УДК 004.652.5

Разработка и реализация модели данных для веб-приложения по сертификации судей спортивной версии игры «Что? Где? Когда?»

Н.С Козлов[1], К.Б. Теймуразов[1,2]

[1]Московский государственный университет им. М.В. Ломоносова
[2]Вычислительный центр им. А.А. Дороницына РАН

Постановка задачи

Необходимо разработать и реализовать модель данных для хранения и выборки тестовых вопросов в ходе сертификации судей спортивного ЧГК (для дальнейшего использования в веб-приложении, упомянутом ранее).

Введем некоторые основные понятия, которые будут использоваться далее в работе:

1) *Сложность вопроса в тесте* – доля неверных ответов, т.е. отношение числа неверных ответов пользователей на этот вопрос к общему числу ответов пользователей на этот вопрос.

2) *Сложность теста* – сумма сложностей вопросов теста.

3) *Логический тип вопроса* – определяет принадлежность вопроса к одному из множеств {простые вопросы}, {составные вопросы}, {подвопросы}. Выделение отдельных логических типов вопросов обусловлено кодексом спортивного ЧГК (например, принятие апелляции на зачет ответа обязывает апелляционное жюри пересмотреть ответы других команд на этот вопрос, что логично организовать в виде составного вопроса и подвопроса к нему).

Такая модель соответствует представлению теста в виде ориентированного графа, вершинами которого являются вопросы, ребрами – ответы пользователя.

Требования, предъявляемые к модели данных:

- объективность к проходящему тестирование, т.е. должна гарантироваться как можно более равная сложность теста для различных пользователей
- поддержание актуальных значений сложностей вопросов, т.е. обновление сложности вопроса после каждого ответа на него
- отображение в модели возможности поставить тест на паузу между вопросами
- отображение в модели возможности дать один или несколько ответов на вопрос в зависимости от его типа
- покрытие тестом всего кодекса спортивного ЧГК, т.е. в ходе тестирования пользователь должен проверяться на знание всех разделов кодекса, за счет использования вопросов всех тематических типов
- равномерное использование тестовых вопросов из базы данных, т.е. должна гарантироваться одинаковая вероятность выбора вопроса из всех доступных
- учет лимита на использование вопроса, т.е. неиспользование вопросов, которые уже встречались в предыдущих тестированиях (в том числе других пользователей) определенное количество раз, а также вопросов, которые уже встречались 1 раз в тестах конкретного пользователя, проходящего тест
- целостность и завершенность теста, т.е. все подвопросы, возникающие в ходе тестирования, должны быть заданы (при каждом ответе, требующем подвопроса, требуемый подвопрос должен быть задан в ходе текущего теста)
- логичность и адекватность, т.е. каждый вопрос в ходе тестирования выбирается в зависимости от предыдущих вопросов и ответов на них

