Databases and information management
Final Project

Piotr Syga

December 14, 2022

1 The goal

The aim of the project is to model, design and implement a complete application that utilizes a database for storing and processing the data. The database used in the application should show the constructions learned during the classes and solve the security and correctness issues discussed during the course. The project can be done in pairs, on a condition that each person in the group works on the database part of the app. Any programming language may be used to implement the application, just as any dialect of relational or non-relational (e.g. MongoDB, Cassandra, CouchDB or ElasticSearch) databases may be used. Note that, the grading is based on the functionality of the application, not on its visual aspect. The aim of the frontend is to show your ability to connect the database with an app, and provide higher UX. We do assume that the final user does not have any programming language, nor database proficiency.

2 Modeling

The authors should provide a pdf-format report including the authors' information, the name of the project, short motivation behind the project and the description of each of the modeling stages during the database creation.
The report should describe the purpose of the application, the requirements (both functional and non-functional) and constraints described by the user. The authors should identify and denote all the most important entities as well as include diagram of those entities showing the attributes and the relations between the entities (use ERD, e.g., Martin's notation or UML). The diagrams should have a brief description, explaining the schema and indicate the tables, views, functions, procedures and triggers in the database. The design should include information about access rights to the various components by different users (database level, not application level) of the database. The **pdf** file with the report should be sent by email at least **36 hours**

before handing in the project in class. Solutions not meeting this requirement will be treated as unfinished and subject to penalty for the skipped deadline.

3 Implementation

Below the minimal requirements to the database-based applications are listed, note that the requirements to the GUI are not listed. Some of the requirements are listed in both NOSQL and SQL for ease of localizing your case.

SQL databases:

- The database should include at least 5 tables. Each table should have no fewer than 150 records (in justified and agreed upon cases a table might be smaller, this is done for auxiliary tables, in cases when the main tables are larger).
- The authors should provide database coherence via foreign keys, triggers and check constraints.
- All the tables should be normalized up to the relevant normal form (2NF or 3NF depending on their type), justification for the proper normal form should be known.
- The application should use of at least 3 different levels of access, depending on the logged on user (for example: admin, CEO, accountant, employee, customer or a guest) – logging in can be implemented as logging into another database user on connection, or directly at the database level. The database users should have appropriate privileges granted and their password hashed. The application level users should have their passwords protected.
- The application should allow the user to SELECT records based on the criteria entered in a text field (dynamic query builder).
- The application should allow adding, deleting and modifying the records from the application-level, without the need for the user to enter SQL queries.
- The database should use SQL-side transactions and PREPARE STATEMENTs, more complex operations should be performed using transactions (there should be at least several in the project). The use of functions, cursors and CTEs is encouraged, yet not mandatory by default. The requirement of using such depends on the specific design.
- The robustness against SQL injection should be provided. A security validator, e.g. Vega may be used.

- For a user with the highest privileges, there should be a button in the app that allows creating a backup or restoring the base from a backup.

NOSQL databases:

- The project should be in a form of web portal or a mobile app.
- The application should allow adding, deleting and modifying the records from the application-level, without the need for the user to enter queries. Note that the app should allow using documents (or different storing entity for a different language, e.g. tables with complex column types for Cassandra) with different schema, both omitting and adding, possibly complex, fields.
- The database should use mirroring, sharding or replicas, so that the data is distributed across multiple nodes.
- The application should use of at least 3 different levels of access, depending on the logged user (for example: admin, CEO, accountant, employee, customer or a guest) – this should be implemented using database mechanisms.
- The application should allow to register new users. Their data should be secured.
- For a user with the highest privileges, there should be a button in the app that allows creating a backup or restoring the base from a backup.
- The application should allow the user to SELECT records based on the criteria entered in the text field (dynamic query builder).
- The application should allow importing new data (which can be inconsistent with existing data in terms of schema) from a JSON, XML or CSV file.
- Security of the app should be provided, a validator may be used.
- Tests for a minimum of 5 simultaneous users connected to the database should be performed and the CAP-model properties should be provided.

Note that the script that creates the database, users, tables, functions, etc. and assigns the appropriate permissions to users is an integral part of the project and should be presented to the lab instructor.