CSE 4410

DATABASE MANAGEMENT SYSTEMS II LAB

SWE 21

CSE
IUT

Contents

Lab 1 Introduction

Welcome to CSE 4410.

1 Marks Distribution

Module	Mark (%)
Attendance	10
Lab Evaluation	40
Lab Report	20
Project	30

2 Approximate Course Outline

1. (Intro) + Basics of Relational Database Model
2. Tablespace
3. JDBC Connection + (Project Proposal Submission)
4. PL/SQL
    a. Function/Procedure
    b. Cursor
    c. Trigger
5. Project Progress Presentation
6. NoSQL [MongoDB]
    a. Theory
    b. Sessional
7. Graph-based Database [Neo4j]
    a. Theory
    b. Sessional
8. Project Presentation

3 Task - Group B

Consider the schema shown in Figure 1.1 for the database of a university:

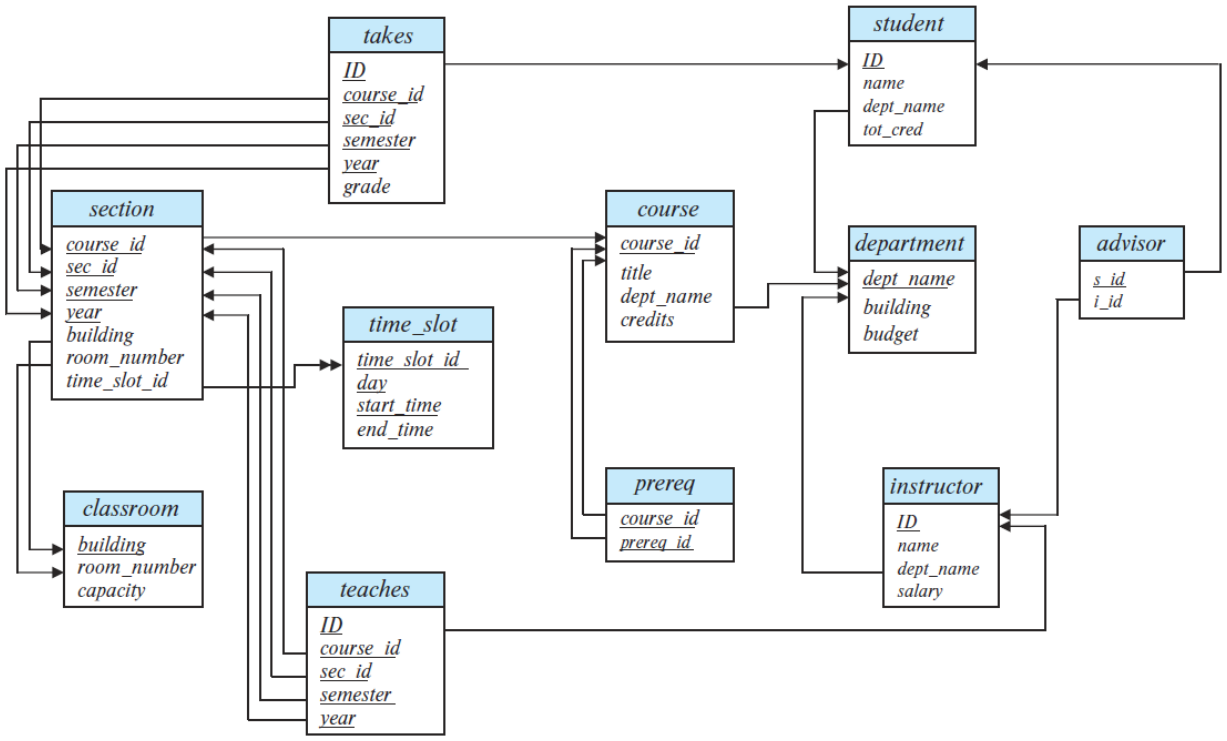

Figure 1.1. Schema diagram for a university database

Write the command `@"<file_path>\<file_name>.sql"` in your SQL command line to execute the provided .sql files. Now, write SQL statements to answer each of the following queries:

1. Find the names of all the instructors from the 'Biology' department.
2. Show the Course ID and the Title of all the courses registered for by the student with ID '12345'.
3. Find the names and department names of all the students who have taken a course offered by the 'Comp. Sci.' department.
4. Find the names of the students who take the 'CS-101' course in 'Spring, 2018'.
5. Find the names of students who have taken the highest number of courses with a specific prefix 'CS'.
6. Find the names of students who have taken courses taught by at least three different instructors
7. Find the course name and section having the minimum number of enrollments. Do not include the sections that do not have any students enrolled.
8. Find the name of the instructor, dept_name, and count of students he/she advising. If an instructor is not advising any student, show 0.
9. Find the name and department of the students who take more courses than the average number of courses taken by a student.
10. Insert each instructor as a student with total credit set to 0 in the same department they are teaching.
11. Remove all the newly added students from the previous query.
12. Update the 'tot_cred' for each student based on the credits taken.
13. Update the salary of each instructor to 10000 times the number of course sections they have taught.
14. Grades are mapped to a grade point as follows: A:10, B:8, C:6, D:4, and F:0. Create a table to store these mappings, and write a query to find the Credit Point Information (CPI) of each student, using this table. Make sure students who have not got a non-null grade in any course are displayed with a CPI of null.

4 Task - Group A

Consider the schema shown in Figure 1.2 for the database of a university:

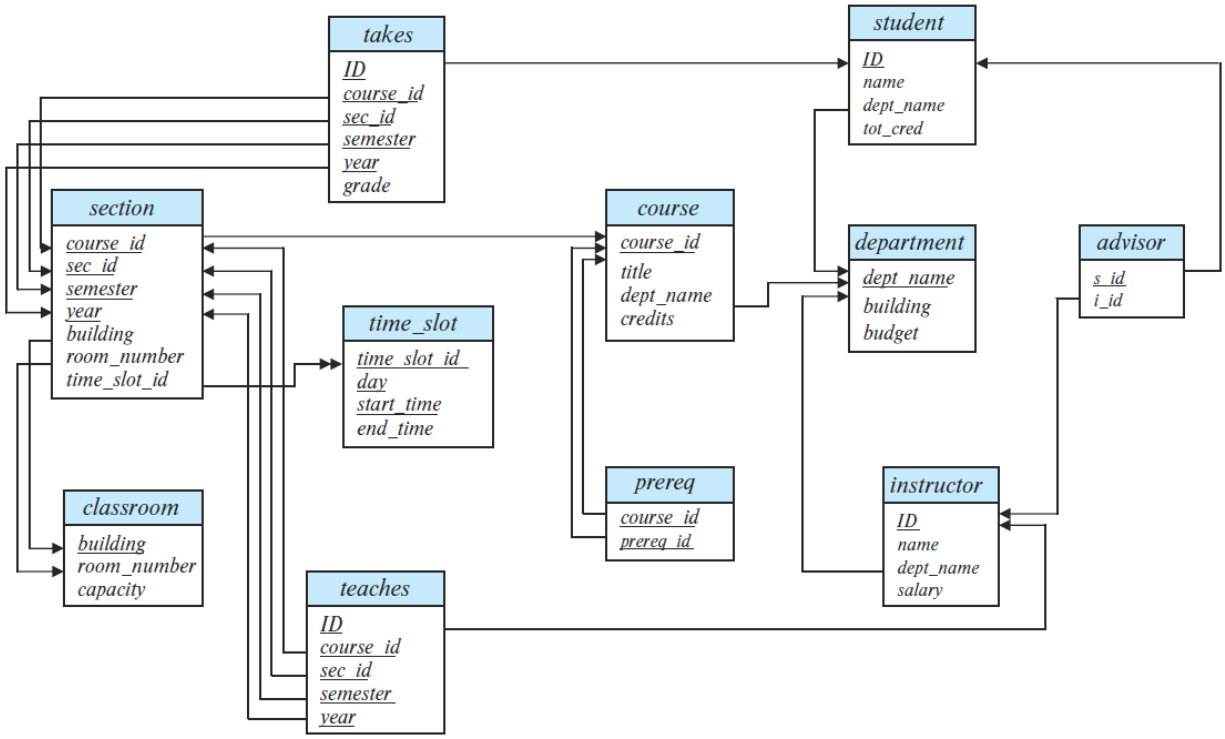

Figure 1.2. Schema diagram for a university database

Write the command `@"<file_path>\<file_name>.sql"` in your SQL command line to execute the provided .sql files. Now, write SQL statements to answer each of the following queries:

1. Find the names of courses offered by the 'Comp. Sci.' department which has 3 credits.
2. For each student, list their ID, name, and total credits s/he has taken. Do not include the students who did not register for any course.
3. Find the names and the department names of all instructors who have not taught a course.
4. Find all the course titles that do not have any prerequisites.
5. Find the name of the student who takes 2nd, 3rd, and 5th maximum total credits.
6. Find the names of the instructors who are taking courses with no students enrolled. Also, show the name of the courses.
7. Retrieve the course titles and the percentage of students who earned an 'A' grade in each course.
8. Find the number of instructors who have taught the same course in consecutive years.
9. Insert each student as a student with total credit set to 0 in the same department they are teaching.
10. Update the 'tot_cred' for each student based on the credits taken.
11. Update the salary of each instructor to 10000 times the number of course sections they have taught.
12. Find all rooms that have been assigned to more than one section at the same time.
13. Create a view that will show the instructor-wise time slot for 'Fall, 2017' sorted by the instructor_Id, course_ID, section_ID (Instructor_ID, name, his/her course information, section_ID, count of students in that section for the course, and time_slot).

Lab 2 Tablespace

A tablespace is a database storage unit that groups related logical structures together. The database data files are stored in tablespaces. A data file physically stores the data objects of the database such as tables and indices on disk. Using multiple tablespaces allows more flexibility in performing database operations.

- To enhance performance, segregate user data from data dictionary data. This minimizes conflicts over I/O resources.
- Prevent cross-application impact by keeping the data of each application separate. This safeguards against disruptions if a tablespace needs to be temporarily disabled.
- Enhance efficiency by storing data files from different tablespaces on different disk drives. This reduces contention for I/O resources.
- Increase overall system availability by selectively taking individual tablespaces offline while keeping others operational. This minimizes disruptions.
- Tailor tablespace allocation to specific database needs, such as high update activity, read-only activity, or temporary segment storage. This optimizes overall database performance.
- Enhance data security and recovery capabilities by conducting backups at the tablespace level. This facilitates targeted recovery and maintenance efforts.

Some operating systems set a limit on the number of files that can be opened simultaneously (For example, 512 in Windows, 1024 in Ubuntu, 4096 in CentOS, etc.). Efficient tablespace planning is essential to avoid surpassing the operating system limit. It is advisable to create tablespaces based on actual needs, keeping the number of tablespaces to a minimum. When expanding a tablespace, rather than creating numerous small data files, adding one or two substantial data files or opting for data files with autoextension enabled is recommended.

1 Default Tablespaces

Oracle comes with 5 default tablespaces:

- The primary tablespace in any database is the SYSTEM tablespace, which contains information basic to the functioning of the database server, such as the data dictionary and the system rollback segment. The SYSTEM tablespace is the first tablespace created at database creation. It is managed as any other tablespace but requires a higher level of privilege and is restricted in some ways. For example, it is not possible to rename or drop the SYSTEM tablespace or take it offline.
- The SYSAUX tablespace, which acts as an auxiliary tablespace to SYSTEM tablespace, is also created during database creation. It contains the schemas used by various Oracle products and features so that those products do not require their own tablespace. The management of the SYSAUX tablespace is similar to that of the SYSTEM tablespace.
- USERS is a permanent tablespace containing the application data. Oracle fills this space with the data created and entered by the users.
- UNDOTBS1 is an auto-extending tablespace containing the undo data. Oracle provides a fully automated mechanism, referred to as automatic undo management, for managing undo information and space. With automatic undo management, the database manages undo segments in an undo tablespace.
- TEMP is the temporary tablespace that is used for storing intermediate results. Oracle uses it as work areas for tasks such as sort operations for users and sorting during index creation. Oracle does not allow users to create objects in a temporary tablespace. By definition, the temporary tablespace holds data only for the duration of a user's session, and the data can be shared by all users.

