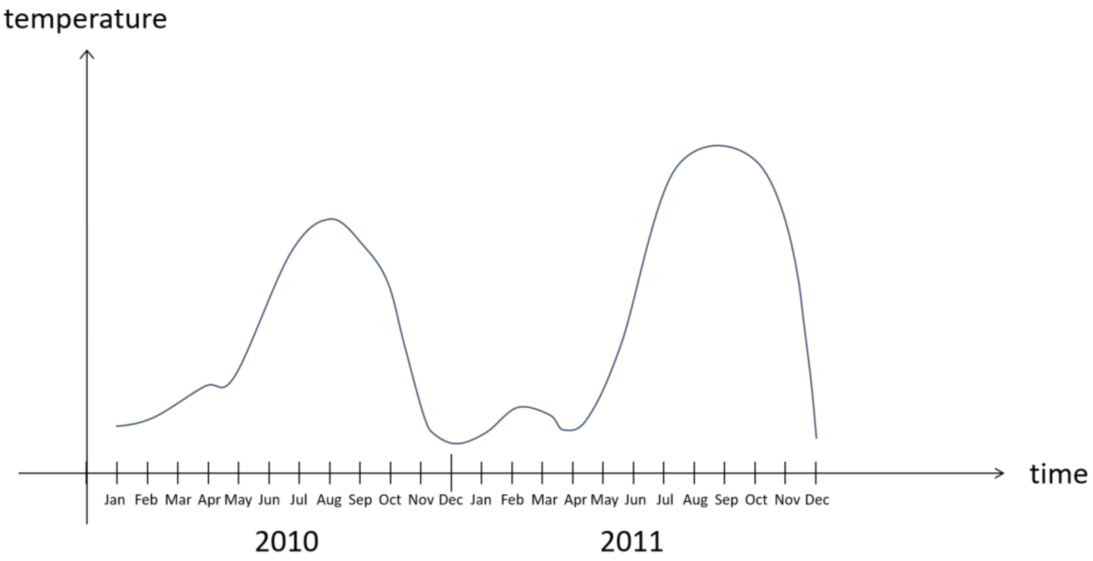


Figure 2

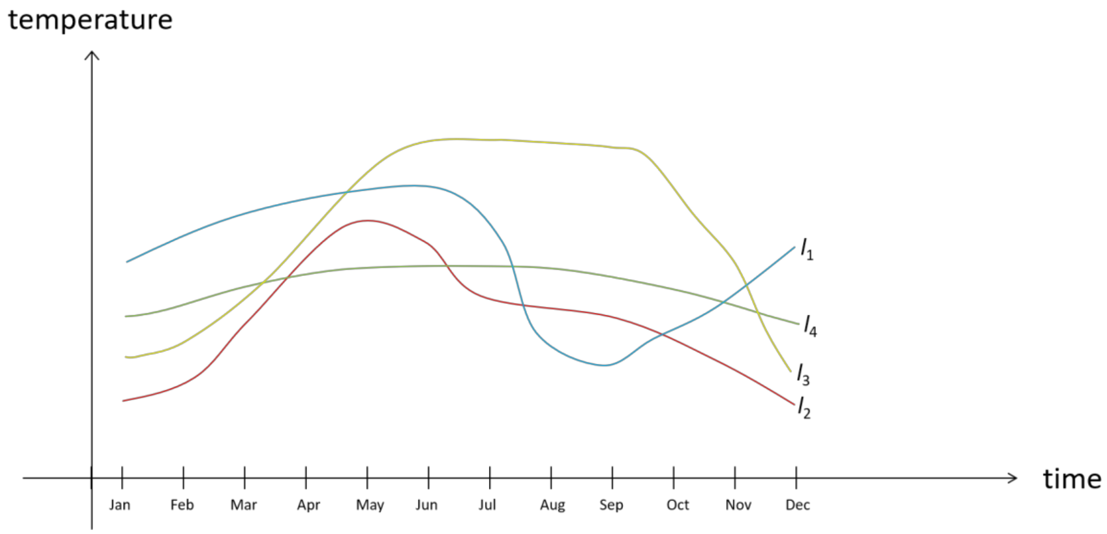


Figure 3

3.2 What is a Complex Trend Query?

While the concepts of Section 3.1 are valid for this project, the queries in Section 3.1 are only *simple* trend queries since their formulation in SQL (not shown here) leads to very simple queries in terms of structural complexity and length. This means these queries can be written down in a few lines of code and do not have to apply the more sophisticated features of SQL. This also means that there is the difficulty at the beginning of the class to design colloquial complex trend queries without having an idea if they can be later converted into complex SQL queries since SQL is unknown at that moment.

A characteristic feature of any trend query is that it considers a continuous development of data *over time*. Therefore, the term *over time* should be included in the colloquial formulation of each complex trend query. As for our example scenario from Section 3.1, a more complex query could ask to group the sensor locations into regions (e.g., states, counties) and ask for the average monthly temperatures in the selected regions between 2010 and 2011 or regarding the timespan of the whole data set. At a higher and more abstract level, a query could explore the phenomenon of climate change in this data set or if the four seasons have preserved their characteristic features over time (hot summers, cold winters, rainy falls, mild springs).

If we leave this example scenario and take *crime trends* as the overall application topic, the following complex trend queries could be of interest:

- If we consider different crime categories (such as burglary, robbery, murder, and arson), what has been the development of the different crime categories in the different districts of city X *over time*? What were the problematic crime categories in which districts? Has this changed *over time*? What are nowadays the problematic crime categories in which districts?
- Which influence *over time* in different U.S. cities has COVID had on crime rates in total and/or based on crime categories?
- How has the crime incident detection rate evolved *over time* in city X in total or based on crime categories? Crime incident detection rate is the number of detected incidents divided

