DV1663

Database Project Report

Adrian Adborn
adad21@student.bth.se

Viktor Listi
vili22@student.bth.se

1. Project Idea

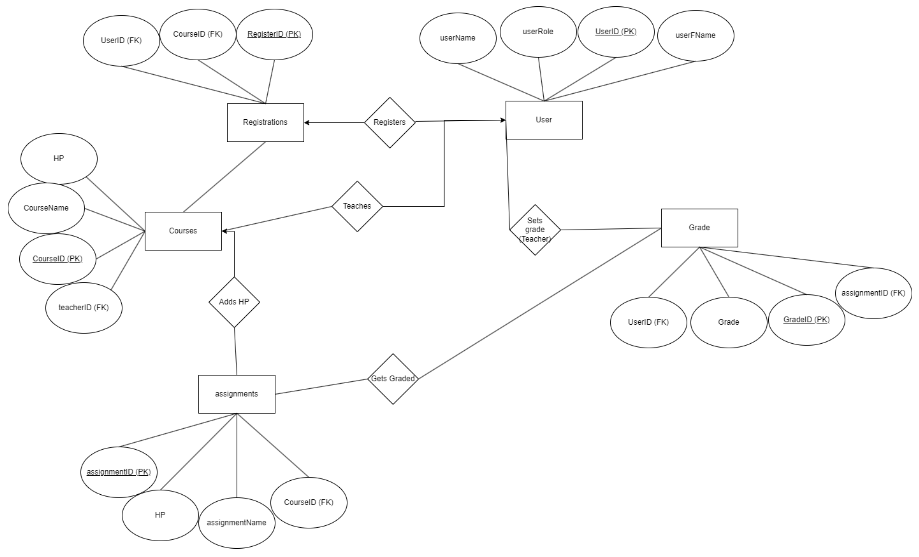
Our idea is to create a Ladok/Canvas like database/platform where courses, tasks, grades and hp get recorded inorder to easily access them at a later date. The application will have different roles with different responsibilities/privilege levels, where for example teachers will be able to create courses, tasks and set grades. Students will be able to see and access their courses and corresponding tasks. Users will “login” with a username which is unique for every user.

The main application will be written in python and use the Tkinter library as GUI , mySQL will be used for data storage and handling.

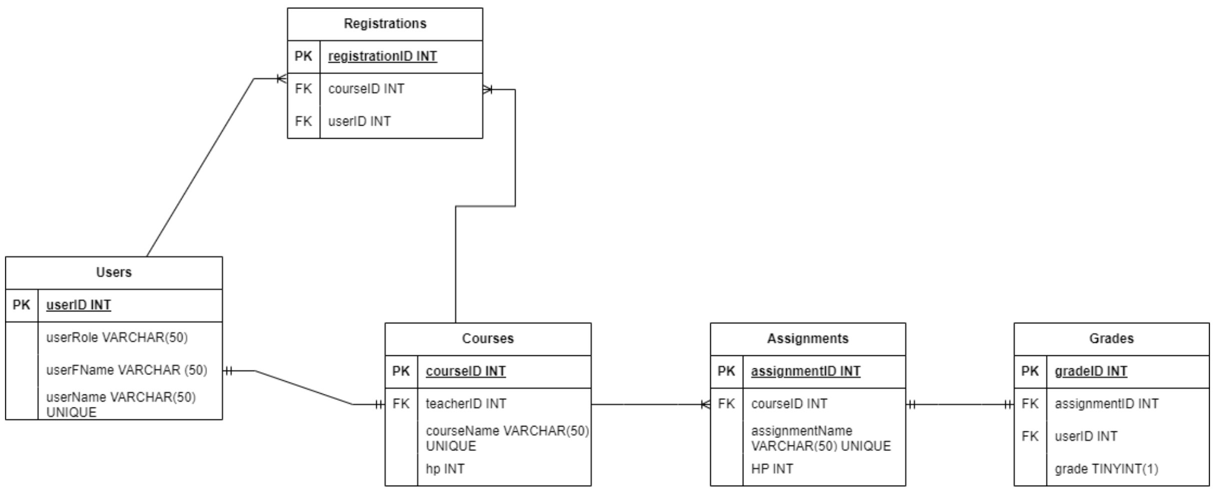
This application will help teachers and students keep track of their courses and assignments.

2. Schema Design

E/R Diagram



SQL Schema



In the SQL Schema which was converted from our E/R Diagram we have a structure containing five different tables. Firstly we have the ***Users*** table which is a table that is designed to store information about users in our system where we can register if the user is a teacher or student (or other role), their username and their actual name. This is the base for the rest of the tables as the userID primary key is used as a foreign key in many of the other tables. More information could have been added to the users table such as email, their program etc, but was not necessary for this implementation.
The second most important table is the ***Courses*** table which is used to create a new course, it stores the courseID as primary key which is an important foreign key in other tables as well as a userID of a user with the teacher role and the total amount of HP that the course consists of, this number is calculated based on the sum of hp points for all assignments linked to a certain course.
A user can be registered to a certain course via the ***Registrations*** table which stores a registrationID primary key which is not very important for the other tables but good to have, a courseID in the form of a foreign key and a userID in the form of a foreign key. This links a user to a course which is later used to check if a user attends certain courses and all the students in a certain course.
To the courses, assignments need to be added so that the students can have something to hand in. This is done via the ***Assignments*** table, this table has an assignmentID as a primary key which is used for adding grades, a courseID as a foreign key to check which course its linked to, an assignmentName and the total HP points that the assignment covers which is used to calculate the total HP points for a course as previously mentioned.
In order to add a grade to an assignment we need something to store the grades in, therefore we have constructed a ***Grades*** table which stores an assignmentID as a foreign key and a userID as a foreign key in order to keep track of which user the grade is for and what assignment the grade is for. The table also includes a column for the actual grade itself.

