УДК 004.65 10.5281/zenodo.3904132

Мустафин Р. Ш.*, *Осмонов М.С.
Кыргызско-Российский Славянский университет, г. Бишкек, Кыргызстан
Email: r.mustafin.work@gmail.com; msosmonov@gmail.com

АВТОМАТИЗАЦИЯ ПРОВЕРКИ РЕШЕНИЯ ЗАДАНИЯ ПО SQL

Способность формулировать корректные SQL-запросы является фундаментальным навыком, который требуется многим специалистам по разработке программного обеспечения.
Овладение этим навыком является сложным процессом, требующим от обучаемого значительных усилий и практики. Автоматизация процесса проверки навыков по правильному написанию запросов значительно повышает эффективность такого обучения. Несмотря на ряд предложенных решений, на настоящий момент отсутствуют единые подходы к решению задачи построения систем автоматизированного контроля знаний, а также требования к их программной реализации.
В данной статье предлагается подход к автоматизации проверки результата запросов SQL, базирующийся на использовании DML (Data Manipulation Language) инструкций, с возможностью проверки решения одного и того же задания на разных реализациях СУБД.

Ключевые слова: SQL, электронное обучение, решение заданий, СУБД, база данных, автоматизация.

Введение

Построение запросов к базам данных на языке SQL является ключевым навыком, который требуется многим разработчикам программного обеспечения, так как он лежит в основе практически любого программного обеспечения и используется для манипулирования данными и получения информации.

Несмотря на кажущуюся простоту (обычно для получения нужного результата необходим один или несколько операторов SQL, которые заменяют достаточно большой фрагмент кода на обычном языке программирования) научиться писать SQL-запросы – довольно сложная задача.

Процесс определения, какая информация требуется из базы данных в соответствующий оператор SQL является не таким простым, как кажется на первый взгляд, так как программное обеспечение базы данных выполняет множество операций, которые невидимы программисту.

Это особенно сложно, поскольку не виден результат, который будет при выполнении запроса возвращен из базы данных.

Обычно, для проверки правильности формирования SQL-запросов программисты создают запрос и проверяют возвращаемый результат. Если результат не соответствует желаемому, то запрос уточняется, и шаги проверки и уточнения повторяются, пока не будут получены результаты, удовлетворяющие требованиям. И эта рутинная работа выполняется вручную и визуально, что отнимает очень много времени и сил.

В данной статье рассматриваются вопросы автоматизации процесса обучения языку SQL, что позволяет упростить обучение студентов путем автоматизации проверок задач, выполняемых ими.

В первой части статьи приведено описание возможных формулировок заданий по SQL. Во-второй – рассматривается алгоритм автоматизации процесса проверки задания по SQL.

I. Описание задачи проверки решения задания по SQL

В обычном случае задача по SQL содержит следующие артефакты:

1) база данных (БД);
2) некоторое количество таблиц со связями или без внутри этой БД, которые могут содержать данные;
3) пользовательские процедуры или функции, ограничения, триггеры, индексы и т.д.
4) артефакты, специфичные для конкретной реализации СУБД [3].

Допустим, что задания сформулированы следующим образом:

- создать таблицы в БД с полями X, которые связаны между собой некоторым образом. Заполнить таблицы данными;
- изменить существующую схему данных (таблицы, связи, ограничения), чтобы соответствовать новым требованиям.
- написать запрос (или несколько запросов), который выдаст данные, удовлетворяющие требованиям по заданию**.**

Эти задания можно разделить на два класса: задачи на использование **DDL** (Data Definition Language) инструкций и задачи на использование **DML** (Data Manipulation Language) инструкций. Эти два класса требуют разные типы проверок.

DDL [4] инструкции используются для внесения изменений в **схему базы данных** – в ее структуру. Например, с их помощью можно добавлять новые таблицы, колонки, ограничения, процедуры и функции.