by the number of recorded incidents. A detected incident means at least one suspected offender has been sanctioned. Which results hold when considering different cities at the same time?

- Is the previous query meaningful? Should the population of a city not have been considered?

Note that ranking queries are not trend queries in the sense of this project. A ranking query arranges a collection of values according to some order. An example is the query that asks for the total crime rates of all cities in decreasing order. This query does not involve time and therefore not a trend since it aggregates over all data by explicitly ignoring the time aspect.

In general, the user should get the chance to have influence on the parameters of a complex trend query. In particular, a selection of the time range and/or time granularity should always be possible. But each application also has its own specific parameters that are of interest for the user and should be adjustable at the user interface to enable a highly flexible trend analysis. Static figures and charts are usually a sign of a restricted user interface and static SQL queries.

The project requires the design and implementation of five complex trend queries. It is important to understand that these five queries must be completely different. This means that later the layout of the SQL queries, that is, their syntactical structure, must be completely different and that their semantics (meaning) must be completely different too. Copying and slightly modifying an existing SQL query will be strictly forbidden.

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 must 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 on 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 should 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 *daily* 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, first, derive new information which is not explicitly stored in the database but can be derived from the data in the database by computations and, second, represent trends (see Section 3). *Each group must list several (at least five) complex database queries in their deliverable in order to show that their application has the potential to have such interesting database queries*.

In summary, the triangle in Figure 4 shows the three main components that must fit together for a successful project. Having achieved less than these three components will not lead to success.