Реализация

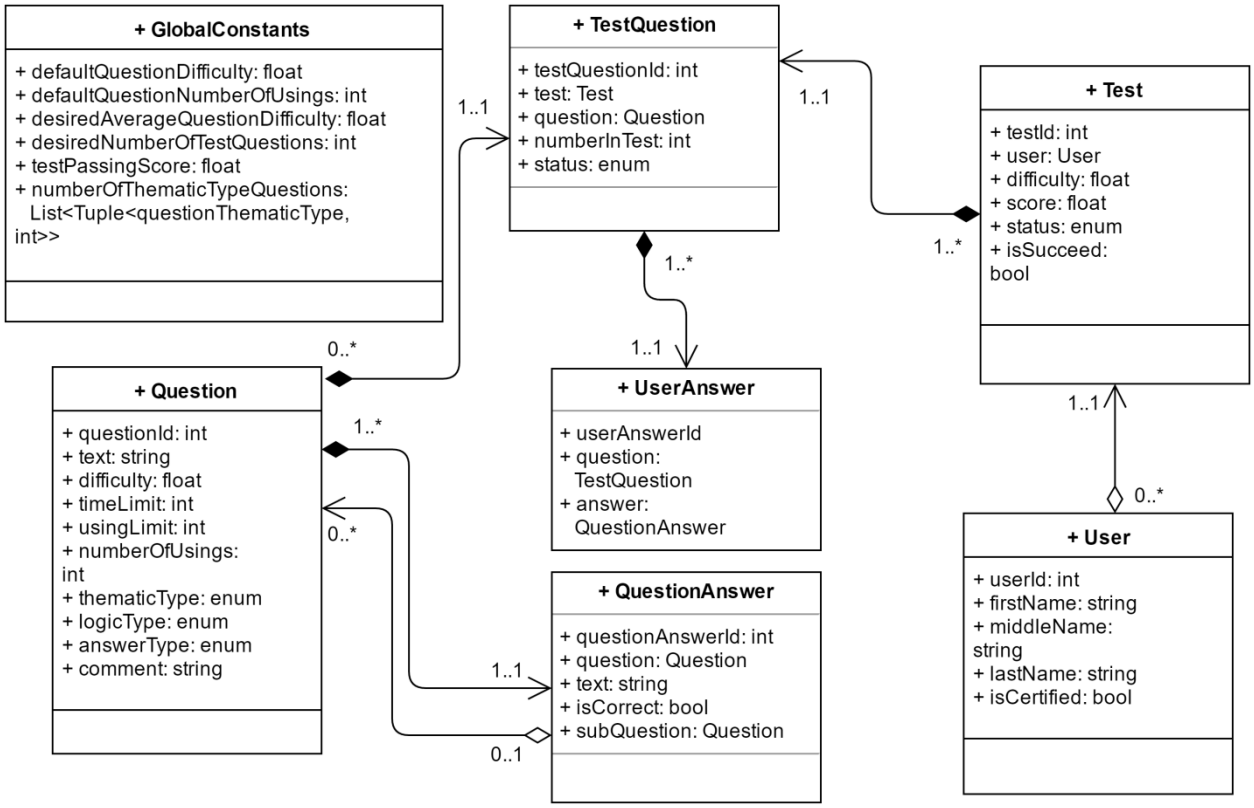

Рис. 1 UML-диаграмма классов предлагаемой модели данных

Смысл классов и их полей:

1) **GlobalConstants** –класс, хранящий основные глобальные параметры приложения:

- defaultQuestionDifficulty – изначальная сложность вопроса
- defaultQuestionNumberOfUsings– изначальное количество использований вопроса

  (эти два параметра обеспечивают адекватную скорость обновления сложности вопроса при поступлении первых реальных ответов на него в ходе тестирований пользователей)
- desiredAverageQuestionDifficulty – желаемая средняя сложность вопроса в тесте (для выполнения требования объективности к проходящему тестирование)
- desiredNumberOfTestQuestions– желаемое число вопросов в тесте (иногда реальное число вопросов может быть незначительно больше из-за требования целостности и завершенности теста)
- testPassingScore – балл, необходимый для удачного прохождения теста
- numberOfThematicTypesQuestions – список пар вида (questionThematicType, число вопросов этой questionThematicType)

2) **User** – основная информация о зарегистрированном пользователе:

- isCertified – логическая переменная, равная *false*, когда все попытки пользователя пройти тест были неуспешными, и *true* иначе

3) **Test** – конкретный тест конкретного пользователя:

- difficulty – текущая сложность теста; при создании экземпляр класса имеет нулевую сложность, которая увеличивается в ходе прохождения теста при добавлении вопросов к нему, конечная сложность должна быть как можно более близкой к desiredNumberOfTestQuestions * desiredAverageQuestionDifficulty
- score – текущее количество очков, набранных пользователем в тесте (подробнее о начислении очков за вопросы в следующем разделе работы); начальное значение равно 0 и увеличивается в ходе прохождения теста при правильно даваемых ответах; условием успешного прохождения теста является score ≥ testPassingScore после прохождения теста
- status – статус теста, перечисляемый тип:

  enum status { paused, active, complete };

  Тест может быть либо active (пользователь получил вопрос и еще не дал ответ на него), либо paused (пользователь ответил на вопрос и, не получив следующий, решил сделать паузу в тесте), либо complete (тест завершен)
- isSucceed – логическое поле, равное *false*, когда score < testPassingScore, и *true* иначе

4) **Question** – вопрос:

- text – текст вопроса
- difficulty – сложность вопроса, изначально полагается равной defaultQuestionDifficulty, и меняется после каждого ответа на вопрос
- timeLimit – время, выделенное на вопрос (в секундах)
- usingLimit – лимит на использование вопроса (количество использований, после которого вопрос больше не используется в тестах)
- numberOfUsings – количество использований вопроса, изначально полагается равным defaultNumberOfUsings
- comment – комментарий к вопросу, содержащий правильный ответ и пояснение со ссылками на кодекс спортивного ЧГК

- thematicType – тематический тип вопроса, перечисляемый тип:

  enum questionThematicType { decision_makings, appeals, sanctions };

- logicType – логический тип вопроса, перечисляемый тип:

  enum questionLogicType { simple, complex, sub_question };

- answerType –тип ответов вопроса, перечисляемый тип:

  enum questionAnswerType { one_correct_answer, few_correct_answers };

5) **QuestionAnswer** – ответ на вопрос:

- text – текст ответа
- isCorrect – логическое поле, равное *false*, если ответ на соответствующий вопрос (question) неверен, и *true* иначе
- subQuestion – ссылка на подвопрос, это поле реализует отношение составного вопроса и подвопроса, для каждого ответа на составной вопрос может быть свой подвопрос (а может и не быть)

6) **TestQuestion** – вопрос в тесте, относится к конкретному тесту и конкретному вопрос, но означает именно составную часть теста:

- numberInTest – порядковый номер вопроса в тесте
- status – статус вопроса, перечисляемый тип:

  enum testQuestionStatus { not_answered, active, answered };

  Вопрос в тесте может быть либо notAnswered (вопрос уже сгенерирован, но пользователь еще не получил его), либо active (пользователь получил вопрос, но еще не ответил на него, и еще не истекло время ожидания ответа на вопрос), либо answered (пользователь получил вопрос и дал ответ на него)

7) **UserAnswer** – ответ пользователя на вопрос, относится к конкретному вопросу теста *TestQuestion* и конкретному ответу *QuestionAnswer*, но означает именно составную часть вопроса в тесте *TestQuestion*;

в зависимости от типа ответа вопроса *Question*, для соответствующего *TestQuestion* может существовать один или несколько ответов пользователя *UserAnswer*.