2 Create Tablespace

Using **CREATE TABLESPACE** statement, a new tablespace can be created. As we have seen a tablespace consisting of one or more data files, we need to specify the path of the data files as well as their size.

```
CREATE TABLESPACE TBS1
   DATAFILE 'tbs1_data.dbf' SIZE 1M,
            'tbs2_data.dbf' SIZE 1M;
```

To allow extent management to be `LOCAL`, one can optionally add the statement `EXTENT MANAGEMENT LOCAL AUTOALLOCATE` or `EXTENT MANAGEMENT LOCAL UNIFORM SIZE size`. Locally Managed Tablespaces (LMT) have a bitmap of the blocks or groups of blocks, allowing them to track extent allocation without reference to the data dictionary. If `UNIFORM` is specified, all extents within the tablespace will be the same size, with `1M` (which stands for 1MB) being the default extent size. The `AUTOALLOCATE` clause allows you to size the initial extent leaving Oracle to determine the optimum size for subsequent extents, with 64K being the minimum.
Once the tablespace is created, all the information about it is available in the `DBA_DATA_FILES` view.

```
SELECT TABLESPACE_NAME, FILE_NAME, BYTES/1024/1024 MB
    FROM DBA_DATA_FILES;
```

To assign a user to a specific tablespace, one can explicitly mention it at the time of user creation.

```
CREATE USER iutlearner
    IDENTIFIED BY test123
        DEFAULT TABLESPACE TBS1;
```

Usually, when we create any new table in Oracle, by default, that is placed in the `USERS` tablespace. However, to create a new table in a user-defined tablespace, one must add the name of the tablespace at the end of the **CREATE TABLE** statement.

```
CREATE TABLE T1
(
   ID INT,
   C1 VARCHAR2(32)
) TABLESPACE TBS1;
```

Then, if we want to check the free space of a certain tablespace, we can fetch that data from the `DBA_FREE_SPACE` view.

```
SELECT TABLESPACE_NAME, BYTES/1024/1024 MB
    FROM DBA_FREE_SPACE
        WHERE TABLESPACE_NAME='TBS1';
```

3 Extend Tablespace

Commonly, the tablespaces of the database get completely occupied. In that case, no further addition of data is possible. We have already learned about locally managed extent by the time on tablespace creation. But even if we forget to do that or do not want to automate that we can manually handle it using **ALTER TABLESPACE** statement. There are two ways of extension:

▹ By adding a new data file.

```
ALTER TABLESPACE TBS1
    ADD DATAFILE 'tbs3_data.dbf' SIZE 1M;
```

Here, if we use the **AUTOEXTEND ON** clause at the end of the code, Oracle will automatically extend the size of the data file as needed.

▹ By resizing the data file

```
ALTER DATABASE
    DATAFILE 'tbs1_data.dbf' RESIZE 15M;
```

4 Drop Tablespace

To remove a tablespace from the database, we use **DROP TABLESPACE** statement.

```
DROP TABLESPACE TBS1
    [INCLUDING CONTENTS [AND | KEEP] DATAFILES]
    [CASCADE CONSTRAINTS];
```

Here **INCLUDING CONTENTS** is necessary when there is any table created in the tablespace. Any attempt to remove a tablespace that has objects without specifying the clause will result in an error. If we do not use **AND DATAFILES**, it will by default keep the data files of those tables stored without any tablespace. And **CASCADE CONSTRAINTS** is necessary in case of referential integrity.
You can use the DROP TABLESPACE command to remove a tablespace regardless of whether it is online or offline. However, it is good practice to take the tablespace offline before removing it to ensure that no sessions are currently accessing any objects in the tablespace.

5 Read-Only or Read-Write Tablespace

The read-only tablespaces allow Oracle to avoid performing edits on large, static parts of a database. It allows you to remove objects such as tables and indexes from a read-only tablespace. However, it does not allow you to create or alter objects in a read-only tablespace.

```
ALTER TABLESPACE TBS1 READ ONLY;

ALTER TABLESPACE TBS1 READ WRITE;
```

By default, any newly created tablespace is in read-write mode.

6 Online and Offline Tablespace

Lastly, a tablespace can be online or offline. If a tablespace is offline, one cannot access data stored in it. On the other hand, if a tablespace is online, its data is available for reading and writing.

```
ALTER TABLESPACE TBS1 OFFLINE;

ALTER TABLESPACE TBS1 ONLINE;
```

Normally, a tablespace is online so that its data is available to users. However, we can take a tablespace offline to make data inaccessible to users when we update and maintain the applications.

