Project report

Introduction

This project implements a test management system. The test management system is used by teachers and students of an educational institution to conduct timed tests. The teacher creates questions and stores them in the system along with their answers for different courses. The system maintains this collection of questions as a question bank and allows the teacher to create tests from these questions. Each test can contain 10 multiple choice questions which are selected by the teacher. Students can take one test at a time and the system grades the test. The students and teachers can view the grades scored in the different tests.

The test management system is implemented as a java application along with an MS SQL server database. The java application provides the user interface for the teachers and students to interact with the test management system. The database stores information about the teachers, and students, their courses, questions, tests and grades. The database uses stored procedures providing the relevant information to the java application and a trigger to grade a student test. We experimented with the query execution time by indexing the most frequently used table, users.

Working of the application

The test management system has two types of users, teachers and students. Both types of users need to first login to the system by providing their username, password and role type (teacher or student). The system verifies the user credentials using a stored procedure. On successful login, the user has to select the course of interest. The information about courses taught by a teacher and courses taken by a student is maintained in a different system like the course registration system and our test management system imports this data from it. Using the course information, the system displays the different courses available to the user. The user then selects the course of interest.

For teachers, the system allows them to add or modify questions, create tests and view all student grades for each test. The teacher can add a new question to the question bank of the selected course. This question should have four answer choices and the correct choice must also be mentioned. The question is stored in the database and each question is assigned a unique identifier called questionId. To modify an existing question, the teacher must choose the question to be modified. The questions they can choose from is shown to the teacher by a drop down box. The questions are shown in text not the questionId, this makes it easier for the teacher to see which question she or he wants to modify.

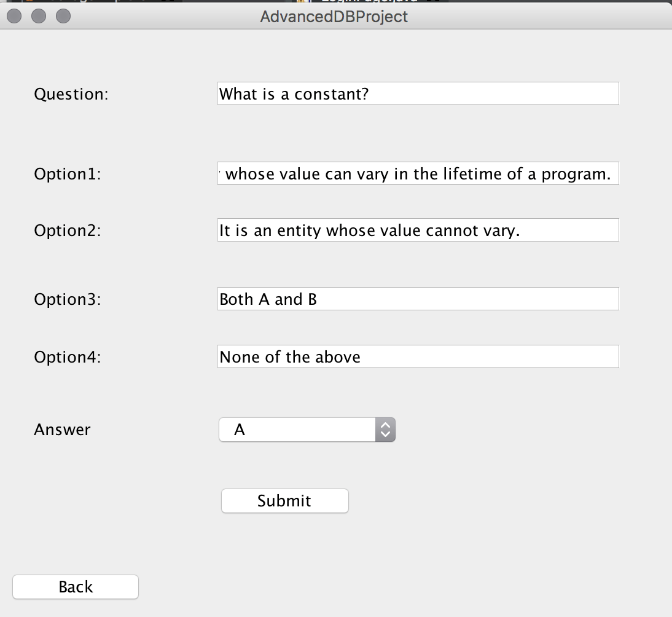


Figure 1 Teacher Adding Question

When a teacher has at least 10 questions in the question bank for a course she or he can make a test. Each test must have ten questions, a test name and the maximum time a student can take on the test (10, 30 or 60 minutes). When creating a test, the teacher will have 10 combo boxes, one for each question, that displays the same information as the modify question combo box which is the actual question itself.