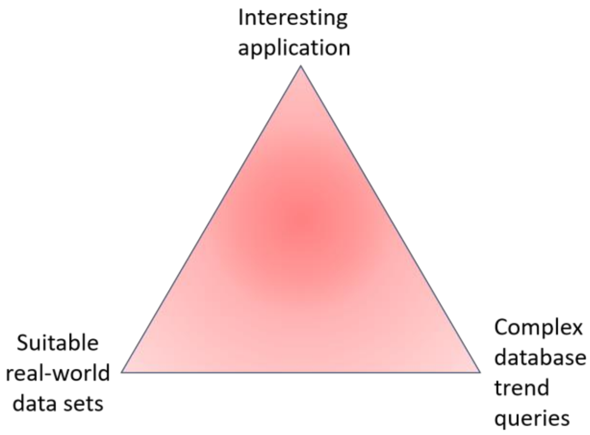
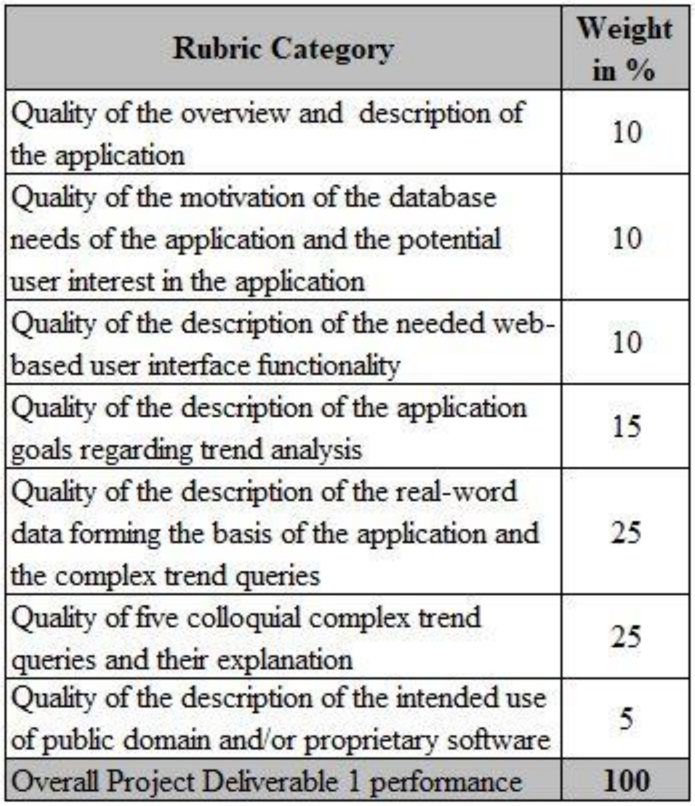


Figure 4

The rubric for grading a group's Project Deliverable 1 submission is presented in Figure 5. The first column lists the rubric categories detailed below. The second column lists the percentage with which each rubric category will be weighted. For each rubric category a group will get a performance value between 0 and 100 according to the grading table in the syllabus in Section 4.5. The weighted rubric category performances will be added up and result in the overall group's Project Deliverable 1 performance. By omitting the prefix "Quality of the", each rubric category must be a section heading in the deliverable according to the order in Figure 5. This will give each group a clear indication for the contents of the deliverable and the deliverable a clear structure. Further, a group's project must provide an application-specific title, the list of group members who have contributed to the writing of the deliverable (group members who have not contributed are not listed), and a table of contents. Each page of the deliverable must be fully filled with text and figures, and normal line spacing must be used (that is, double line spacing is not allowed). A signature page is not required.



Rubric Category	Weight in %
Quality of the overview and description of the application	10
Quality of the motivation of the database needs of the application and the potential user interest in the application	10
Quality of the description of the needed web-based user interface functionality	10
Quality of the description of the application goals regarding trend analysis	15
Quality of the description of the real-word data forming the basis of the application and the complex trend queries	25
Quality of five colloquial complex trend queries and their explanation	25
Quality of the description of the intended use of public domain and/or proprietary software	5
Overall Project Deliverable 1 performance	**100**

Figure 5

The meaning of the rubric categories is as follows:

1. *Quality of the overview and description of the application*. A reader of Project Deliverable 1 is not familiar with the application a group has in mind. Therefore, it is the group's task to provide a detailed description of the application so that the reader has a chance to understand it. Therefore, this description should be written from a reader's perspective and not from the group's perspective that already has full knowledge of the project. The group members should ask themselves what should be described so that the reader has a chance to understand the application.