7 Task - Group B

1. Create two tablespaces `tbs1` and `tbs2`.
2. Set quota for a single user on both tablespaces.
3. Create two tables `student(name, ID, fk[dept_ID])` and `department(ID, name)` in `tbs1`.
4. Create another table `course(course_code, name, credit, fk[offered_by_dept_ID])` in `tbs2`.
5. Insert a large amount of data in the `student` table and `course` table.
6. List the title and the name of the offering department of each course.
7. Check the free space of the tablespaces.
8. Extend `tbs1` by adding extra data files.
9. Extend `tbs2` by resizing data files.
10. Check the size of the tablespaces.
11. Set `tbs1` to offline and show that the data cannot be accessed.
12. Delete tablespace `tbs1` including the data files.
13. Delete tablespace `tbs2` excluding the data files.

8 Task - Group A

1. Create two tablespaces `tsp1` and `tsp2`.
2. Set quota for a single user on both tablespaces.
3. Create two tables `author(name, ID)` and `publisher(ID, name)` in `tsp1`.
4. Create another table `book(ISBN_code, name, price, fk[publisher_ID])` in `tsp2`.
5. Insert a large amount of data in the `book` table and `publisher` table.
6. List the title and the name of the publisher of each book.
7. Check the free space of the tablespaces.
8. Extend `tsp1` by adding extra data files.
9. Extend `tsp2` by resizing data files.
10. Check the size of the tablespaces.
11. Set `tsp1` to offline and show that the data cannot be accessed.
12. Delete tablespace `tsp1` including the data files.
13. Delete tablespace `tsp2` excluding the data files.

Lab 3 Java Database Connectivity (JDBC)

Java Database Connectivity (JDBC) is a Java API facilitating the interaction with relational databases. It provides a standard interface for connecting to databases, executing SQL queries, manipulating data, and handling the results. Prior to JDBC, ODBC was used but had dependencies on C-language drivers. Later JDBC introduced its own API, employing Java-based JDBC drivers.

The JDBC driver serves as a bridge, allowing the Java application to send queries, receive results, and generally interact with the database management system seamlessly. The driver essentially translates Java calls into a format that the database understands and vice versa, enabling effective and standardized communication between the Java application and the underlying database.

There are four types of drivers. The `JDBC-ODBC bridge driver`, the `Native-API driver` (partially Java driver), the `Network Protocol driver` (fully Java driver), and the `Thin driver` (fully Java driver). Among these, the Thin driver excels in performance, as it directly translates JDBC calls into the specific protocol of the associated database and does not require any supplementary software installation on either the client or server side.

1 Environment Setup

All the files mentioned below have been provided in Google Classroom.

1. Install Java Development Kit (JDK) using `jdk-19_windows-x64_bin.msi`.
2. If you want to use VSCode install the extension pack for Java.
3. Create a new Java Project and add `ojdbc14.jar` or `ojdbc6.jar` file as an external JAR file.
   a. For VSCode:
      from the Explorer Bar, go to `Java Projects` → `<projectname>` → `Referenced Libraries`. Then click on the `plus` sign and select the `ojdbc14.jar` or `ojdbc6.jar` and click the `Select Jar Libraries` button.
   b. For Eclipse IDE:
      from the Menu Bar, go to `Project` → `Properties` →. Then on the opened window, click on `Java Build Path` → `Classpath` → `Add External JARs`. This will open a File Explorer where you can navigate to the path where `ojdbc14.jar` or `ojdbc6.jar` is located to select and add it to your project.
   c. For IntelliJ IDE:
      from the Top Bar, go to `File` → `Project Structure` →. Then on the opened window, click on `Modules`. In the `Export` section, click on the `plus` sign → `JARs or Directories`. There provide the path to `ojdbc14.jar` or `ojdbc6.jar` and hit the `OK` button.

   If you are using any other IDE, Google "how to add external JAR files to <XYZ> IDE" to get help. Replace <XYZ> with the IDE that you are using.

