**University System Documentation:**

## Table of Contents

1. Introduction
2. Database design
   1. Tables
   2. Normalization
3. SQL implementation
   1. Create tables
   2. Insert data
4. PL\SQL
   1. Creating functions & procedures
   2. Creating Seq-trigger pairs.
5. Automation Script
6. Java Application
   1. Explore the scenes.
   2. Explore code.

# Introduction:

This document outlines the design and implementation of a relational database for managing student, course, department, and grade information. The goal is to define a schema that ensures data integrity and to document the database design.

# Database Design:

This document has provided a comprehensive overview of the relational database schema for managing student, course, department, and grade information. The design adheres to normalization principles to ensure data integrity. The outlined schema will serve as a foundation for subsequent steps, including SQL script creation, data population, and the development of PL/SQL procedures and Java applications.

A diagram of a student

Description automatically generated

**Students Table**

The **Students** table stores information about students.

**st\_id:** Unique identifier for each student.

**st\_fname:** First name of the student.

**st\_lname:** Last name of the student.

**do**b: Date of birth of the student.

**address:** Address of the student.

**year\_level:** Current academic year of the student.

**dept\_id:** Foreign key referencing the Departments table.

**Departments Table**

The Departments table contains details about academic departments.

**dept\_id:** Unique identifier for each department.

**dept\_name:** Name of the department.

**dept\_loc:** Location of the department.

**dept\_manager:** Manager of the department.

**Courses Table**

The **Courses** table holds information about academic courses.

**crs\_id:** Unique identifier for each course.

**crs\_name:** Name of the course.

**credits:** Credits associated with the course.

**Grades** Table

The **Grades** table records student grades for each course.

**g\_id:** Unique identifier for each grade entry.

**st\_id:** Foreign key referencing the Students table.

**crs\_id:** Foreign key referencing the Courses table.

**grade:** Grade obtained by the student.

**feedback:** Feedback related to the student's performance.

**crs\_year:** Academic year in which the course was taken.

**SQL Script Documentation:**

The code for creating the database schema is in a file named **metadata.sql**

This SQL script is designed to create tables for a university-related database, specifically focusing on students, departments, courses, and grades. Below is an example for table creation :

A computer screen shot of a computer code

Description automatically generated

**PL/SQL Function Documentation:**

Creating some functions and procedures to help organize code by grouping related tasks together. This enhances code readability and makes it easier to maintain and troubleshoot.

* The **calculate\_gpa** PL/SQL function is designed to convert numerical grades into the corresponding GPA (Grade Point Average) according to a standard grading scale. Below is a detailed documentation:
* **97 and above:** 4.0
* **93 to 96:** 4.0
* **90 to 92:** 3.7
* **87 to 89:** 3.3
* **83 to 86:** 3.0
* **80 to 82:** 2.7
* **77 to 79:** 2.3
* **73 to 76:** 2.0
* **70 to 72:** 1.7
* **67 to 69:** 1.3
* **65 to 66:** 1.0
* **Below 65:** 0.0
* The **update\_all\_gpa** PL/SQL procedure is designed to update the GPA of all students based on the average grades they have achieved. Below is a detailed documentation:

The purpose of this procedure is to recalculate and update the GPA of all students in the database based on the average grades they have obtained in their courses.

Example Usage:  
A screenshot of a computer

Description automatically generated

* The **getBestGpa** PL/SQL function is designed to retrieve the first name of the student with the highest GPA in a specific year level. Below is a detailed documentation:

The purpose of this function is to find and return the first name of the student with the highest GPA in a given academic year level.

Example Usage:  
A screen shot of a computer program

Description automatically generated

Seq-trigger pair:

**Sequence: ITI.GRADES\_SEQ**

**Purpose:** The sequence **ITI.GRADES\_SEQ** is designed to generate unique values for the primary key column **G\_ID** in the **ITI.GRADES** table.

**Details:**

* **Start Value:** The sequence starts with the value 131.
* **Maximum Value:** The maximum value is set to a very high number to accommodate a large number of unique values.
* **Minimum Value:** The minimum value is set to 1.
* **No Cycle:** The sequence does not cycle back to its start value.
* **Cache:** The sequence preallocates and caches 10 values for better performance.
* **No Order:** The sequence may not generate values in order of request.

**Trigger: ITI.GRADES\_TRG**

**Purpose:** The trigger **ITI.GRADES\_TRG** is associated with the **BEFORE INSERT** event on the **ITI.GRADES** table. Its purpose is to automatically populate the **G\_ID** column with the next value from the **ITI.GRADES\_SEQ** sequence.

**Details:**

* **Event:** The trigger is fired before an **INSERT** operation on the **ITI.GRADES** table.
* **Referencing Clause:** The **REFERENCING** clause is used to reference the new and old values of the rows being affected.
* **Trigger Body:** In the trigger body, the **:new.G\_ID** is assigned the next value from the **ITI.GRADES\_SEQ** sequence.

(AUTOMATION SCRIPT)

DISK MONITORING - Bash Script to monitor the Hard Disk usage and send alerts in case a specific threshold is exceeded => in a file named backup.sh. - The log.log file contains the disk monitoring and the backup logs.

DATABAS BACKUP - Perform a full backup of the database => the script is in a file named backup.sh. - The script.bat file is used by windows task scheduler to run the bash script.

A screenshot of a computer program

Description automatically generated

Java Application:

In my project I provided the **Java Source Code:**

This folder contains the source code for the Java application, enabling you to use any code editor for building and running the application.

* **Client:** Contains classes shared across applications (DTOs).
* **database:** Houses the Singleton class managing the Database Connection.
* **gui:** Encompasses code for all front-end (GUI) classes, including the source code for the application's backend.
* **Images:** Stores assets (pictures) used by the application.

**The Application contain 5 main scenes:**

* 1. Report & Homepage:

This section provides comprehensive information and analysis concerning students, courses, departments, and student GPAs within the application.

A screenshot of a computer

Description automatically generated

* 1. Search Students:

This scene offers detailed information on specific students, allowing users to view their complete profiles. Additionally, users can seamlessly enroll students in courses and grade those courses, all within the same interactive interface.

A screenshot of a computer

Description automatically generated

* 1. All Students:

This scene serves as a central hub for accessing information on all students. Users can not only view existing data but also conveniently insert new students directly within the same interface. Additionally, it features the "Update Students GPA" function, seamlessly connecting with a database procedure for maintaining accurate student records.

A screenshot of a computer

Description automatically generated

* 1. Departments:

This scene provides a comprehensive display of departmental data, offering insights into various aspects related to each department within the application.

A screenshot of a computer

Description automatically generated

* 1. Courses:

Within this scene, you can explore detailed information about courses, including relevant course data and the average grade associated with each course. This feature provides a quick overview of the performance metrics for individual courses within the application.

A screenshot of a course

Description automatically generated

Having explored the Scenes in detail, let's now delve into the implementation by providing code snippets from the data access layer. This will facilitate a deeper understanding of the integration between the scene and the underlying data functionality.

A screenshot of a computer program

Description automatically generated

A white background with text

Description automatically generated

A screen shot of a computer

Description automatically generated