Сложно сформулировать требования к корректности использования **DDL** инструкций. В этом случае результатом (решением задачи) будет являться правильная схема БД, удовлетворяющая некоторым поставленным требованиям. Она может отличаться у преподавателя и студента, однако это не всегда означает, что студент построил схему неверно. Чтобы оценить правильность решения, преподавателю придется практически явно указать студенту в задаче, какие таблицы он должен создать и как их связать, иначе сложно оценить правильность выполнения автоматизированным способом. Однако такой подход снижает полезность заданий на проектирование БД, поскольку студент не проявляет собственного творчества.

DML [5] инструкции позволяют внести изменения в содержащиеся в БД данные – изменить значение в строке, отфильтровать набор данных по условию, сгруппировать и

отсортировать значения. Иными словами, они не изменяют структуру базы данных (схему базы данных).

Проверку решения задания по **DML** осуществить несколько проще, поскольку требования для DML проще сформулировать. Например, они могут быть сформулированы следующим образом.

- «*Необходимо написать запрос, который возвращает таблицу с колонками X, Y, Z… Таблица должна содержать данные, которые …*» Проверка для такой задачи заключается в построчной сверке таблицы, полученной в результате выполнения запроса студента, с таблицей, полученной в результате выполнения запроса преподавателя.
- «*Необходимо изменить данные в таблице, которые соответствуют определенным условиям (Инструкции INSERT, DELETE, UPDATE, MERGE и т.д.)*». Такую задачу можно проверить либо сверкой таблиц студента и преподавателя (в таком случае нужно получить таблицу преподавателя, выполнив его скрипт на копии той же БД), либо преобразовать этот тип задачи в первый тип задачи DML (в этом случае можно сверить результаты запросов SELECT студента и преподавателя, выполненных на измененных данных).

Предметом настоящего исследования является последняя из указанных подзадач, поскольку ее требования более прозрачны и понятны. Основной фокус в работе делается **на инструкциях DML**.

Одной из важных сторон рассматриваемой задачи является необходимость **ограничения возможностей студента** при решении задачи на написание запроса**.**

Требуется установить некоторые границы для запросов, чтобы при решении задачи пользователи или недоброжелатели не написали в запросе что-то, не предусмотренное автором. Например, необходимо, чтобы был закрыт доступ к системным базам, иначе можно повредить всю систему. Также пользователям должен быть закрыт доступ к другим БД на СУБД, не имеющим отношения к его задаче.

Данную проблему можно решить, используя профили безопасности. Автор задания при создании тестовой БД вводит код для создания пользователя базы данных, через которого будет работать студент, и список его прав. Пример выдачи разрешения пользователю «Larry» на запросы, содержащие SELECT и INSERT, для таблицы Employee:

```
GRANT SELECT, INSERT ON HumanResources.Employee TO Larry;
```

Таким образом, профиль ограничивает возможности пользователя при написании и выполнении запросов, тем самым повышая безопасность СУБД.

На данный момент нам известны несколько **аналогичных** систем для обучения, которые предоставляют возможность решать задачи с автоматической проверкой результата.

Одной из таких систем является **sql-ex.ru [6].** Эта система с закрытым исходным кодом позволяет самостоятельно обучаться языку SQL и решать типовые задания на написание запросов. Пользователю дается возможность прочесть задание и посмотреть схему базы данных, затем ввести свое решение в специальное текстовое поле и отправить его на проверку. Задание привязано к типу СУБД.

Другим примером может служить **olymp.krsu.edu.kg [7]**. Эта система тестирует правильность выполнения заданий для императивных языков программирования [8] методом черного ящика [9]; на вход подается поочередно серия тестовых данных и проверяется выход программы.

II. Алгоритм проверки решения

Нами предложен следующий алгоритм проверки решения.

1) Студент выбирает предметную область.
2) Студент выбирает задачу из предметной области.
3) Студент изучает материал, предоставленный автором задачи (ссылки на учебный материал, описание задачи, диаграмму БД и т.д.)
4) Студент пишет скрипт-решение в специальное поле, выбирает тип СУБД, на которой нужно выполнить скрипт, и нажимает «отправить».
5) Система получает решение, тип СУБД, и код задачи от студента.
6) Система отправляет на соответствующие типу СУБД подключенные экземпляры СУБД скрипт студента и скрипт автора задачи. Экземпляры возвращают результаты запросов системе.
7) Система в соответствии с правилами, поставленными преподавателем (например, проверить порядок строк), сверяет результат студента и автора.
    a) Если результаты совпадают (с учетом правил), то решение считается верным.
    b) Если по некоторым причинам, скрипт не удалось выполнить на БД, то студент получает сообщение, которое возвратила СУБД (ошибку).
    c) Если результаты студента и автора не совпадают, то студент получает результат своего запроса (если это разрешено автором) и результат автора (если это разрешено автором). На основании результатов он может изменить свой скрипт-решение.