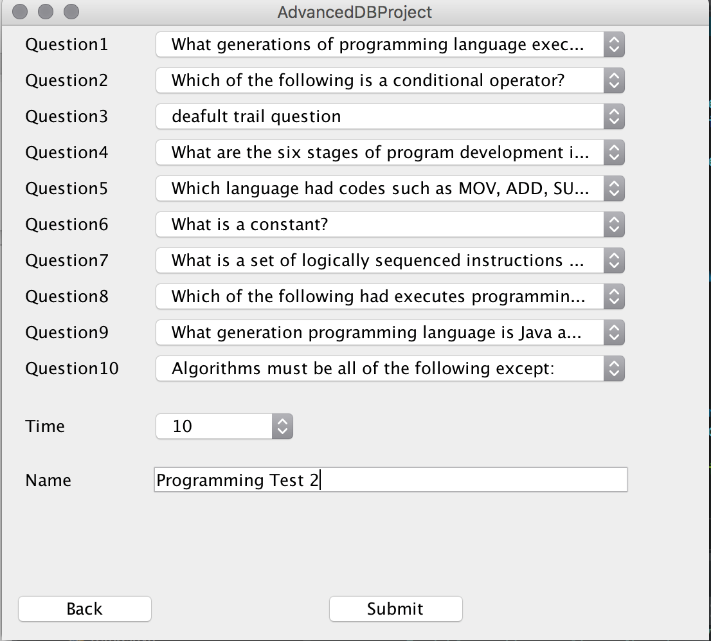


Figure 2 Teacher Adding a Test

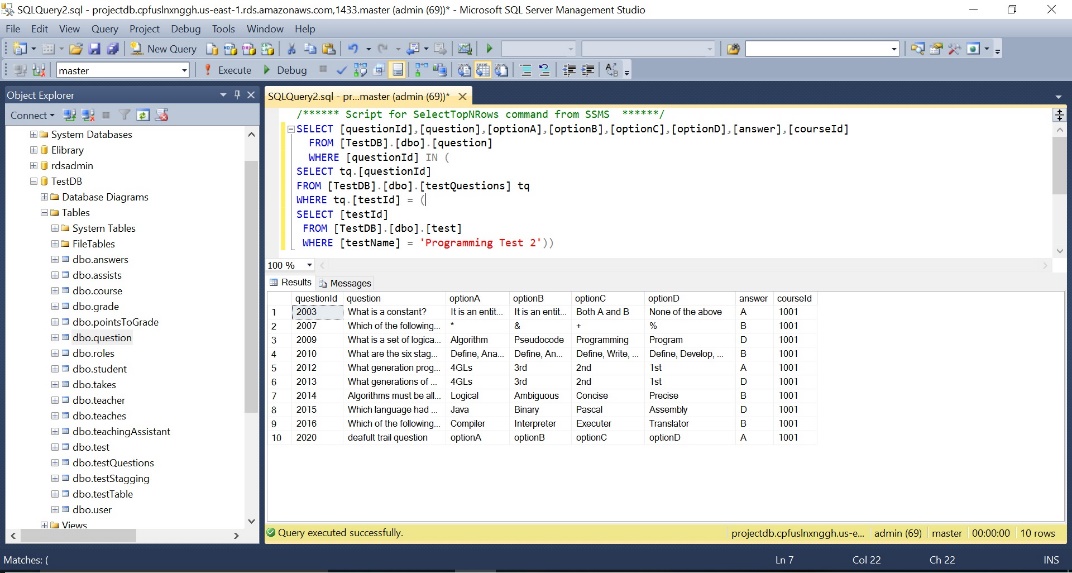


Figure 3 New Test in the Database

A teacher can also view the grades of the students in different tests for a course. The student grades will be displayed for each test.

