# IS 420: Database Application Development
# Spring 2025
Group Project

## Overview

You will be assigned into groups of four to five people in this project. Please read the whole document carefully before starting your project. Your assignment is to design a conference management system. You will design the database, insert some sample data, and implement a set of required features in the backend. Each feature will be implemented as one Oracle PL/SQL procedure. You do **NOT** need to write a graphic user interface. You can test your features using SQL or PL/SQL scripts to call the implemented procedures.

## Assumptions:

You can make the following assumptions in this project.

1. The system stores information about institutions (universities, companies, government), including an institution id, institution name, and country)

2. The system stores information about users, including a user ID, institution id, name, address, zipcode, email, and country. Each user is also affiliated with exactly one institution.

3. The system stores information about conferences. Each conference has a conference ID, a conference title, a conference year, and the start and end date of the conference, submission due time (a timestamp), review due time, camera ready due time, a city and country, early registration date, early registration fee, regular registration fee.

4. The system stores roles a user has at a conference. The roles can be organizer, author, reviewer, participant. A user can have multiple roles at a conference. Please store user id, conference id, and role.

5. The system stores information about a paper. A paper has a paper ID, a title, a conference ID (the conference the paper is submitted to), submit time, average review score, and status (submitted, under review, accepted, rejected, camera ready received).

6. Each paper has one or more authors, each author is a user who has an author role for that conference. Each author can submit one or more papers to a conference. Please store paper id, author's user id, and author order (1 means first author and 2 second author and so on)

7. The system stores a list of topics, including a topic ID and a topic name.

8. Each paper is associated with one or more topics. Each topic is associated with one or more papers.

9. Reviewers are assigned to review a set of papers submitted to a conference. Each review consists of a review id, user id of the reviewer, paper id, review score, comments, review upload time.

10. A user can register for a conference. The system stores registration id, conference id, user id, conference id, registration fee, and payment date and payment status (paid or not).

11. The system stores a message table which contains message ID, user ID, message time, and message body.

**Features:** There are five individual features and five group features. Individual features will be graded individually (your group member's individual feature will have no impact on your grade), but group features will be graded group-wise. So each group should work together to make sure the group features are done correctly.

Each member needs to implement one individual feature. So if your group has X members your group will implement **any** X individual features. Each group also needs to implement **any** X group features. For example, if your group has five members, your group will do all 10 features. If your group has four members, your group will implement four individual features plus four group features.

**Individual features: (one feature per member)**

**Feature 1:** add a user. Input includes user name, institution id, address, zipcode, email, and country. This feature does the following:

1) It checks whether any user with the same email exists. If so, it prints a message 'the user already exists' and updates address, zipcode and country.

2) If no such user exists, it generates a new userID (using sequence) and inserts a row into the users table with a new user id, input name, institution id, address, zipcode, email, and country. Please also print out the new user ID.

**Feature 2:** add a list of roles of a user at a conference. The input includes a user id, a conference id, and an array of roles (using varray data type). This procedure does the following:

1) First check whether the conference id and user id are valid (i.e., there is a user with that user id and a conference with that conference id). If not print a message invalid user or conference ID.

2) Next for each role in the input array, check whether the user already has that role for the conference. If so, print a message 'role already exists'. Otherwise, insert a role to the table that stores user roles at each conference with the input user id, conference id, and role.

**Feature 3:** List papers submitted to a conference on a topic. Input includes a conference id and topic id.

1) The procedure first checks whether the input conference ID and topic ID are valid. If not, it prints out a message saying invalid conference ID or topic ID.

2) If both IDs are valid, it will find papers submitted to the input conference and on the input topic, and print out each paper's paper ID and paper title.

**Feature 4:** List reviews of a paper. Input is a paper ID. This feature does the following:

1) check whether there exists a paper with the input paper ID. If not, print a message 'Invalid paper ID' and stop.
2) if the paper ID is valid, print out each review of this paper, including reviewer name, reviewer's institution name, and review score. If the review score is null, print 'score missing'.

**Feature 5:** Update status of a paper and compute average review score. Input: paper ID, a new status. This feature does the following:

1) The procedure first checks whether the paper ID is valid. If not, it prints a message 'Invalid paper ID' and stops.

2) It then computes the average review score of all reviews for that paper, and updates both the review score and status of the paper to the new values.

3) It then inserts a message to the message table for each author of the paper, with a message time of the current time and, message body as 'Paper X: status is updated to Y' where X is the paper ID and Y is new status.

**Group features:**

**Feature 6:** Enter a review. Input: a user id, paper id, review score, review comment, upload time. This procedure does the following:

1) The procedure first checks whether the user id and paper id are valid. If not, print a message 'Invalid user id or paper id' and stop.

