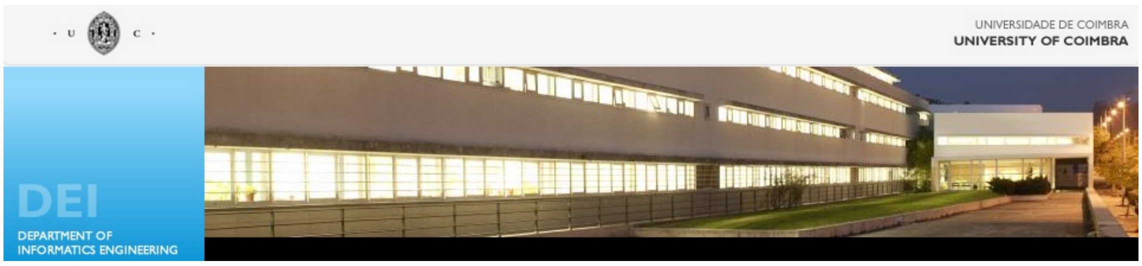

Databases Project

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 3 courses: BD, SD and ES (all 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 (see below for more information);
(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.

Concurrency conflicts

Before reading this section you should read the project description in annex A. This project aims to provide a way for people to exchange ideas and to validate them by obtaining an indication of how much each idea is worth. The users of the system submit new ideas, which they can trade much like in a stock exchange market.

In this program there are at least 3 critical sections, where the concurrent accesses can cause inconsistencies in data:

1. You must prevent 2 or more concurrent buy operations from de same buyer to result in negative credit for the buyer. Buy operations decrement the value of an acquisition from the buyer's credit, thus 2 concurrent buys must not take the credit value below zero.
2. When two users issue simultaneous and immediate buy orders for the same idea, you must assure that they do not get more shares then those available.
3. When buy and sell operations are performed over the same idea, they must be completed atomically in order to prevent inconsistencies in the final number of shares per idea.

Milestones and deliverables

Deliveries must be submitted in **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.**

Midterm presentation – 17th November

- **Upload into Inforestudante a presentation** (PDF file) 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
- The team must **present their work in the PL classes running from 18th to the 21st of November**.

Final delivery – 13th December

- 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, ect)

Defense – 16th to 20th 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

IdeaBroker

Databases, Distributed Systems, Software Engineering 2013/2014

This project aims to provide a way for people to exchange ideas and to validate them by obtaining an indication of how much each idea is worth. Ideas are submitted by users, who then trade each idea much like in a stock exchange market.

1 Ideas

At first, your application should give two options to every user that connects to it: register an account, and login. After logging into the system, there are several options you should provide the user with:

1. *Create a Topic.* Topics work as themes for ideas to be connected to.
2. *Submit an Idea.* An idea consists of a short text (a few hundred characters), and should be connected to at least one topic. Ideas can also have attached files that should be uploaded to the server (you may choose to accept only images). As a suggestion on how to connect topics to ideas, you may use Hashtags to identify topics, such as Twitter does.
3. *Delete an Idea.* If an idea is added by mistake, the author should be able to remove it. However, that is only possible if the user owns 100% of the shares of the idea.
4. (Groups of 3 elements only.) *Create and manage private groups.* Groups are a mandatory feature for teams with 3 members. Both Topics and Ideas can be connected to a group, making them restricted to group members. Group managers should be able to add and remove users from groups. This feature is intended to let organizations use **IdeaBroker** without sharing the topics and ideas with the rest of the world.

2 Trading

Once an idea is published in the **IdeaBroker**, users may trade it as in a stock exchange market.

1. *New users.* Every new user is awarded 1 million deicoins after registering. Each user must be validated through an unique identifier, such as an Identity Card Number.

2. *New ideas.* Every new idea is divided into 100 000 shares. When creating an idea, the user decides how many of his/her own deicoins should be invested into the idea. The initial sale price of the idea is then the amount invested divided by 100 000, and the amount invested is deduced from the user's account.

3. *Buy shares of an idea.* When an idea is submitted, 100% of its shares belong to the author. Other users can buy X% of that idea instantaneously, without any need for approval from the seller.

4. *Selling shares of an idea.* Idea shares are always for sale, and a user is only able to choose the selling price.

5. *Set selling price.* Users may increase or decrease the price at which their shares can be automatically sold. A new selling price should also be set upon buying shares. The creator of an idea is unable to increase the selling price of that idea before the first trade takes place.

6. When a user gives a buy order for a given amount of shares, the system should automatically choose the set of users from whom the shares will be bought. For example, if a user wishes to buy 100 shares of an idea, and two other users have 50 shares each, there will be two transactions.

7. *Takeover by the root.* A special user account named *root* should be able to acquire, at any time, 100% of the shares of an idea, by paying every user (regardless of his/her selling price) the price of the last trade of the corresponding idea.

8. *Hall of fame.* After being taken by *root* and removed from the market, ideas should be displayed in a *hall of fame*, with related information (idea description, valuation, market lifecycle, etc.).

3 Views

At a very minimum, users should be able to browse the existing ideas, including four specific views:

1. *By search.* Users search for ideas or topics.

2. *By topic.* By selecting a given topic, users are presented with the ideas connected to that topic.

3. *Portfolio.* Users view every idea for which they own shares.

4. *Watchlist.* Users may add ideas to their personal watchlist, and follow the ideas being watched.