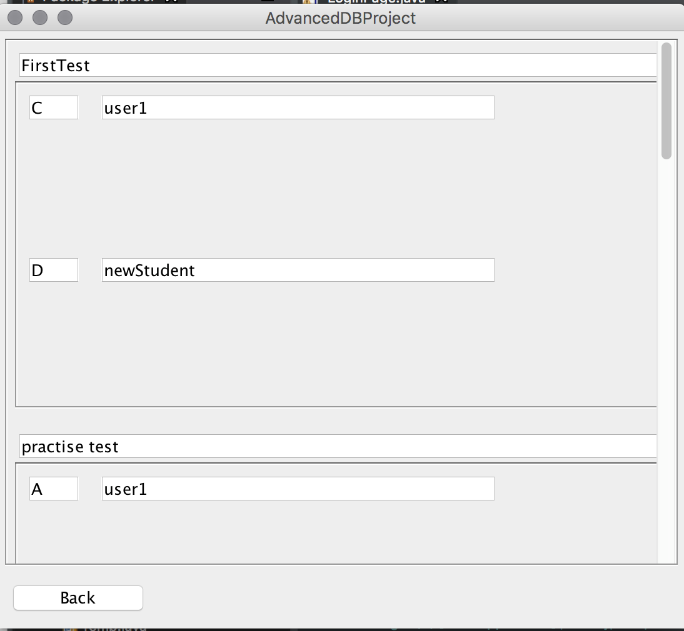


Figure 4 Teacher viewing student grades

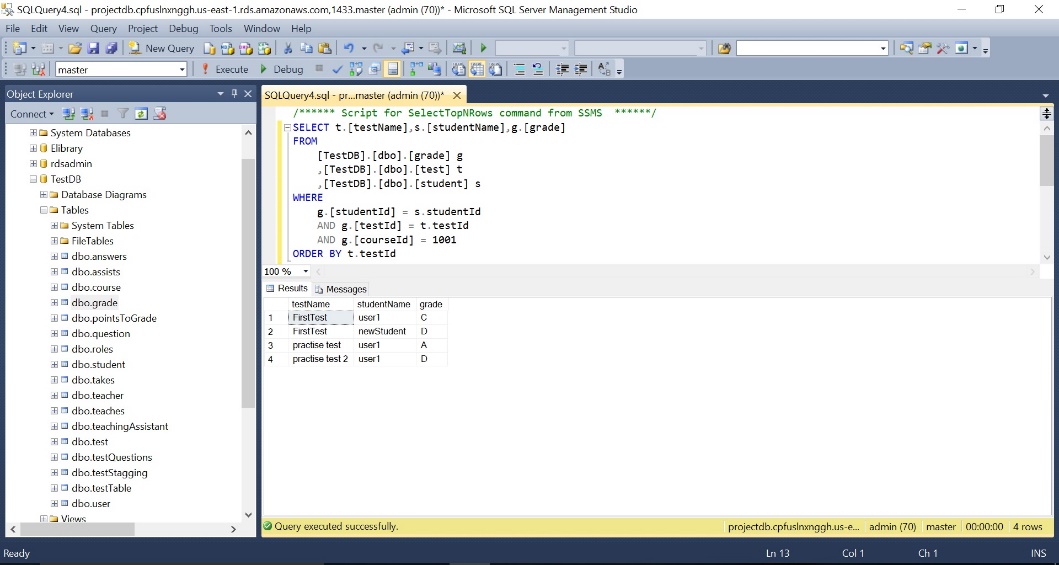


Figure 5 Student grades for all tests in a given course

For students, the system allows them to take a test or view their grades. A student can choose a test that the teacher has created for the specific course. Just like the teacher’s question combo box the student can see the name of every test that the teacher has made that the student has currently not taken. Once the student chooses which test she or he wants to take the system will retrieve each question for the test from the question bank using a stored procedure in the system. The student will have a certain time limit that the teacher has set to take the test. The test will be submitted either by the student or after a time out and it will be graded by the system using an SQL trigger.

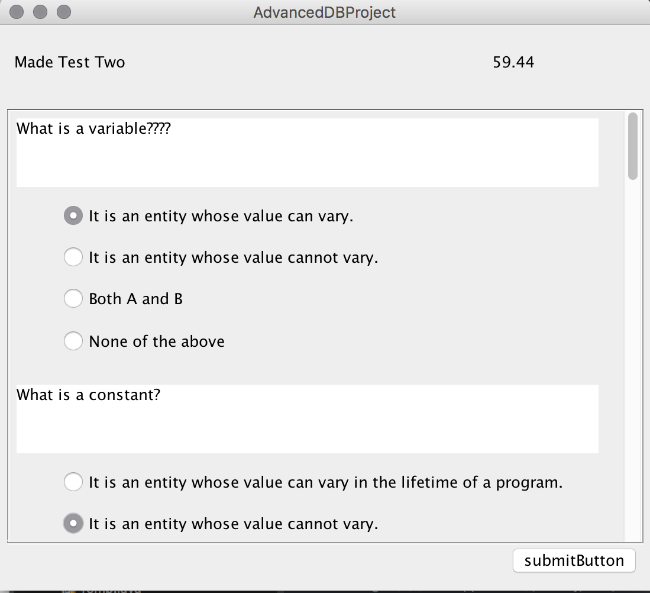


Figure 6 Student taking a test

Once the student has completed taking a test for a specific course she or he can view the grade for all tests that she or he has taken for that course. The grades will be shown under the specific test name with a letter grade received for the test (A, B, C, D, E or F).