Здесь есть тонкий момент. В базе данных может возникнуть несогласованность данных, если при добавлении очередного *UserAnswer* не производить проверку того, действительно ли для *Question*, соответствующего данному *UserAnswer.question*, есть *QuestionAnswer*, соответствующий данному *UserAnswer. answer*. Поэтому такая проверка обязательно должна производиться на стадии обработки ответов пользователя на вопрос и занесении их в базу данных, исключительная ситуация должна быть перехвачена и обработана должным образом.

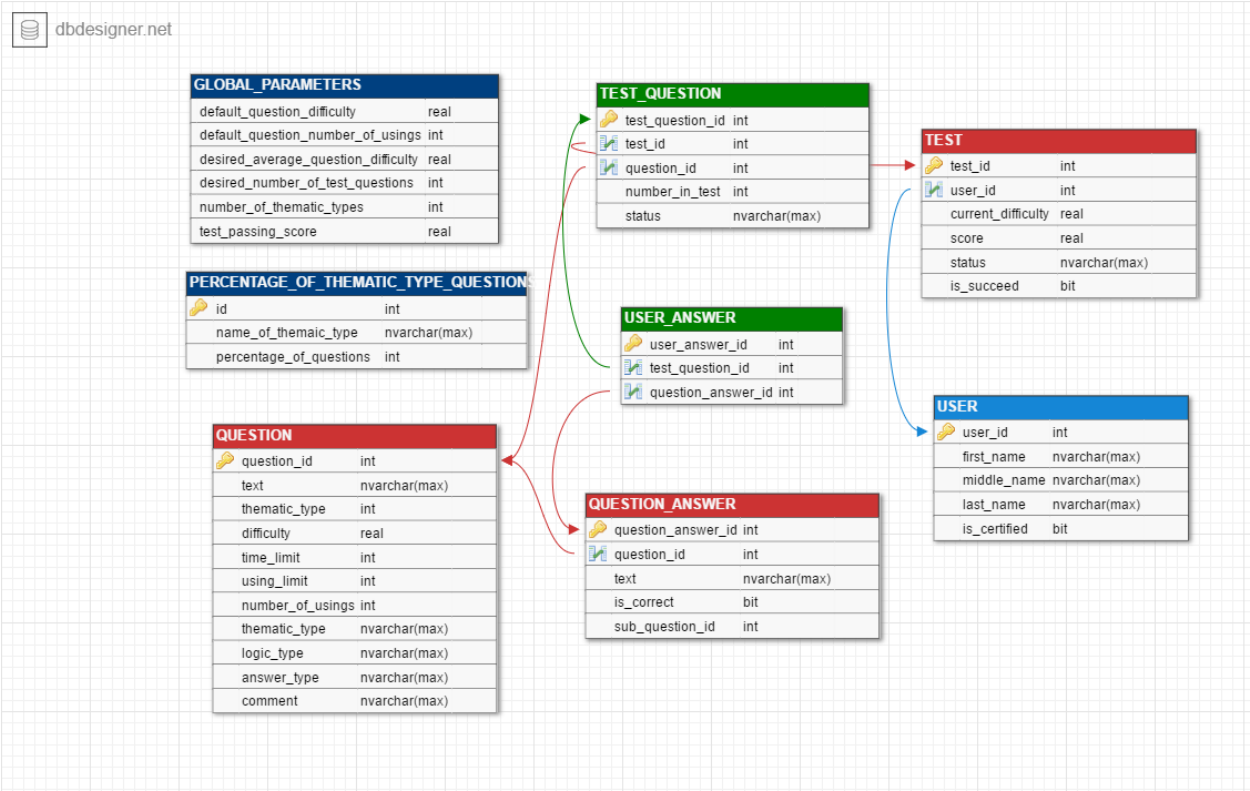

Рис. 2 Схема базы данных, соответствующая представленной диаграмме классов

Все сущности этой реляционной модели абсолютно схожи с соответствующими классами приведенной UML-диаграммы объектно-ориентированной модели, за исключением таблицы-справочника *PERCENTAGE_OF_THEMATIC_TYPE_QUESTIONS*, очевидным образом реализующей поле *numberOfThematicTypesQuestions* класса *GlobalConstants*.

Литература

1. *А.В. Леоненков* Самоучитель UML — СПб: БХВ-Петербург, 2007 – 576 с.

2. *Буч Г., Рамбо Д., Джекобсон А.* Язык UML Руководство пользователя. — М.:ДМК Пресс, 2007, 496 с.

3. *Мухортов В.В., Рылов В.Ю.* Объектно-ориентированное программирование, анализ и дизайн. Методическое пособие. — Новосибирск: Новософт, 2002 – 108 с.

4. *Орлов С.* Технологии разработки программного обеспечения: Учебник. — СПб: Питер, 2002, 464 с.