2 Java Database Connectivity with Oracle

There are 5 steps to connect any Java application with the database using JDBC. These steps are as follows:

1. Register the Driver class
2. Create a connection
3. Create statement
4. Execute query
5. Close connection

Consider the Code Snippet 3.1 for connecting with JDBC step by step.

```
import java.sql.*;

class jdbc_practice {
    public static void main(String args[]) {
        try {
            // step1 load the driver class
            Class.forName("oracle.jdbc.driver.OracleDriver");

            // step2 create the connection object
            Connection con = DriverManager.getConnection(
                    "jdbc:oracle:thin:@localhost:1521:xe", "DBMS", "dbms");

            // step3 create the statement object
            Statement stmt = con.createStatement();

            // step4 execute query
            // drop table
            System.out.println("Drop table...");
            String sql = "DROP TABLE REGISTRATION";
            stmt.executeUpdate(sql);

            // create table
            String sql1 = "CREATE TABLE REGISTRATION " +
                    "(id INTEGER not NULL, " +
                    " first VARCHAR(255), " +
                    " last VARCHAR(255), " +
                    " age INTEGER, " +
                    " PRIMARY KEY ( id ))";

            stmt.executeUpdate(sql1);

            // insert table
            System.out.println("Inserting records into the table...");
            String sql2 = "INSERT INTO Registration VALUES (100, 'Zara', 'Ali',
   18)";
            stmt.executeUpdate(sql2);
            sql2 = "INSERT INTO Registration VALUES (101, 'Mahnaz', 'Fatma', 25)";
            stmt.executeUpdate(sql2);

            // select table
            System.out.println("Selecting records from the table...");
            String QUERY = "SELECT id, first, last, age FROM Registration";
            ResultSet rs = stmt.executeQuery(QUERY);
            while (rs.next()) {
                // Display values
```

```
                System.out.print("ID: " + rs.getInt("id"));
                System.out.print(", Age: " + rs.getInt("age"));
                System.out.print(", First: " + rs.getString("first"));
                System.out.println(", Last: " + rs.getString("last"));
            }
            rs.close();

            // update table
            System.out.println("Updating records from the table...");
            String sql3 = "UPDATE Registration " +
                    "SET age = 30 WHERE id in (100, 101)";
            stmt.executeUpdate(sql3);

            // delete table
            System.out.println("deleting records from the table...");
            String sql4 = "DELETE FROM Registration " +
                    "WHERE id = 101";
            stmt.executeUpdate(sql4);

            // step5 close the connection object
            con.close();

        } catch (Exception e) {
            System.out.println(e);
        }

    }
}
```

Code Snippet 3.1. Sample JDBC Connection code

Consider the following:

- Driver class: The driver class for the Oracle Database is "`oracle.jdbc.driver.OracleDriver`".
- Connection URL: The connection URL for the Oracle10G database is "`jdbc:oracle:thin:@localhost:1521:xe`" where `jdbc` is the API, `oracle` is the database, `thin` is the driver, `localhost` is the server name on which oracle is running (we may also use an IP address here), `1521` is the port number, and `XE` is the Oracle Service Name. You may get all these information from the `tnsnames.ora` file.
- Username: The default username for the Oracle Database is system.
- Password: It is the password given by the user at the time of installing the Oracle Database.

3 Some interfaces of JDBC API

The `java.sql` package contains classes and interfaces for JDBC API. Some popular interfaces and classes in the JDBC API include:

- `DriverManager: java.sql.DriverManager` is a class that manages a list of database drivers. It is used to establish a connection to the database by selecting an appropriate driver from the list.
- `Connection: java.sql.Connection` represents a connection to a relational database. It provides methods

for creating statements, committing or rolling back transactions, and managing other aspects of the connection.

- `Statement`: `java.sql.Statement` is an interface that represents an SQL statement. There are different types of statements, such as `Statement`, `PreparedStatement`, and `CallableStatement`, each serving a specific purpose. These are used to execute SQL queries and updates.
- `ResultSet`: `java.sql.ResultSet` represents the result set of a database query. It provides methods for iterating over the rows of the result set and retrieving data from each column.
- `PreparedStatement`: `java.sql.PreparedStatement` is a sub-interface of Statement. It is used to execute pre-compiled SQL queries with parameters. `PreparedStatement` is more efficient than `Statement` for executing queries repeatedly with different parameter values.
- `ResultSetMetaData`: `java.sql.ResultSetMetaData` is an interface that provides information about the columns of a `ResultSet`, such as the column names, types, and properties.
- `DatabaseMetaData`: `java.sql.DatabaseMetaData` provides methods to obtain metadata about the database, such as information about its tables, columns, and supported SQL features.

