

CSE 532: Project 1 - Online Conference Submission System

Fall 2005

Deadline: Oct 31

In this project, you will design and implement an object-relational database for a system that enables online submission and reviewing of conference papers.

You will be using Java/JSP/Servlets to build the application front end to your database. Database connectivity should be done using Java/JDBC. The most convenient way to work with servlets is to use NetBeans (<http://www.netbeans.org/>), which is installed in the lab. It has the Tomcat server in it, which is started and shut down automatically when you start/shutdown NetBeans. Another option is to use Eclipse.

1 General Description

The Conference Submission System is designed to enable the work of conference program committees, which review and select papers for presentation. The system has three main functions:

1. Enable paper authors to submit papers. For simplicity, we assume that paper titles and authors are entered in batch mode, so no GUI is required for that.
2. Enable program committee members to submit their reviews and rankings of papers. This function will require a separate GUI.

Normally each paper is ranked on relevance, technical content, readability, etc., but we will assume that there are only two types of ranking: overall quality and reviewer's confidence in the ranking. A rank is a number between 1 and 5 (with 5 being the highest). We will assume that the actual reviews on the papers are not entered into the system (so ignore that aspect).

Each committee member is assigned several papers to review, and a member can give a ranking only to the papers assigned to him. Because of this restriction, each committee member would have to be identified to the system through a login.

3. Enable analytical querying. These queries allow the committee members to ask a variety of queries about the submitted papers, their ranking, etc. The analytical side of the Conference Submission System requires a separate GUI.

2 System Users

Normally, conference submission systems have three types of users: paper authors, program committee members, and program committee chairs (typically one to three). In our application, only program committee members—each with a private login—will be able to access the system.

3 Required Data

The data items required by your system roughly fall into these categories:

- *Information about program committee members.*
- *Information about papers.*
- *Information on paper authors.*

Note that committee members can be authors and vice versa.

When arranging the data items in tables, you are to use the *object-relational* features of Oracle as much as possible. You should determine the relationships among tables, identify the key attributes, etc. In addition, you should associate indices with your tables to speed up processing of the given queries. Finally, you should specify and enforce referential integrity constraints and other CHECK-style constraints on the data.

The schema should reflect the fact that committee members and paper authors have parts of their structure that are common to both. These parts must be factored out using inheritance. Every paper author has

- Name
- Papers authored or co-authored (among the papers submitted to the conference)
- Affiliation

A committee member has

- Name
- Affiliation
- Papers that are assigned to that member for review

Papers have

- Title
- Authors
- Reviewers
- Ranking information. Ranking information is a set of objects, one per reviewer, which describe the rankings given by each reviewer to the paper. As mentioned earlier, each ranking includes two numbers in the range 1–5: a measure of the overall quality of the paper and reviewer’s confidence in his assessment.

4 Queries

You are to implement the following queries and provide an natural graphical interface to them. All these queries must be **implemented entirely in SQL** using its object-relational features.

1. List all papers in the order of the *weighted average* of the rankings (the highest going first). You are free to devise a *reasonable* formula for a weighted average, which should take both components of ranking: paper’s quality and reviewer’s confidence. For instance,

$$\frac{1}{\#of_reviewers} \sum_{i=1}^{\#of_reviewers} quality(i) * confidence(i)$$

2. Find the papers that have a conflict of interests and must be reassigned to other reviewers because some of the paper authors happen to be also that paper’s reviewers. For each such paper, show the list of reviewers who must be reassigned due to the conflict of interest.

A separate update operation should be provided to remove the reviewing assignments that have a conflict of interest. This function would normally be reserved for program chairs, so we are simplifying things here.

3. Display papers such that the majority of its reviewers gave it an “accept” rating (4 or 5) with high confidence (also 4 or 5).
4. Find the reviewers who have reviewed the largest number of papers.
5. Find the most controversial papers (the papers that have the largest discrepancy between their weighted rankings). That is, for each paper, find the maximal distance between the weighted rankings assigned to that paper by the different reviewers, and display the papers with the largest distance.
6. Find the *median* weighted ranking among all papers. This is a number, X, such that the number of papers with weighted average rankings lower than X is the same as the number of papers with weighted average rankings greater than X.
7. Let *reviewerOf*(person1,person2) represent the relation such that person1 is a reviewer of a paper written by person2. Compute the transitive closure of this relation.

This is a recursive query. Unfortunately, Oracle’s recursive query facility is botched, so you will have to do it in two ways. First, in your final document you will write a proper SQL:1999 query for the transitive closure. Second, in your system you will need to use the botched Oracle syntax so that you could implement that query. Include the Oracle formulation of that query in the final document as well.

5 Documentation

Your complete database implementation must be accompanied by a project document. The document should include:

- The Entity-Relationship (ER) design and the diagram of the complete project.
- A clear description of the database scheme, including a discussion of your design decisions. Remember that your design should be object-relational.
- Description of **integrity constraints**, including referential integrity and CHECK-constraints.
- All SQL CREATE TABLE/VIEW commands used to build the database. These statements must include all the applicable FOREIGN KEY and CHECK statements.
- Source code. Please use **4 pages per sheet**, if you can. (On Unix, *mpage* can put 4 pages on 1 sheet of paper.)
- For each query listed in Section 4, the document must supply the SQL statement underlying the query.
- All SQL queries must be listed in a conveniently readable form. Please note that a dump of your program’s code that contains the requisite queries is **not acceptable**.
- A brief user guide. Explain how to install and run your program.

You will submit the documentation to the TA during the demo session where you will be showing your project. Demo sign-up sheets will be posted.

6 Teaming

You can choose one partner to do the project **jointly**. You can also choose to work alone, but this will not reduce your scope of work.

There are pitfalls in working with a partner whom you do not know well: it can be a frustrating experience to find out that your partner is piggy-backing on you. So, choose your partner carefully—we will **not** mediate disputes.

The division of work between partners must be **vertical**, not horizontal. This means that the two partners must collaborate on each part of the project and be on top of what the other partner is doing. Each student must be aware of and understand the techniques used by his/her partner. **Your final document must clearly state who did what part of the job. We will question your code.**

7 Planning Your Work

You have to plan the work on your own. At the end of the project, you will be asked to present a short (15 minute) demo to the TA (there will be sign-up sheets).

Your first step should be to produce an Entity-Relationship design and the ER diagram for the admission system. The diagram must then be translated into the object-relational model. This diagram and the translation should eventually become part of your project documentation.

The next step is to create your database and populate it with sample data. Then you should write and test the queries. Following that, start designing the forms for display and input of the data. Finally, write and test the code needed to produce these forms and connect them to your database queries.

We will provide an initial test data set. However, you might need to add more information to ensure that your queries have enough data to chew on.