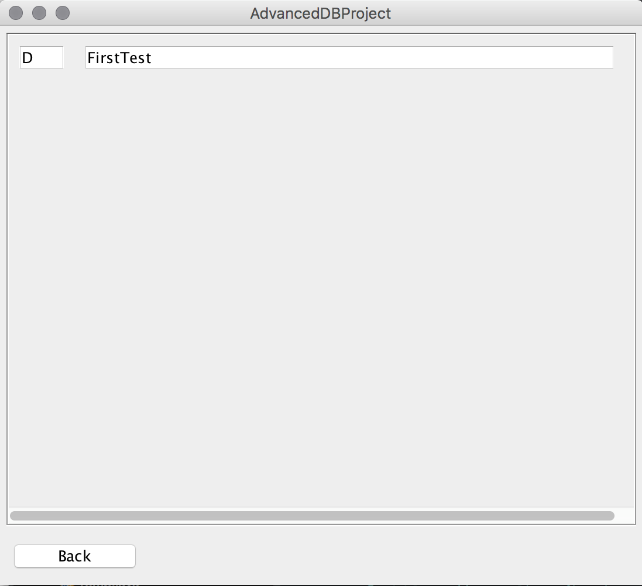


Figure 7 Student viewing his grades for all tests in a given course

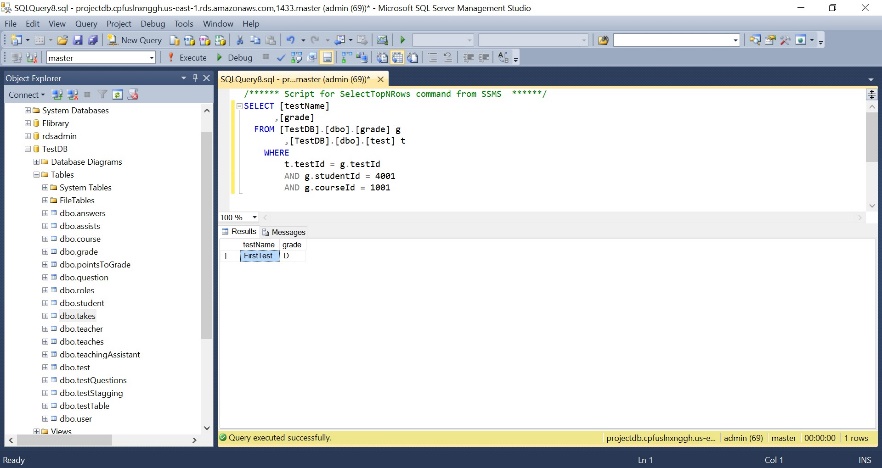
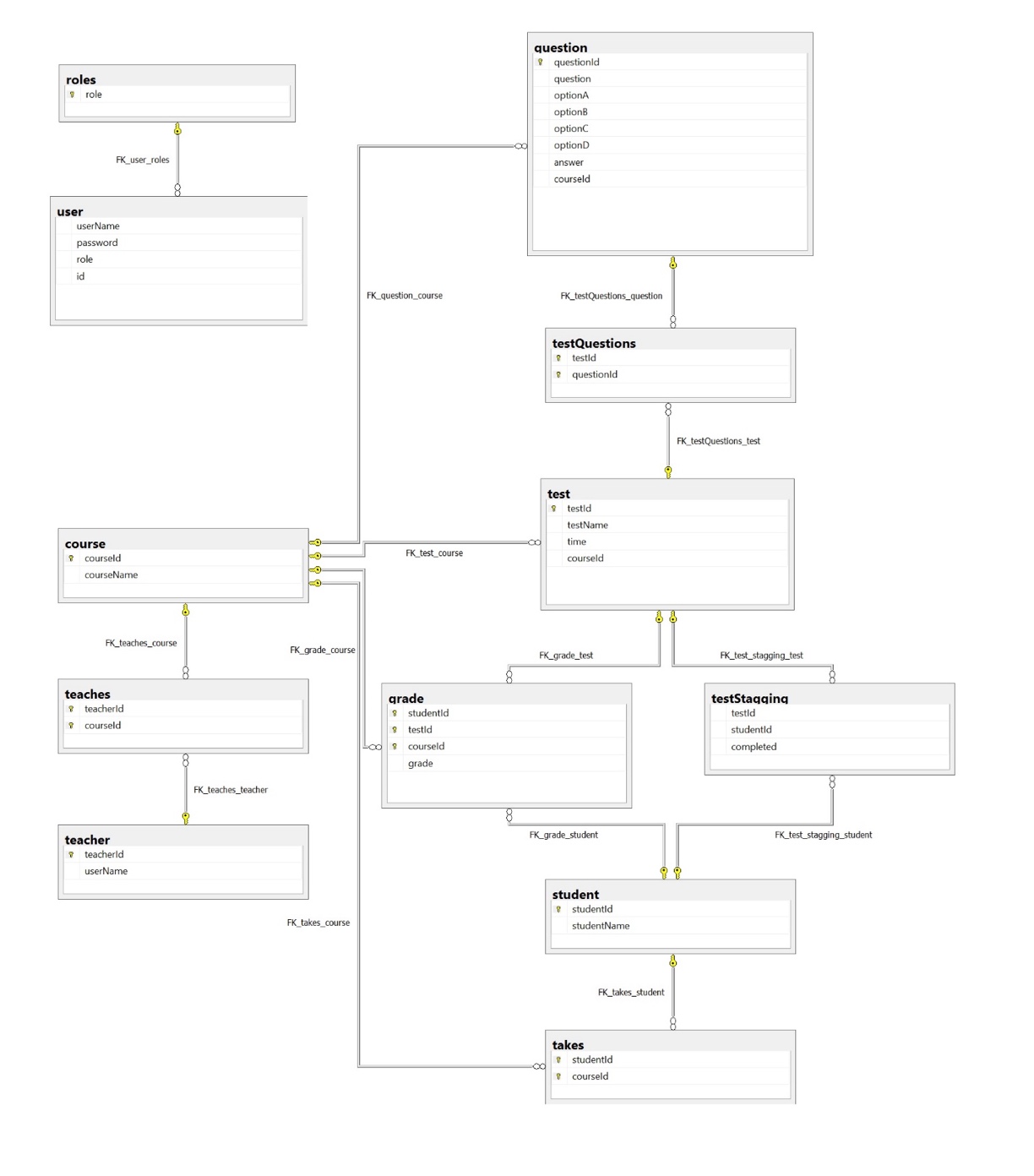