2. *Quality of the motivation of the database needs of the application and the potential user interest in the application*. A first question a group should answer to themselves and to the reader is why and how their application would benefit from database support and complex trend queries. Not every application requires database support. A second question is who the users of the application are and what their interests in the group's application are.

3. *Quality of the description of the needed web-based user interface functionality*. The

web-based user interface provides support for input and output. Input refers to the possible user interactions and data input. Output refers to the graphical presentation of trend query results. The question is what the requirements of user input and graphical output are regarding the group's application.

4. *Quality of the description of the application goals regarding trend analysis*. Trend analysis by means of complex trend queries is the overall topic of the database project. The question is what the group's goals and planned achievements regarding trend analysis are.
5. *Quality of the description of the real-word data forming the basis of the application and the complex trend queries*. Finding real-world data that supports the group's application and enables the formulation and execution of complex trend queries is one of the main steps of the project. The real-world data sources must be described in a manner that both the group and a reader of Project Deliverable 1 can understand their nature, properties, and weaknesses and make an assessment about their suitability for trend analysis.
6. *Quality of five colloquial complex trend queries and their explanation*. Five different colloquial queries (not copies!) that represent trend queries must be listed (at least) and described in detail. A trend query can be recognized by the fact that (i) the graphical visualization of its result is a diagram with a time unit at its *x*-axis and a numerical unit at its *y*-axis, (ii) the trend is represented as a smooth curve, and (iii) each (time, number)-pair of the trend curve is a computed value.
7. *Quality of the description of the intended use of public domain and/or proprietary software*. Each group should already begin to determine the software to be used in the implementation phase of the project. This choice can be changed at any time later if the group decides this. But this enables the group to learn the missing software components (e.g., a web-based programming language) until the actual implementation phase begins.

Several submissions of a group's first deliverable are allowed before the deadline. The most recent submission will be graded. If the most recent submission should be 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 this 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 of the flows of action and web pages the user can expect. It starts with an optional welcome or login web page and spreads out into several 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 Figure 6.

