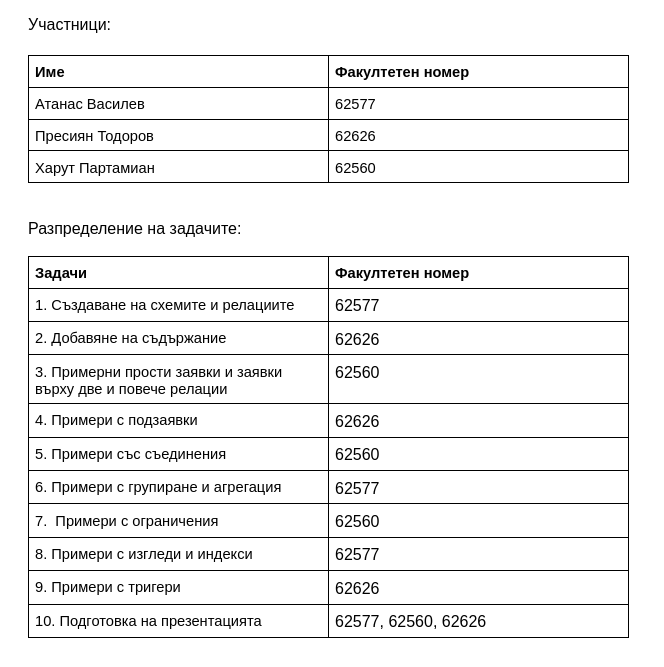
**Система за оценка на задачи**



**1.Описание на заданието**

***Основната идея*** е да реализираме система за оценяване на задачи, която наподобява Hackerrank и Leetcode. Това са системи, в които има много задачи, които се отнасят за даден урок/раздел. Всяка задача има също така и тестове, с които се оценява дали тя е решена правилно от потребителя, който предава решението. В тях също така има и състезания, които се водят за определен период от време и накрая има класиране, спрямо това кой е решил задачите най-бързо. Системата трябва да е проектирана и предназначена да се използва в училища и университети. Системата трябва да може да оценява автоматично програмен код. Тя ще дава възможност на учениците и студентите да тестват решенията си и да подобрят своите умения. Те ще могат да изпратят решението си и да получат отговор до няколко секунди дали решението им е коректно. Потребителите ще имат възможността да решават много задачи и да следят прогреса си, което им дава възможност да станат по-добри програмисти.

***Нашето решение*** - В основата на системата ни стоят курсовете(Courses). Всеки курс има уроци(Lessons), а към уроците има ресурси(Resources) за учене. В нашата система има задачи(Problems), които принадлежат към даден урок, всяка и имат тестове(Tests) и за всяка от задачите потребителите(Users), могат да предават решение(Submission). В системата също така има 3 типа потребители – Администратор, Преподавател и Обикновен потребител. Друга важна част на системата са изпълнените тестове(ExecutedTests). Имаме и състезания(Contests), които отново принадлежат към даден урок. Във всяко състезание имаме определен набор от разрешени IP адреси(AllowedIpAddresses). Също така в него могат да участват потребители.

**2.Дефиниране на схемата на релациите - *01\_Create\_Statements.sql -*** в този файл също се съдържат тригерите и част от ограниченията.

Users(Id, UserName, Email, EmailConfirmed, PasswordHash, PhoneNumber, CreatedOn, ModifiedOn, FirstName, LastName)

Roles(Id, Name)

UserRoles(UserId, RoleId)

Courses(Id, Name, OrderBy)

Lessons(Id, Name, CourseId, OrderBy)

Problems(Id, Name, IsExtraTask, MaxPoints, LessonId, AllowedMemoryInMegaBytes, AllowedTimeInMilliseconds, TimeIntervalBetweenSubmissionInSeconds, OrderBy)

Resources(Id, Name, FilePath, LessonId, CreatedOn, OrderBy)

Submissions(Id, Code, ProblemId, SubmissionDate, UserId, CompilationErrors, ActualPoints)

Tests(Id, ProblemId, Input, ExpectedOutput, IsTrialTest, OrderBy)

ExecutedTests(Id, CreatedOn, IsCorrect, Output, TestId, SubmissionId, Error, ExecutionResultType, MemoryUsed, TimeUsed)

Contests(Id, StartTime, EndTime, LessonId, Name, PasswordHash)

UserContests(UserId, ContestsId)

AllowedIpAddresses(Id, Value, DisplayName)

ContestAllowedIpAddresses(ContestdId, AllowedIpAddressId)

**Отношения много към много:**

Users – Contests (UserContests), един потребител може да участва в много състезания и в едно състезание може да участват много потребители.

Users – Roles (UserRoles), Един потребител може да има различни роли и една роля може да бъде на повече от един потребител.

Contests – AllowedIpAddresses (ContestAllowedIpAddresses), едно състезание може да има много разрешени адреси и на един разрешен адрес могат да се водят много състезания.