4 Task - Group B

Consider the schema shown in Figure 3.1 for the database of a university:

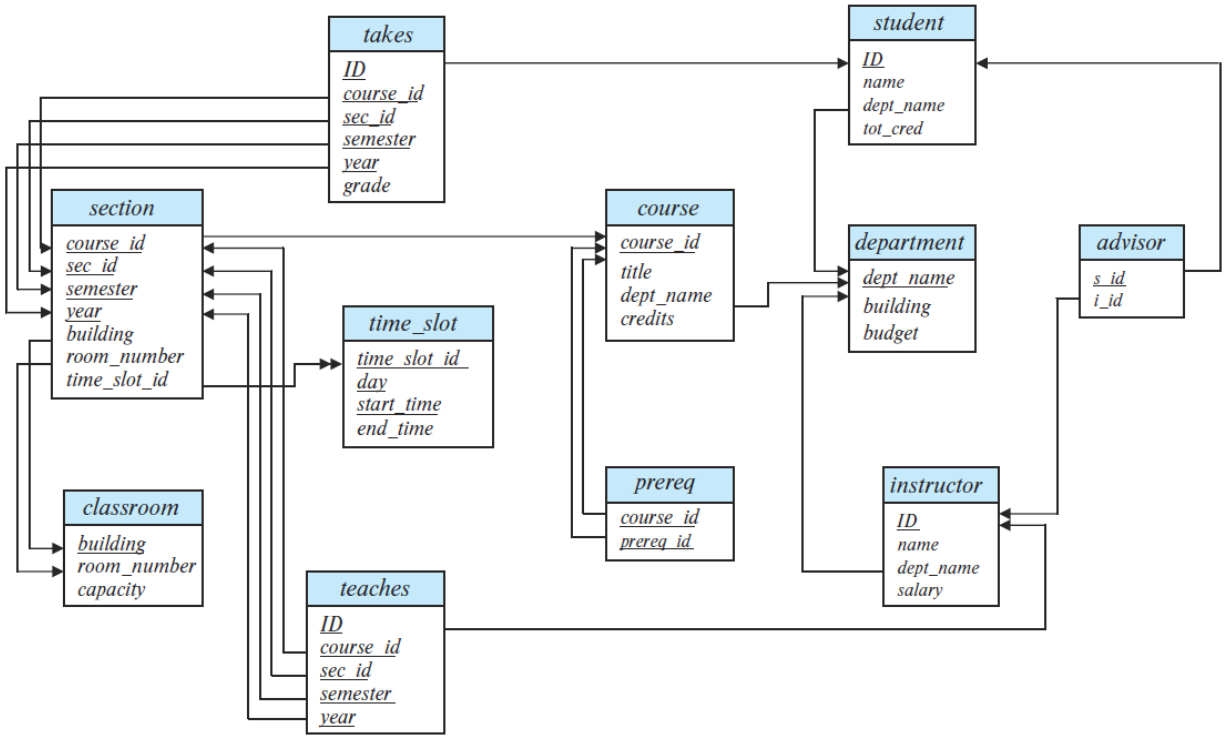

Figure 3.1. Schema diagram for a university database

Write the command `@"<file_path>\<file_name>.sql"` in your SQL command line to execute the provided .sql files. Now, write Java functions to perform each of the following tasks:

1. Decrease the budget of the departments having a budget of more than 99999 by 10%. Then show the number of departments that did not get affected.
2. Take the day of the week, starting hour, and ending hour as input from the user. Then print the names of the instructors who will be taking classes during that time.
3. Find the top N students based on the number of courses they are enrolled in. You should take N as input and print the ID, name, department name, and the number of courses taken by the student. If N is larger than the total number of students, print the information for all the students.
4. Insert a new student named 'Jane Doe' in the STUDENT table. The student should be enrolled in the department having the lowest number of students. The ID of the student will be $(X + 1)$, where X is the highest ID value among the existing students.
5. Find out the list of students who do not have any advisor assigned to them. Then assign them an advisor from their department. In case there are multiple instructors from a certain department, the advisor should be selected based on the least number of students advised. Finally, print the name of the students, the name of their advisor, and the number of students advised by the said advisor.