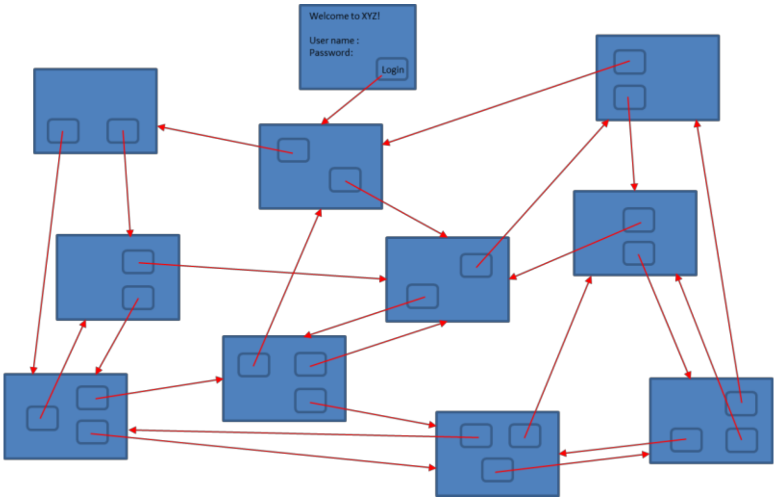


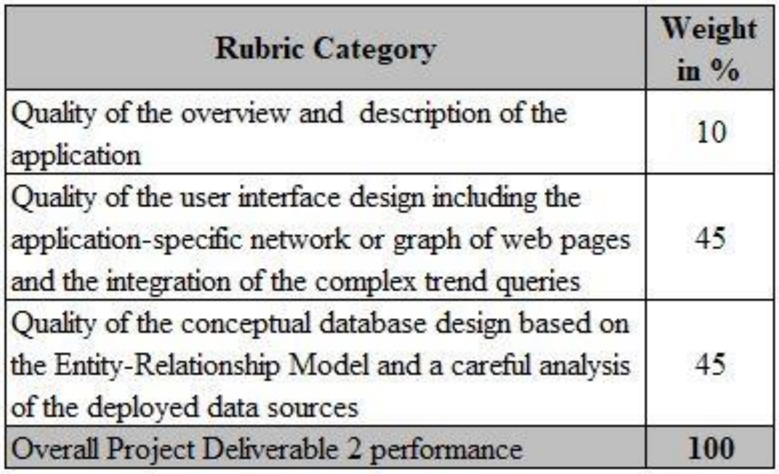
Figure 6

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 must be stored later in the database, in an ER diagram. But an ER diagram alone will not be enough. Each group should provide 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 must be put into the second deliverable. For the design of the user interface, no programming and no HTML web page construction is required. Drawings by hand, if 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. For drawing the ER diagram, the ERDPlus website and tool should be used.

The rubric for grading a group's Project Deliverable 2 submission is presented in Figure 7. The first column lists the rubric categories detailed below. The second column lists the percentage with which each rubric category will be weighted. For each rubric category a group will get a performance value between 0 and 100 according to the grading table in the syllabus in Section 4.5. The weighted rubric category performances will be added up and result in the overall group's Project Deliverable 2 performance. By omitting the prefix "Quality of the", each rubric category must be a section heading in the deliverable according to the order in Figure 7. This will give each group a clear indication for the contents of the deliverable and the deliverable a clear structure. Further, a group's project must provide an application-specific title, the list of group members who have contributed to the writing of the deliverable (group members who have not contributed are not listed), and a table of contents. Each page of the deliverable must be fully filled with text and figures, and normal line spacing must be used (that is, double line spacing is not allowed). A signature page is not required.



Rubric Category	Weight in %
Quality of the overview and description of the application	10
Quality of the user interface design including the application-specific network or graph of web pages and the integration of the complex trend queries	45
Quality of the conceptual database design based on the Entity-Relationship Model and a careful analysis of the deployed data sources	45
Overall Project Deliverable 2 performance	**100**

Figure 7

The meaning of the rubric categories is as follows:

1. *Quality of the overview and description of the application*. A detailed overview and summary of the group's application and a motivation of the relevance of trend queries for it will enable the reader to better relate the next two rubric categories to the application.
2. *Quality of the user interface design including the application-specific network or graph of web pages and the integration of the complex trend queries*. The user interface design comprises the design of the application-specific network (as described above), the design of each web page, the determination and description of the final complex trend queries, the description of their integration into the web pages, and the description of the kind of output for each complex trend query. Sufficient explanations of all aspects of the user interface design must be provided.
3. *Quality of the conceptual database design based on the Entity-Relationship Model and a careful analysis of the deployed data sources*. The conceptual database design requires a final selection, analysis, and description of the relevant objects and attributes of the selected data sources and their arrangement as entity sets, relationships sets, and attributes in an Entity-Relationship diagram. References to the data sources and sufficient explanations of all aspects of the conceptual database design must be provided so that it becomes clear how the Entity-Relationship diagram is designed.

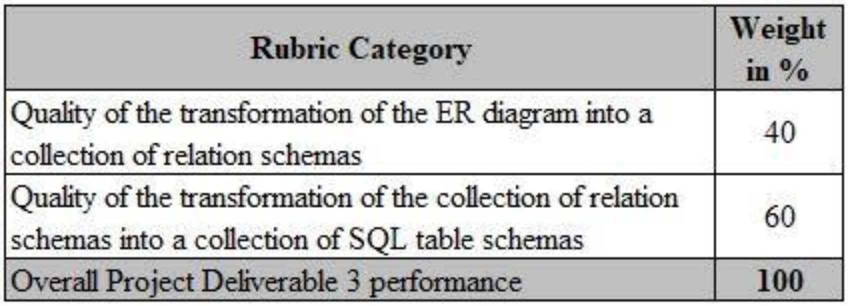
Several submissions of a group's second deliverable are allowed before the deadline. The most recent submission will be graded. If the most recent submission should be so dissatisfactory that the group cannot move on with the next phase, a revision will be requested from the group that will not be considered for (re)grading. This means, the more effort a group puts into this 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) represents 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 must be submitted, the normalization step will be skipped. Groups that are willing to rearrange their database schema later may perform the normalization process. However, this will require a new upload of all data according to the modified database schema.