2) It then checks whether the user has a 'Reviewer' role in the conference the paper is submitted to. If not, print a message 'The user is not a reviewer.' and stop.

3) Next, it checks whether there is a conflict of interest for the user. A COI occurs if one of the authors of the paper is affiliated with the same institution as the user. If a COI exists, print a message 'A COI exists' and stop.

4) Check whether there is a row in the review table with the same review ID and user ID. If so, print 'Update existing review' and update the review score, review comment and upload time of the existing review with the same user id and paper id.

5) If there is no existing review, insert a new row into the review table with a new review id, and input user id, paper id, review score, comment and upload time. Print a message 'review added'.

**Feature 7:** Send a review reminder. Input: a conference ID and an input time. The procedure does the following:

1) it first checks whether the conference ID is valid. If not print a message 'Invalid conference ID';

2) it then finds all reviews for papers submitted to the conference and the review due time has passed but the review has not yet been uploaded. The review due time is passed means the review due time is earlier than the input time. The review is not uploaded if the review score or review comment is null.

3) for each past-due review found in step 2, insert a row to the message table with a newly generated message ID, user id as the user id of the review, and message time as the input time, and message body as 'Your review for paper X submitted to conference Y is past-due, please upload your review asap', where X is the paper ID of the review, and Y is the title of the conference. Please also print a message 'Sent reminder to user X for missing review for paper Y' where X is user id and y is paper ID.

**Feature 8:** Register a user for a conference. Input: user id, conference ID, current date, payment status.

1) The procedure first checks whether the user ID and conference ID are valid. If not, print a message invalid user ID or conference ID and stop.

2) It then checks whether there is a row in the registration table with the same conference ID and user ID as input. If so, print a message 'Update status of an existing registration', and update the payment status to the input status.

3) If there is no existing registration, check whether the input date is after the early registration date of the associated conference. If so, use the regular registration fee for the conference. Otherwise, use the early registration fee.

4)insert a row into the registration table with a newly generated registration ID, the input conference ID, the input user ID, the appropriate registration fee (decided in step 3), the input date and input status.

5) insert a row into the massage table with a new message ID, the user ID as the input user ID, current time as message time, and the body 'You are registered with status X' where X is input status.

**Feature 9:** Send a reminder to authors of papers that have been accepted but none of the authors has registered for the conference (usually conferences require at least one of the authors of an accepted paper to register). Input: conference ID. The procedure does the following:

1) It first checks whether the conference ID is valid. If not, print a message 'Invalid conference ID' and stop.

2) It then goes through each accepted paper of the conference, and checks whether any author has registered for the conference.

3) For any paper that none of the authors has registered, it inserts a message for each author of the paper, with a newly generated message ID, the time as current time, and user ID as the author's user ID, and message body as

'Dear X, your paper Y has been accepted to Conference Z, please register by D', where X is user name, Y is paper ID, Z is conference title, and D is the early registration date.

**Feature 10:** print statistics. There is no input. The procedure does the following:

1) It first prints out the acceptance rate for each conference, including the conference title, conference acceptance rate.

The acceptance rate equals #of papers accepted by the conference divided by the number of papers submitted to the conference.

2) It then prints out the number of papers accepted for each conference, the number of users registered for the conference, along with the conference title.

3) It prints out for each conference, the number of papers submitted for each topic name (allow double counting in different topics), the number of papers accepted for each topic, and acceptance rate for each topic (#of accepted on the topic divided by the number of submitted).

4) For each user, it prints out user name, total number of reviews the user has completed (with a not null review score and review comment), total number of reviews the review has not completed (with null review score or review comment).

**Deliverables:**

1. 10%. Due 2/18. Project Management Schedule.

   a. Include team members and a timeline showing each phase of your project with its activities and time duration, for the entire effort.

   b. It is expected that every member should participate in all phases of the project.

   c. Please specify which feature is assigned to which member (for group features it is still possible to assign a lead for each feature).

   d. Activities should include system design, populating tables, writing code, testing code, running example queries, writing documents, preparing for presentation, etc. Smaller milestones shall be set for deliverable 3 and 4.

e. This deliverable will be graded based on whether items a) to d) are included and whether the schedule is reasonable (e.g., enough time such as 2-3 weeks are left for testing and integration).

2. 25%. Due 3/25. Design Document which includes the following:

    a. ER diagram of the database. You don't have to follow exact notations of the ER diagram, but need to show tables, columns, primary keys, and foreign key links.

    b. SQL statements to create database tables and to insert some sample data (at least 3 rows per table). Please include drop table and drop sequence statements before create table, create sequence and insert.

    c. Specification for each required feature. The specification should include a description of input parameters and output (usually printing a message), and a few test cases with sample input parameter values (normally there should be one normal case and a few special cases). You don't need to implement any of these procedures at this point.