Правила позволяют проверять разные детали решения, такие как сортировка данных, порядок столбцов, имена столбцов, отсутствие запрещенных ключевых слов.

Более подробно процесс проверки решения на сервере отображен на Рисунок 1.

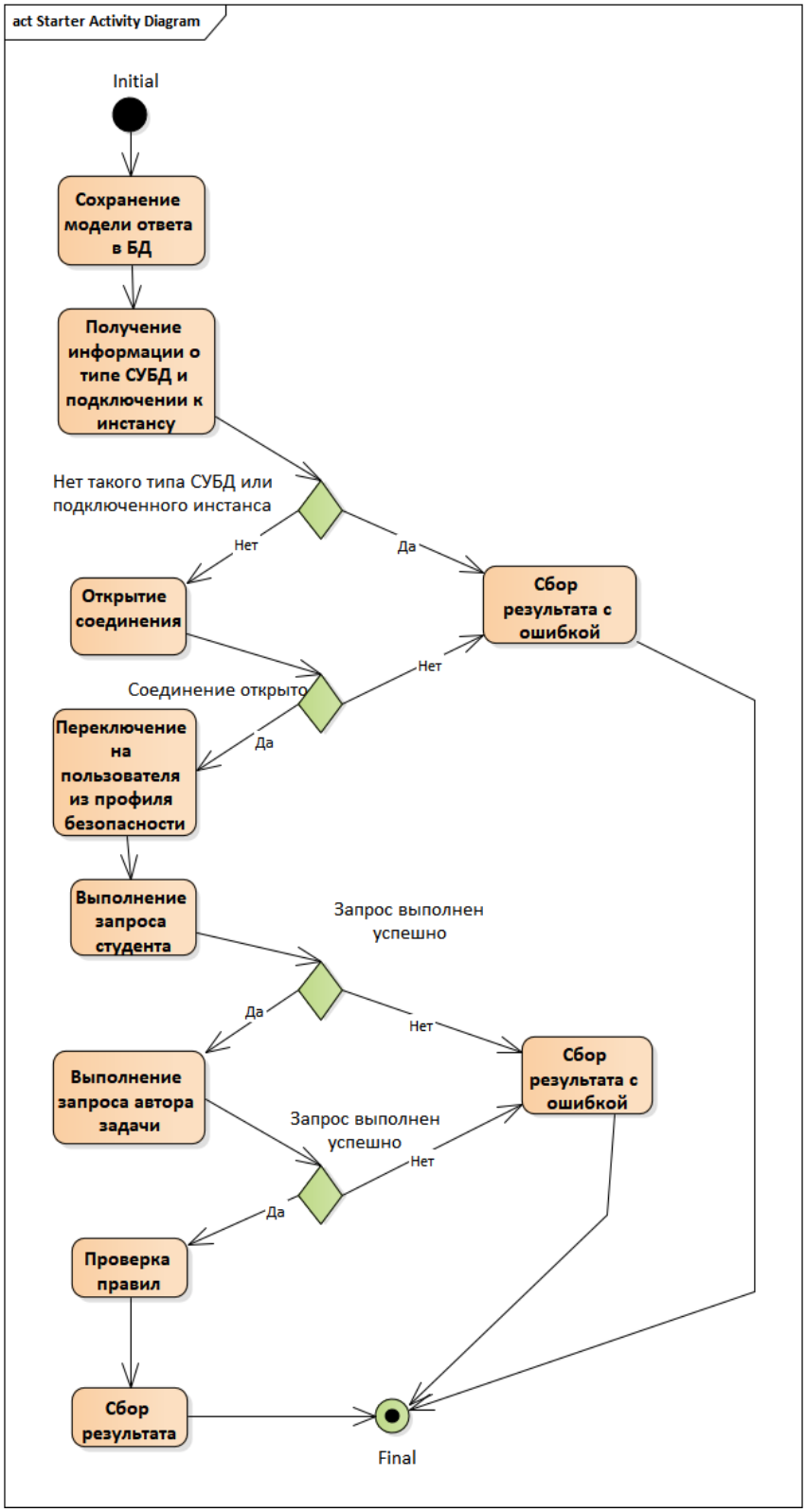

Рисунок 1 – Проверка решения на СУБД

Стоит обратить внимание на то, что первым выполняется скрипт студента, поскольку, очень вероятно, что скрипт будет с ошибками и может не выполниться. В этом случае не имеет смысла выполнять скрипт автора задания.

На основе предложенного алгоритма составлена альфа версия системы на ASP.NET MVC [88], которая имеет следующие возможности[1]:

- Автор задач может создавать абстрактную БД и заполнять ее данными;.
- Автор задач может создавать задачи в системе, внеся их описание, картинки диаграмм БД и свое эталонное решение к ним;
- Студент может изучить выбранную задачу из списка и решить ее, внеся в специальное поле SQL скрипт-решение и выбрав тип СУБД (пока только MS SQL);
- Система проверяет решение студента и выдает результат с подчеркиванием строк в результирующей таблице (ах), в которых допущена ошибка, выявленная при сравнении с ответом автора.

На Рисунок 2 показаны основные классы для выполнения команд на БД и проверки результата.