Figure 8 Student grades in a given course

Database Schema



The database schema of the test management application consists of the following tables:

1. course: This table stores information about the course like course id and course name. The course id is a unique identifier assigned to each course and hence is chosen as the primary key of the table.

2. grade: This table stores information about the grade scored by each student in each test for each course. The columns testId, courseId and studentId are foreign keys relating the grade table to test, course and student tables respectively. The combination of all the three foreign keys is used as the primary key of this table. The grade column can have the values ‘A’, ‘B’, ‘C’, ‘D’, ‘E’ or ‘F’.

3. question: This table stores information about the questions and serves as a question bank for all the courses. Each question is related to one course and hence the courseId is added as a foreign key to the table. Each question is assigned a unique identifier called questionId and is used as the primary key of the table. Each question has four answer options and one correct answer.

4. roles: This table stores the different user roles present in the testing system. There are two user roles-student and teacher.

5. student: This table stores information about students like student name and unique identifier called studentId which is used a primary key of the table.

6. takes: This table lists the course ids of the courses taken by each student. This table is assumed to be populated by an external system like student registration system which knows the courses taken by all students.

7. teacher: This table stores information about teachers like teacher name and unique identifier called teacher id which is used a primary key of the table.

8. teaches: This table lists the course ids of the courses taught by each teacher. This table is assumed to be populated by an external system which knows the courses taught by all teachers.

9. test: This table stores information about every test created for all courses. Each test is assigned a unique identifier called testId and hence is used the primary key of the table. Every test has a name and a time duration. Every test is created for a course and hence the courseId is added as a foreign key to represent the course for which the test is created.

10. testQuestions: For all tests, this table lists the questionIds of all the questions that are assigned to the test. Every test has ten questions and hence this table will have ten records, one for each question in the test.

11. testStagging: This table is used to temporarily store information about a test that is being taken a student. It stores the studentId of the student taking the test, the testId of the test being taken and the status of the test. The status indicates if the test is being taken or completed.

12. user: This table stores user information like username and password, which is used for verifying the user credentials while logging the user into the testing system. It also stores the user role. Each user is assigned a unique identifier called id, which is used as the primary key of the table.

