Juhani Välimäki and Kari Silpiö

DM / CASE ASSIGNMENT (*Mandatory assignment*)

REQUIRED DELIVERABLES

1. **Database Design documentation** (One Word or PDF document: ***DM_Case_Design**.docx or.pdf*)
   1.1 ER diagram, 1.2 Relational schema, 1.3 Repository, and 1.4 Database diagram (in STEP 2)
2. **SQL Scripts** (four ***.sql*** files)
   - 2.1 SQL script to create the database structure *in SQL Server*
   - 2.2 SQL script to create the indexes
   - 2.3 SQL script to populate the database with a *reasonable amount of proper data* for testing
   - 2.4 SQL script to test the database (*see STEP 2 below*)
3. **Project Closing Report** (One Word or PDF document: ***DM_Case_Closing_Report**.docx or .pdf*)
   - 3.1 Evaluation of the product and the process
   - 3.2 Working time records per each person per each task

THE DEVELOPMENT PROCESS / Group work (*Group size: 2-3 students, 3 is the preferred size*)

STEP 1: Database Modeling

Administrative

- Record and ***report working time by person and task*** (otherwise, your points will be reduced)

Conceptual database modelling

- Create a conceptual model of the database and visualise it as an ***ER diagram***.
- Write entity type definitions to the ***repository*** (*see the separate repository template*)

Logical database design

- Derive a ***relational schema*** from your ER diagram.
- Validate the relations with the BCNF rule. Fix the relations if they are not in BCNF.
- Define integrity constraints for your relations. Document them in the ***repository***.

→ **Submit STEP 1 results** to Moodle in a **single ZIP file** (***DM_Case_Step1_**Surname_Surname.zip*)
Submit the results ***before*** you move on to STEP 2!
* You might need to improve the design later. This will be included in the final submission ☺

STEP 2: Physical design, database implementation in SQL Server and testing

Administrative

- Record working time ***by each person and task*** (otherwise, your points will be reduced)

Physical design and implementation

- Write an SQL script to create the database structure in SQL Server
- In SMSS, create a database diagram of your database tables
- Design a basic set of indexes on your tables and write an SQL script to create them

Testing

- Design test data and write an SQL script to insert it into the database.
- Write an SQL script for the 15 user transactions listed at the end of this document.
- Test your implementation by executing the SQL statements and write a short test report.

STEP 3: Evaluate your work

Write a ***project closing report*** where you discuss the questions below.

1. *How well does the product meet customer's requirements?*
2. *What went well? What was difficult?*
3. *What did we learn? What will we do better next time?*

Include in the report the ***working time records per each person***.

STEP 4: Submit everything to Moodle in a ***single ZIP file*** (***DM_Case_Final_**Surname_Surname.zip*)

The Case - Helsinki Historic Bike tours

After the covid-19 was defeated in late May 2020, a group of history and bicycle enthusiasts in Helsinki decided to organize historic bicycle tours around the city. Some tours would go through e.g. a few of the 1952 Olympics' sites (like Velodrome, Kumpula swimming pool, Laakso riding hall, Tennispalatsi, Olympiaterminaali). Some other sites could include e.g. Architectural sites, or Urban culture spots. Of course organizer of the tour is free to select any combination of sites.

The tours are free for the "customers", but the group wants to know the customers' age group (15-17,18-50,51-69, or 70+) to adjust the speed of the tour. Because of the government ruling, maximum attendance to each tour has been limited to 10 attendees, including the guides=organizers.

Further details

- To be able to enroll on a tour, a registration is required. To enable that the customers have to give permission to store first name, last name, email, one phone number. If the customer doesn't agree, => cannot register, cannot enroll, cannot join.
- Tours have a name and description.
- People often change their plans; thus, we want to give the place to the next person waiting (based on the earliest reservation time.
- Different tours have different routes. Even if the customers make the final decision, it would be good to know if there are elderly people enrolled for a very demanding tour. (A lot of climb) - Each tour has a responsible organizer.
- We keep the same information about the organizers as about the customers. Of course, they differ by that that they have been **accepted** to be organizers or tours.
- The customers and organizers can both give their evaluation for each Site. The evaluation must follow the categorization: "Highly recommend, enjoyable, okay, only as a filler, would never visit"
- Organizers can also attend tours as a customer. Provided that they are not the
- To give evaluation the customers need to give consent to store the.
- (No need to take the GDPR into account more than that don't store anything really personal, like PIN or home address. This is a school exercise)

The new database application should help to complete the tasks (user transactions) listed below[1].

1. Listing available tours with their enrolment count and sites to be visited
2. Making an enrolment for a tour
3. Cancelling a tour enrolment
4. Listing the waypoints (Sites) of a tour.
6. Listing the Sites and reading the details of each site, e.g. the tags related to the site. E.g. **Helsinki Velodrome** might have the tags "1940","1952 Olympics","Functionalism". **Malmi airport** might have the tags "1936", "Functionalism", "Aviation". **Kisakylä** in Käpylä might have tags "1952 Olympics","Housing".
7. Cancelling the whole tour (e.g. when organizer sick, or only 1-2 customers enrolled)
8. Emailing all enrolled people in the case of a cancelled tour.
9. Maintaining a custom instruction text (specific to each age group).
10. Even if customer is encouraged to select the age group, this is not obligatory. So, we might have customers for whom the age group is not specified.
11. For the organizer: Listing names (and possible age groups) for all customers of a tour.

Examples of the queries (user transactions) the new application should support are the following:

1. Who are the organizers of the tours organized 2020-06-03?
2. What are all the tags that can be associated to a certain tour X?
3. List of tours on a certain day, and along each tour the list of tags for each tour.
4. Which sites have lowest evaluations by the biggest agegroup?
5. What are the tags (in order of occurrances) for the sites that people belonging to agegroup X have evaluated with the highest evaluations? This is a quite long join query, do not expect to see the needed tables be connected directly but via a long path.
6. Full report of tour X attendees: organizer + enrollees with their names and agegroup.
7. The itinerary for tour X, the way points = sites in order, and the tags for each site
8. What tags are not yet in use = not used by any site?

Helpful hints:

- Some entities that appear to be separate, might actually be tackled with single one. But don't try to cram things into one either, if they seem to have separate nature.

- Start by looking at one entity only first. And think how it's related to others. => concentrate on the quality of your findings, not rushing to get everything in the model first.

- Some requirements actually talk about the application that will later be run, database just has to have the base information. We don't want to include derived attributes in the database

NB! In this case assignment, you will use the user transactions mentioned above to

1. Validate the design of the database
2. Test the physical implementation of the database.

INSTRUCTION

Validate and test your database structure by ***writing and executing the SQL statements*** for the 11 user transactions mentioned above.

In this case assignment, your task is just to ***write the required SQL statements*** for the listed user transactions. There are a few SQL DDL scripts and 11 + 8 SQL DML commands to be written.

In this case assignment, you can test this by executing an update manually. The query should get the current date from the DBMS.