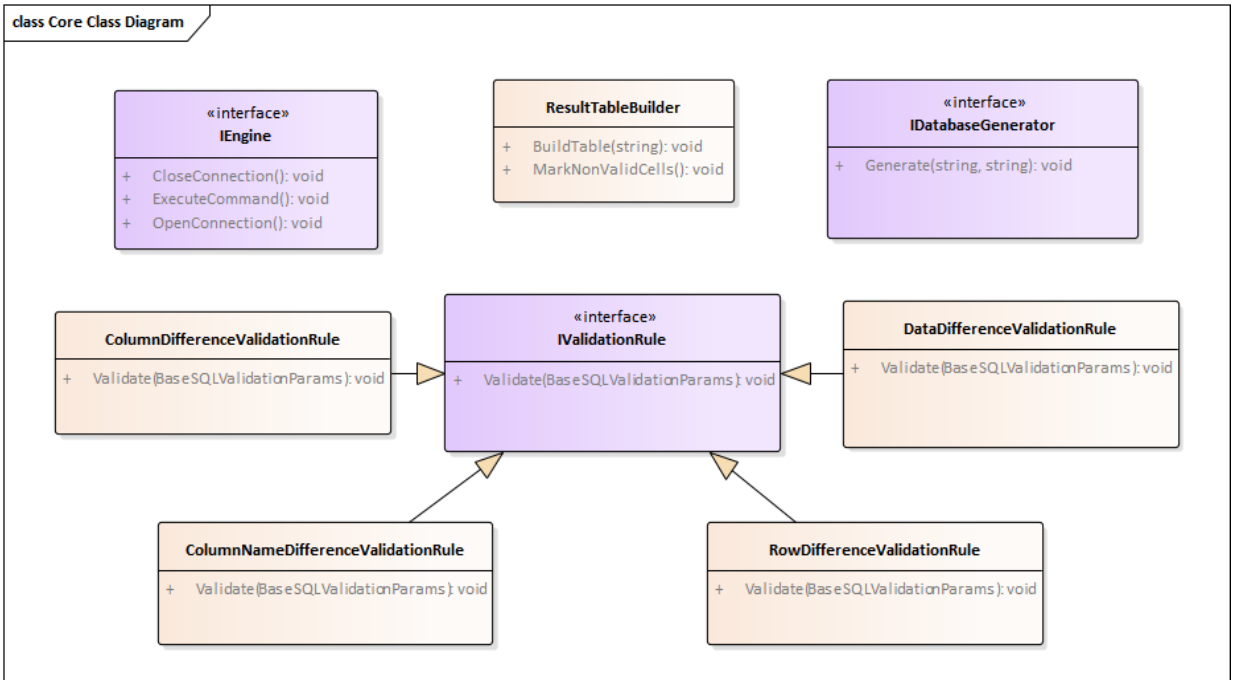

Рисунок 2 – Основные классы и интерфейсы для проверки решения

[1] Данная система была реализована как внутренний проект компании ОсОО «Таймлисофт» (Timelysoft LLC). Сайт компании: http://www.timelysoft.net/

ResultTableBuilder – строит таблицы с результатами (в виде моделей). Неправильные строки маркируются (отмечается поле IsValid = false в объекте строки).

IEngine – интерфейс для выполнения команд на СУБД. Классы, реализующие этот интерфейс, зависят от СУБД, с которой они работают и реализуются в отдельной библиотеке.

IDatabaseGenerator – интерфейс для выполнения команд по созданию и заполнению тестовыми данными БД. Классы, реализующие этот интерфейс, зависят от СУБД, с которой они работают и реализуются в отдельной библиотеке.

IValidationRule – интерфейс для правил проверки решения. Все классы, реализующие данный интерфейс, получают таблицы от запросов студента и автора для своей проверки.

RowDifferenceValidationRule – сверяет количество строк в таблице.

ColumnNameDifferenceValidationRule – сверяет имена колонок в таблицах автора и таблицах студента.

ColumnDifferenceValidationRule – сверяет количество колонок в таблицах автора и таблицах студента.

DataDifferenceValidationRule – сверяет данные в таблицах автора и таблицах студента, также может учитывать порядок строк.

Заключение

Предложен подход по автоматизации проверки результата запросов SQL, базирующийся на использовании DML (Data Manipulation Language) инструкций, с возможностью проверки решения одного и того же задания на разных реализациях СУБД.

Реализована альфа версия системы автоматизации проверки результатов запросов SQL, позволяющая автоматически проверять решение студента без необходимости вмешательства со стороны преподавателя, тем самым сокращая время, затрачиваемое на проверку задания.