All of these five tables combined are the building blocks of a basic model for what we discussed in the project idea, and we believe that this is the most efficient SQL structure for achieving those goals.

3. SQL Queries

Q: Add a new user to the database.
This **procedure,** which is called add_user takes one input parameter fName representing the first name of the user to be added. It generates a username for the user based on fName and inserts the new record into the users table.

```
CREATE PROCEDURE `add_user`(IN fName VARCHAR(50), OUT ret BOOL)
BEGIN
DECLARE x BOOL;
DECLARE counter INT;
DECLARE uName VARCHAR(50);
SET x = FALSE;
SET counter = 0;
WHILE (!x) DO
```

```
      SET uName = concat(fName, counter);
    IF NOT EXISTS(SELECT 1 FROM users WHERE userName = uName)THEN
      SET x = TRUE;
    END IF;
    SET counter = counter + 1;
END WHILE;
INSERT INTO users (userFName, userName) VALUES (fName, uName);
END ;;
DELIMITER ;
```

Q: Update HP of a course after a new assignment is created.
This **trigger**, named hp_updater_insert, automatically updates the hp of a course whenever a new assignment is inserted into the assignments table. It makes use of the **aggregation** method **SUM()** in order to calculate the total amount of HP a course covers by adding all assignments linked to a course. This specific trigger is made for when an assignment is inserted, there is also an additional trigger for when an assignment is removed.

```
CREATE TRIGGER hp_updater_insert
AFTER INSERT ON assignments
FOR EACH ROW
UPDATE courses
SET courses.hp = (SELECT SUM(assignments.hp) FROM assignments WHERE
courseId = NEW.courseId)
WHERE courseId = NEW.courseId;
```

Q: Retrieve registered users for a specific course.
This **procedure,** which is called registrations_course takes one input parameter CourseName which represents the name of the course. It performs an **INNER JOIN** between the registrations and courses tables using the courseId foreign key. The **SELECT** statement fetches registrationId and userId columns from the registrations table based on the provided CourseName.

```
DELIMITER //
CREATE PROCEDURE registrations_course(IN CourseName VARCHAR(50))
BEGIN
    SELECT registrations.registrationId, registrations.userId
    FROM registrations
    INNER JOIN courses ON registrations.courseId = courses.courseId
    WHERE courses.courseName = CourseName;
END //

DELIMITER ;
```

Q: Retrieve courses taught by a specific teacher.
This **multirelational procedure,** which is called courses_teacher takes one input parameter teacherName representing the name of a user with the teacher role. It performs an **INNER JOIN** between the courses and users tables using the teacherId foreign key. The **SELECT** statement retrieves the courseName and hp columns from the courses table based on the provided teacherName.

```
DELIMITER ;;
CREATE PROCEDURE `courses_teacher`(IN uName VARCHAR(50))
BEGIN
    DECLARE uId INT;
    SET uId = (SELECT userId FROM users WHERE userName = uName);
      SELECT courseName, courseId, hp
      FROM courses
      WHERE teacherId = uId;

END ;;
DELIMITER ;
```

Q: Remove a course and its related data.
This **multirelational procedure,** which is called remove_course accepts one input parameter cName representing the name of the course to be removed. It removes all related data from registrations, grades, assignments, and the course tables.

```
DELIMITER //
CREATE PROCEDURE remove_course(IN cName VARCHAR(50))
BEGIN
    DECLARE cID INT;

    SELECT courseId INTO cID
    FROM courses
    WHERE courseName = cName;

    DELETE FROM registrations WHERE courseId = cID;

    DELETE FROM grades WHERE assignmentId IN (SELECT assignmentId FROM
assignments WHERE courseId = cID);

    DELETE FROM assignments WHERE courseId = cID;

    DELETE from courses WHERE courseName = cName;
END //
DELIMITER ;
```

4. Discussion and Resources

4.1 Discussion

Overall we believe that the project went well and the SQL portion of the project is complete. The python application may be lacking in some user interface features that would improve the user experience, this may include more menu choices so the user does not have to input the name of courses or assignments.

If the project was to be done again some things we would change are, more design of the whole system in the beginning of the project. A Lot of functions and procedures were thought of later which lead to things needing to be changed in already established functions. This would also make it easier to work as a team since variable names and such would be already decided.

4.2 Implementation

The project implementation uses MySQL and Python with the Inquirer library for user interface. The python program is only for user interface and also parsing the returns of some procedures on the database.

Check README.txt in github for installation and implementation details. All relevant/required files are in the MAIN branch. Branch v1 was used during development and contains separated sql files.

GITHUB:

5. Appendix

Changelog can be found via the commits section in v1 branch in the github repository.