Databases Project

2014/2015

Introduction

The aim of this project is to provide students with experience in developing database systems. The project is based in industry's best practices for software development, thus students will experience all the main stages of a common software project, from the very beginning to production release.

The **functional description** of the project is available in the **companion document** (in annex A) and it is common to 2 courses: BD and SD (both from the 3rd year of the LEI).

Objectives

At the end of this Project students should be able to:

- Understand how a database software development project is organized, planned and executed;
- How developers gather and identify the requirements for a software product;
- Master the creation of conceptual and physical data models for supporting and persisting application data;
- Design, implement, verify and validate, and deploy a database system;
- Install, configure, manage and tune a modern DBMS;
- Understand client and server side programming in SQL and PL/SQL (or similar).

Quality attributes for a good project

Your application must make use of:

(a) SQL and PL/SQL, or similar;

(b) Triggers, functions and procedures running on the DBMS side;

(c) A transactional DBMS (the use of ORACLE is optional);

(d) A distributed architecture (e.g., client-server);

(e) Good error avoidance, detection and mitigation strategies;

(f) A functional user interface;

(g) Good strategies for managing the concurrency conflicts;

(h) Good documentation.

Your application must also respect the functional requirements defined in the companion document (in annex A) and execute without "visible problems" or "crashes".

To fulfill the objectives of this assignment you should be as creative as you want, provided you have included this list of features in your solution.

Milestones and deliverables

Midterm presentation – The team must present their work in the PL classes running from 3rd to the 7th of November.

- **Presentation** with the following information:
- Name of the project
- Team members and contacts
- Brief description of the project and **potential concurrency conflicts**
- Provide a description of a potential solution if you already have one
- Core technologies: Programming Language, DBMS, Libraries, etc.
- Development plan: planned tasks, initial work division per team member, timeline and estimates
- **ER diagram**
- Description of entities, attributes, integrity rules
- **Relational data model** (tables)
- Estimation of the DB size for a specific time period (e.g., 3 years)
- Only tables (leave indexes and other structures out)
- Size of a table = size of each record X number of records

Final delivery – 12th December - Deliveries must be submitted to ***Inforestudante*** until **23:55** in the day of the deadline. Each group must select a member for performing this task. All submissions must clearly identify the team and the students working in the project.

- Upload into *Inforestudante* the following materials and documents:
- **User manual** describing how users can interact with the application.
- **Installation manual** describing how to deploy and run the software you developed
- Include the source code, scripts, executable files and libraries necessary for compiling and running the software (identify the used SGBD, do not upload its binaries)
- **Final ER and Relational data models**
- **Development plan**: make sure you specify which tasks were done by each team member and the effort involved (e.g., hours)

- **Scripts**: PL/SQL and DB creation

- DB creation scripts containing the definitions of tables, constraints, sequences, users, roles, permissions, triggers, functions, procedures and physical storage parameters (Next, Initial, PCTFree, PCTUsed, etc.)

Defense – 15th to 19th of December

- Prepare a **10 min presentation** and **full live demonstration** of you software

- **Prepare yourself** to answer questions regarding all deliverables and implementation details

- **Sign up** for any available time slot for the defense in *Inforestudante*

- The list of available slots will be release in December

- Sign up until the final delivery due date

- Location will be briefed in time

Notes

- Do not start coding right away. Take time to think about the problem and to structure your development plan and design;
- Always implement the necessary code for **error detection and correction**;
- Assure a **clean shutdown** of your system (no memory-leaks);
- **Plagiarism or any other kind of fraud will not be tolerated**.

Annex A – Functional Description

Meeto: Collaboration and Social Networking

Databases / Distributed Systems - 2014/2015

1. Objectives of the assignment

At the end of this practical project, you should:

- Understand how a database software development project is organized, planned and executed;
- How developers gather and identify the requirements for a software product;
- Master the creation of conceptual and physical data models for supporting and persisting application data;
- Design, implement, verify and validate, and deploy a database system;
- Install, configure, manage and tune a modern DBMS;
- Understand client and server side programming in SQL and PL/SQL (or similar).

2. Work description

Meetings are necessary when several people collaborate and work together. Students meet to coordinate their work on practical assignments; employees meet to discuss the status of projects; managers meet to make key decisions concerning their organizations; and so on. We need to prevent people from wasting time in useless meetings, because *time* is one of our most valuable resources.

The goal of this assignment is to build a distributed system to improve the effectiveness of meetings. The system shall be able to support the preparation of meetings, writing down the minutes, and following up on the status of action items resulting from a meeting.

3. Functional requirements

Every user that connects to your application should be given two options: register a new account, and login as an existing user. Once you've logged into the system, you may choose from several options:

- **Schedule a meeting**. The meeting leader creates a new meeting by setting a title, the desired outcome, date, time, and location for the meeting. It is possible to immediately add other users that will be invited to attend the meeting and who will be able to access all the information concerning the meeting.
- **Set agendas democratically**. The meeting leader may initially add items to the meeting agenda. However, any users invited to the meeting may add, modify, and delete items from the agenda. The meeting agenda is therefore set democratically, with a last item "any other business" belonging to all agendas.
- **Discussion**. Each item on the agenda of a given meeting may be discussed within your application. In essence, each agenda item has a chat room where meeting participants can write down their comments and thoughts.

- **Finalizing a meeting**. Throughout the discussion, participants may add *action items* and *key decisions* to the result of the meeting. An action item is a task that is given to a user (not necessarily someone who attended the meeting). The user who becomes responsible for an action item should, after the meeting, be able to mark the action as done once it is finished. A key decision is the result of a discussion that took place during the meeting, and should be associated to the agenda item that was being discussed.
- **Check all meetings**. Any user may, at any time, use the application to check upcoming meetings, to read any information belonging to a past meeting, to accept/decline meeting invites, etc.
- **Manage user groups**. Instead of inviting users one at a time, groups of people working together should all be invited at once to attend a meeting. Having groups of users also allows one to easily select people to become responsible for action items at the end of each meeting. Therefore, it is necessary to create user groups and manage the members.

4. Non-functional requirements

The application should obey the following non-functional requirements:

- Clients should never be aware of network problems, except for long network failures.
- Data should be persistent. This means that crashes and network problems should not compromise the existing data, neither on the client nor on the server.
- Any operations requested by the user should not be lost. The application is not allowed to lose any operation even if it is not committed to the server.
- The messages exchanged in the discussion (of each agenda item) should be delivered to all clients in causal order.
- When a meeting agenda is being set, democratically, all operations should be transactional. If there is a network problem, one such operation should either be fully completed or fully reverted.
- Separation of concerns: the application's communication, business logic, and data should be separated and should be able to run on different machines (just requiring configuration).
- Any user interface is acceptable. Graphical user interfaces will not be rewarded when it comes to grading, so you should focus on the contents of the project, robustness, fail-over, and communication protocols.
- During the project defenses, the demo has to be performed on 2 or 3 different computers (one per team member).