Помимо MS SQL, система может также получить поддержку работы с другими СУБД, такими как PostgreSQL, MySQL. Поскольку результатами выполнения запросов в целом являются обыкновенные таблицы с данными, это позволяет студенту выбирать, на каком типе СУБД он хочет выполнить скрипт, не привязываясь к типу СУБД, на котором решил задачу автор. Достаточно будет сверить получившиеся таблицы от двух разных СУБД, чтобы определить – решена ли задача.

Литература

1. Что такое язык SQL (Structured Query Language). URL: https://ru.wikipedia.org/?oldid=103455867 (дата обращения 1 декабря 2019).
2. Определение «электронное обучение». URL: https://ru.wikipedia.org/?oldid=103020624 (дата обращения 29 октября 2019).
3. Определение «Система управления базами данных». URL: https://ru.wikipedia.org/wiki/Система_управления_базами_данных.
4. Data Definition Language (на англ.). URL: https://en.wikipedia.org/w/index.php?title=Data_definition_language&oldid=923093270 (дата обращения 1 декабря 2019).
5. Data Manipulation Language (на англ.). URL: https://en.wikipedia.org/w/index.php?title=Data_manipulation_language&oldid=915843955 (дата обращения 1 декабря 2019).
6. Сайт для самостоятельного изучения SQL. URL: http://sql-ex.ru (дата обращения 1 декабря 2019).
7. Сайт для проведения олимпиадных соревнований и тренировки по программированию университета КРСУ. URL: http://olymp.krsu.edu.kg/ (дата обращения 1 декабря 2019).
8. Императивное программирование и императивный язык программирования. URL: https://ru.wikipedia.org/?oldid=101934922 (дата обращения 1 декабря 2019).
9. Тестирование по стратегии черного ящика. URL: https://ru.wikipedia.org/?oldid=86881359 (дата обращения 1 декабря 2019).
10. Альфа версия системы SolveWay. URL: https://www.solveway.club