The rubric for grading a group's Project Deliverable 3 submission is presented in Figure 8. The first column lists the rubric categories detailed below. The second column lists the percentage with which each rubric category will be weighted. For each rubric category a group will get a performance value between 0 and 100 according to the grading table in the syllabus in Section 4.5. The weighted rubric category performances will be added up and result in the overall group's Project Deliverable 3 performance. By omitting the prefix "Quality of the", each rubric category must be a section heading in the deliverable according to the order in Figure 8. This will give each group a clear indication for the contents of the deliverable and the deliverable a clear structure. Further, a group's project must provide an application-specific title, the list of group members who have contributed to the writing of the deliverable (group members who have not contributed are not listed), and a table of contents. Each page of the deliverable must be fully filled with text and figures, and normal line spacing must be used (that is, double line spacing is not allowed). A signature page is not required.



Rubric Category	Weight in %
Quality of the transformation of the ER diagram into a collection of relation schemas	40
Quality of the transformation of the collection of relation schemas into a collection of SQL table schemas	60
Overall Project Deliverable 3 performance	**100**

Figure 8

The meaning of the rubric categories is as follows:

1. *Quality of the transformation of the ER diagram into a collection of relation schemas*. First, the possibly modified and improved ER diagram from Phase II must be shown. Second, the ER diagram has to be transformed into a collection of relation schemas of the form $R(A_1 : D_1, \ldots, A_n : D_n)$ as described above. The data types should not be SQL data types but can be more abstract (e.g., *string, real*). Explanations of the transformation process are required.

2. *Quality of the transformation of the collection of relation schemas into a collection of SQL table schemas*. The relation schemas must be transformed into SQL table schemas by using the create table command. The SQL table schemas should be enhanced by all needed and desired integrity constraints and executed in the Oracle SQL Developer. Original screen snapshots from CISE Oracle and the Oracle SQL Developer are required that show all *create table* commands and the created empty tables. Explanations of the transformation process are required.

Several submissions of a group's third deliverable are allowed before the deadline. The most recent submission will be graded. If the most recent submission should be 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 this 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 fill the database schema with data.

4.4 Phase IV: Project Software Implementation

At this point, each group should have a rather clear picture of what 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, (ii) 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 groups starting too late with the implementation in the semester and then getting 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 must 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 establish 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 bulk loaded into the database?
- Has the database schema been created in Oracle?
- Have the pre-processed data been bulk loaded into the database according to the database schema?
- Has the minimum of 500,000 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 has 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

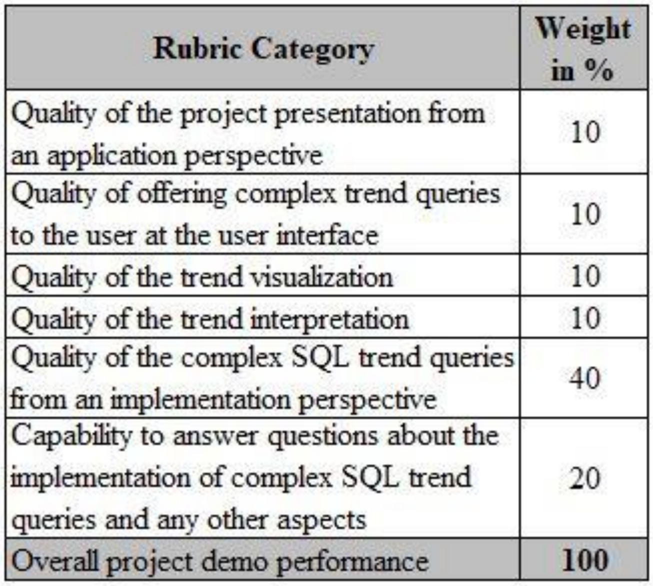
On the demonstration day(s), which will be at the end of the semester, groups must present their software system. The groups must arrange an appointment with the instructor. Demonstrations will last 30 to 45 minutes and will be conducted online with a web-based presentation tool that each group selects 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 to 45 minutes. In the first 10 to 15 minutes each group will give 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 to 10 minutes, the instructor will determine the group’s project demo grade. Note that these time intervals are flexible and can be changed by the instructor during the presentation.
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 about all aspects of the group's software with respect to both its functionality and its implementation. The group should be able 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. Documentation of the software is not required.
7. Further rules are given in the syllabus.