**Отношения едно към много:**

Users – Submissions, един потребител може да предава много решения, но всяко решение принадлежи на 1 потребител.

Problems – Submissions

Lessons – Problems

Courses – Lessons

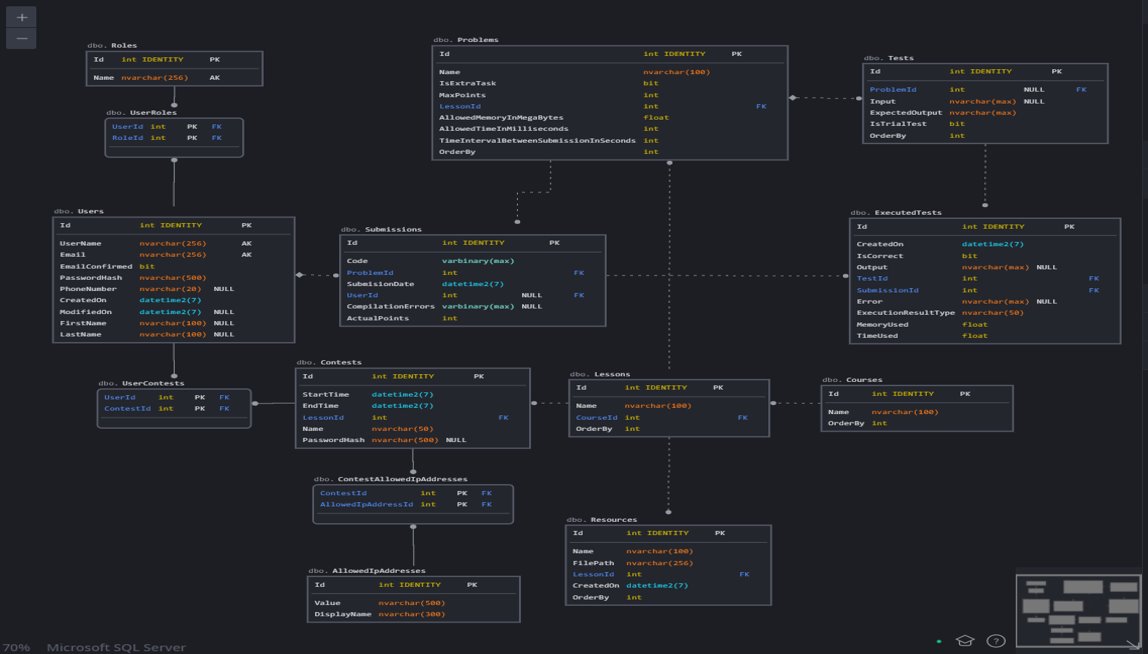
Lessons – Resources

Problems – Tests

Tests – ExecutedTests

Submission – ExecutedTests

Lessons - Contests



**3. Примерно съдържание, за доказване чрез примери на работоспособността на системата - *02\_Insert\_Statements.sql***

**4. Примерни заявки - Прикачваме следния файл, в който са описани всички видове заявки, изискуеми по проекта - *03\_Queries.sql***

**5. По-сложни ограничения - ограничения за уникалност на записите в колоната и ограничение за множество от стойностти - *04\_Constraints.sql***

**6. Индекси - *07\_Indicies.sql***

**7 Тригерите и изгледите са описани съответно във файловете - *05\_Trigger\_Demo.sql, 06\_View\_Demo.sql,***

Trigger за изтриване на Submission

В системата ще имаме много Submissions и след определено време ще искаме да архивираме решенията на потребителите. Но не искаме и да губим решенията на потребителите, за това добавяме Trigger, който се активира, когато се изтрие запис от таблицата Submissions и записва изтрия запис в друга таблица. Обаче един Submission има и изпълнени тестове към него и затова създаваме същия Trigger за таблицата ExecutedTests. Също така задаваме ON DELETE поведението за чуждия ключ SubmisisonId на таблицата ExecutedTests да е CASCADE.

При изтриване на Submission ще се изпълнят следните действия:

1. Ще се изтрият всички изпълнение тестове за решението

2. Изтритите тестове ще се архивират в съответната таблица

3. Изтрива се решението

4. Решението ще се архивира в съответната таблица

Trigger за промяна на тест

Искаме да пазим предишната стойност на теста и момента, в който e update-нат, понеже от бизнес гледна точка промяната на теста е много ключове и трябва да се следят тези промени.

Затова създавамe Trigger, който запазва предишната стойност в нова таблица при промяна на тест.

**8. Насоки за бъдеща работа -** Проектиране на WEB приложение, което да визуализира нашата идея, така че потребителите да могат да използват пълноценно системата. Също така, в момента системата работи само с един език за програмиране. Могат да се добавят различни видове езици за програмиране и да се нагласят максималните времена на задачите, за да се съобразят с разликите в скоростта на изпълнение на различните езици.