3. 35%. Due 5/7. Presentation of database design and demonstration of all individual features plus two group features. You can finish the remaining group features by D4 deadline (5/20). The project demo will be online (through Webex). You can sign up for the time of presentation/demo through a google form. To demo each feature, you need to prepare a couple of test cases, usually one normal case and the rest as special cases. For each test case you need to be able to explain why your answer is correct (this can be typically done by showing some tables or screen output).

4. 30%. Due 5/20. Please upload the final code through blackboard. The code should include:

    a. Drop table and sequence statements to drop tables if they exist (remember to use cascade constraints).

    b. Create table statements and create sequence statements

    c. Insert statements

    d. Create procedure statements (with code for the procedures) for all features. Each feature can be implemented as one PL/SQL procedure (in the procedure you may call other procedures or functions). Please include some comments in your code explaining the major steps. You should use create or replace to avoid procedure name conflict.

    e. Test script to show that all your features work correctly. The script shall include some examples to test different cases. E.g., for feature 1, one example for new customer added (phone is not in database) and one example for existing customer. Please include:

        i. PL/SQL script to call the appropriate PL/SQL procedure for this feature. E.g., exec procedure-name(parameter values)

ii. Explanation of what should be the correct output. The output could be updated tables (you can have some select statement to show the updated tables), some print out, etc.

iii. Make sure you have tested your examples from beginning to end. Remember that database tables may have been changed in the process. So you may need to start with a clean database (i.e., right after you execute all the drop table, create table, and insert statements).

**Grading Guidelines**

What I look for while grading software code (deliverable 4):

1. Existence of code and whether all code can be compiled without any error.
2. Comments: Both descriptive and inline for every procedure/function
3. Software quality
    a. Whether it is correct (giving correct results).
    b. Whether it is complete and clear.
    c. Efficiency of code. You shall not use too many SQL statements, and you shall put as much work as possible in SQL. For example, if you can do a join, do not use two select statements and then do a join in your program.
    d. Whether it has considered all special cases such as whether a user has already registered in Feature 1.

Regarding the presentation of your project: Each student must participate in the project demonstration by presenting to the entire class some slides. You will be graded on:

1. Timeliness of presentation
2. Presentation Style
3. Demo (running the code)

For the demo, you will be graded on the following items:

1. Existence of tables and data. You need to have at least 3 rows in each table.
2. The correctness of features. This can be shown by checking whether the screen output is correct and the database has been updated correctly.

Each member of the team shall contribute more or less equally. It is unfair for a few members to do most of the work while others do less. You will be asked to evaluate your teammate's effort at the end of the project. The instructor will adjust the grade based on the evaluation. Normally if most of your teammates agree that you do not contribute at all or contribute too little (e.g., your group has 4 members and you contribute only 5%), you may lose up to 80% of your project grade. If your teammates agree that you contribute much more than anyone else (e.g., your group has 4 members and you contribute 40%), you may gain up to 20% of your project grade (but not exceeding 100% of project grade). A peer evaluation will be conducted at the end of the semester to determine the contribution of each team member.

Tips:

1. Work as a team. Each member can do individual features by yourself but should work on other parts of the project including group features as a team. This means do NOT miss group meetings, do not miss internal deadlines, and help each

other for tasks that belong to the whole group. You should also divide up group tasks fairly and according to everyone's strength.

2. Start early. Do not wait until last month to start coding. Do not wait until one week before the demo to start putting things together. Past experiences show that more than 50% of time shall be devoted to testing and putting things together.
3. Learn how to debug SQL and PL/SQL code. Most of the time the error is from the SQL part of your code. So you can test SQL part separately (e.g., by copy & paste the SQL statement in a cursor and replace PL/SQL variables/parameters with values). You can insert screen output statements to check intermediate results. Oracle also returns error messages and error code. You can google the error messages and error code to find possible causes. You may also use Oracle SQL Developer which allows you to insert breakpoints during debugging.
4. It is highly recommended to use SQL Developer rather than the web interface for the project.
5. Use homework, in class exercises, and programs in slides as templates of your PL/SQL program. For example, if you need to write a cursor, find a cursor example and use it as a starting point.
6. Make sure special cases are handled.
7. At demo time, different data in the database may lead to different results. So usually you will start with a standard database (with a fixed set of tables and rows), and keep track of the sequence of the demo (e.g., a course can only be scheduled if it has been added first).