Rubric Category	Weight in %
Quality of the project presentation from an application perspective	10
Quality of offering complex trend queries to the user at the user interface	10
Quality of the trend visualization	10
Quality of the trend interpretation	10
Quality of the complex SQL trend queries from an implementation perspective	40
Capability to answer questions about the implementation of complex SQL trend queries and any other aspects	20
Overall project demo performance	**100**

Figure 9

The rubric for grading a group’s project software and demonstration is presented in Figure 9. The first column lists the rubric categories detailed below. The second column lists the percentage with which each rubric category will be weighted. For each rubric category a group will get a performance value between 0 and 100 according to the grading table in the syllabus in Section 4.5.

The weighted rubric category performances will be added up and result in the overall group's project demo performance.

The meaning of the rubric categories is as follows:

1. *Quality of the project presentation from an application perspective*. The ability to "sell" the highlights of a created software in a project demonstration is an important, general skill. The software producer (that is, a group) must convince the customer (that is, the TA, grader, and/or the instructor) about the high quality and meaningfulness of the software product. Therefore, the expectation is that the project presentation is clear, fluent, convincing, based on arguments, and focusses on the highlights of the application.
2. *Quality of offering complex trend queries to the user at the user interface*. The design of complex trend queries is the main objective of the database project. From an application perspective, the question is therefore how to offer them in the user interface. A user should get motivation for the meaningfulness and interestingness of the proposed trend queries. Further, it is important that a user can exert influence on the parameters of such a query. They introduce some dynamics and flexibility into the application and avoid static query results. An example is the ability to allow the user to execute a query with respect to different time ranges and thus in different temporal resolutions. But application-specific parameters are important too.
3. *Quality of the trend visualization*. The expressiveness of computed complex trend queries is limited without an adequate graphical visualization. Hence, it is important to find a suitable graphical visualization method for each trend query such that a good interpretation of the query result is enabled. Graphical visualization techniques nowadays allow a user to move the mouse over a figure and get additional information loaded in real-time from the database or pre-computed by a database query.
4. *Quality of the trend interpretation*. The value and quality of a complex trend query is reflected by its interpretation. Important aspects are what the graphical visualization of a trend query shows us, how we can interpret the visualization and thus query result, and what we can learn from it. The software producer must be able to provide a convincing and clear interpretation of the trend query results.
5. *Quality of the complex SQL trend queries from an implementation perspective*. Learning the design of complex queries is one of the main objectives of the database project. Therefore, the focus is set on the structural complexity of SQL trend queries embedded into the source code of the application. Structural complexity is given by an appropriate combination of aggregations, groupings, and nested queries and must always be seen in the context of the meaningfulness and expressiveness of trend queries.
6. *Capability to answer questions about the implementation of complex SQL trend queries and any other aspects*. A software producer must be able to answer any question about her software product. The questions of interest here mainly refer to the design and implementation of complex SQL queries and other relevant database aspects. But questions regarding the user interface are not excluded. SQL queries will be explored in the original source code.

In summary, 60% of the overall project grade (points 5 and 6) refer to the database work. The remaining 40% (points 1 to 4) relate to the application.