It can be noted that all the columns of the tables have atomic values. Hence, the database schema is in first normal form. In each table, all the non-prime columns are functionally determined by the primary key only. There are no transitive dependencies. Hence, the database schema is also in second and third normal form.

Features used in the project

In this project, we have used stored procedures for processing the data in the database and providing the front end java application with the required data. We have used a trigger to grade the test taken by the students. We have also experimented with an index for one of the most frequently used table, the users table.

1. Trigger

The gradeTest trigger is executed after updates on the testStagging table. This table is used to store information about ongoing tests. When a student starts taking a test, a record is created in the testStagging table with the completed column set to 0. When the student submits the test or the test times out, the completed column is updated to 1, indicating that the test has completed and the answers table contains the answers submitted by the student for the given test. When the completed column is updated to 1, the gradeTest trigger is executed. It checks if the completed column is set to 1, then grades the student answers. It finds the number of student answers which are correct and accordingly assigns the grade to the student’s test. It inserts the student grade in the grade table and deletes the records from the testStagging and answers table.

CREATE TRIGGER [dbo].gradeTest ON [dbo].[testStagging] AFTER UPDATE

AS

BEGIN

SET NOCOUNT ON;

If (SELECT completed FROM INSERTED) = 1

BEGIN

DECLARE @sId AS INT

DECLARE @correctAnswers AS INT

DECLARE @tId AS INT

DECLARE @cId AS INT

DECLARE @grade AS NCHAR(10)

SELECT @sId = studentId, @tId = testId FROM INSERTED

SELECT @cId = courseId FROM [TestDB].[dbo].[test] WHERE [testId] = @tId

SELECT @correctAnswers = COUNT(\*)

FROM [TestDB].[dbo].question q, [TestDB].[dbo].answers a

WHERE q.questionId = a.questionId

AND a.studentId = @sId

AND a.answer = q.answer

SELECT @grade = grade

FROM [TestDB].[dbo].[pointsToGrade]

WHERE points = @correctAnswers

INSERT INTO [TestDB].[dbo].[grade]([studentId], [testId], [courseId], [grade])

VALUES(@sId, @tId, @cId, @grade)

DELETE [TestDB].[dbo].[answers]

WHERE [testId] = @tId AND [studentId] = @sId

DELETE [TestDB].[dbo].[testStagging]

WHERE [testId] = @tId AND [studentId] = @sId

END

END

2. Stored procedures

The following stored procedures are used in the test management application:

1. getAllStudentGrades: It returns the student names and grades for the given testId and courseId.

2. getCourseAvailableTests: It returns the test details of the all the tests of given course which are not taken by the student.

3. getCourseQuestions: It returns all the questions for a given course.

4. getCoursesTaken: It returns the course names of all courses in which the student is currently enrolled.

5. getCoursesTaught: It returns the course names of all courses taught by a teacher.

6. getStudentGrades: It returns all the test names and grades obtained by a given student in a given course.

7. getTestIds: It returns the testId of all tests for a given course.

8. getTestQuestions: It returns the questions in a given test.

9. verifyLogin: It verifies if the given username, password and role are valid and returns the studentId or teacherId if the user is valid, -1 otherwise.

3. Index

The user table is most frequently used, as it is needed for verifying the user credentials whenever a user logins to the test management system. To experiment with the index and understand the performance improvement by using the index, we loaded the users table with 42,241 user records. We created a non-clustered index on the user table based on the columns username, password and role. This index is internally maintained as a b-tree by SQL server. Preforming a successful log in of the system took 3 milliseconds without the b-tree indexing and less then 1 millisecond with the indexing.

Conclusion

Through this project it was shown how a well planned database can assist an application who is in need of a database over the internet. This project uses a third normal form database schema that has stored procedures, triggers, and indices to not only speed up the retrieval of information but to help decrease the number of complex SQL statements being sent over the internet. The stored procedures join multiple tables to retrieve an array of information like questions from the question bank or courses that a student is taking from the taking table. The trigger allows the java application to push off the grading of the test to the faster database system and allows the application to move on to other tasks. Indexing is useful for speeding retrieval of information from tables that have a massive amount of entries like the users table. These techniques allowed us to build a smart system that distributed the work according to which system can handle the work the best.