Impact Factor:					
	ISRA (India) = 6.317	SIS (USA) = 0.912	ICV (Poland) = 6.630
	ISI (Dubai, UAE) = 1.582	РИНЦ (Russia) = 3.939	PIF (India) = 1.940
	GIF (Australia) = 0.564	ESJI (KZ) = 8.771	IBI (India) = 4.260
	JIF = 1.500	SJIF (Morocco) = 7.184	OAJI (USA) = 0.350

SOI: 1.1/TAS DOI: 10.15863/TAS

International Scientific Journal

Theoretical & Applied Science

p-ISSN: 2308-4944 (print) **e-ISSN:** 2409-0085 (online)

Year: 2022 **Issue:** 05 **Volume:** 109

Published: 24.05.2022 **http://T-Science.org**

Issue Article

Elizaveta Bairovna Tsydypova
Peter the Great St. Petersburg Polytechnic University
Bachelor's Student
Institute of Computer Science and Technology

Oleg Yurievich Sabinin
Peter the Great St. Petersburg Polytechnic University
Candidate of Engineering Sciences, Docent
Institute of Computer Science and Technology

ANALYSIS AND THE DEVELOPMENT OF A PROTOTYPE OF AN AUTOMATED QUERY VALIDATION SYSTEM IN ORACLE SQL LANGUAGE USING ORACLE APPLICATION EXPRESS

Abstract: *The purpose of the work is to develop a prototype of an automated query validation system in Oracle SQL.*

Key words: *automated systems, SQL queries, Oracle Application Express, databases.*

Language: *Russian*

Citation: Tsydypova, E. B., & Sabinin, O. Yu. (2022). Analysis and the development of a prototype of an automated query validation system in oracle sql language using oracle application express. *ISJ Theoretical & Applied Science, 05 (109),* 724-734.

Soi: http://s-o-i.org/1.1/TAS-05-109-67 ***Doi***: https://dx.doi.org/10.15863/TAS.2022.05.109.67

Scopus ASCC: *1700.*

АНАЛИЗ И РАЗРАБОТКА ПРОТОТИПА АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ ПРОВЕРКИ ПРАВИЛЬНОСТИ ЗАПРОСОВ НА ЯЗЫКЕ ORACLE SQL С ИСПОЛЬЗОВАНИЕМ ORACLE APPLICATION EXPRESS

Аннотация: *Цель данной работы – разработка прототипа автоматизированной системы проверки правильности запросов на языке Oracle SQL.*

Ключевые слова: *автоматизированные системы, SQL-запросы, Oracle Application Express, базы данных.*

Введение

UDC 004.6

Современный период развития высших учебных заведений характеризуется значительным увеличением числа студентов, обучающихся по очной и заочной формам. Эта тенденция усугубляется значительным возрастанием нагрузки на преподавательский состав в вузах. В результате процесс обучения в высших учебных заведениях становится все более сложным и менее надежным: решение острых проблем обучения традиционными методами все чаще не дает удовлетворительных результатов, т.е. страдает качество обучения.

Поэтому одной из актуальных задач в области образования является автоматизация проверки работ студентов, которая существенно облегчит образовательный процесс как для студентов, так и для преподавателей. Данная работа фокусируется на задаче проверки правильности запросов студентов на языке Oracle SQL, а точнее – SELECT-запросов, используемых для извлечения и преобразования данных.

Цель и задачи исследования

Целью данной работы является разработка прототипа автоматизированной системы проверки правильности запросов на языке Oracle SQL.

Для достижения заданной цели были сформулированы следующие задачи:

- Проанализировать предметную область, а именно провести краткий обзор языка Oracle SQL.
- Рассмотреть существующие технологии проверки правильности запросов.
- Разработать собственный алгоритм проверки.
- Рассмотреть возможные инструменты разработки системы и обосновать выбор Oracle Application Express как наиболее подходящее средство разработки.
- Организовать хранение данных, основываясь на анализе возможностей системы.
- Описать реализацию автоматизированной системы в Oracle Application Express.

Анализ предметной области

Язык SQL содержит множество операторов, которое можно поделить на группы по предназначению – распространенной классификацией является деление на подъязыки, такие как DDL, используемый для описания структуры БД, или DML, используемый для манипуляции данными в БД. Далее мы рассмотрим только оператор SELECT и соответствующий ему класс SQL-запросов, предназначенный для извлечения данных из БД и их преобразования. SELECT-запросы для краткости назовем просто запросами.

При составлении запроса вводятся следующие параметры отбора:

- названия таблиц, из которых необходимо извлечь данные; [12]
- поля, значения которых требуется вернуть к исходным после внесения изменений в БД; [12]
- связи между таблицами; [12]
- условия выборки; [12]
- вспомогательные критерии отбора (ограничения, способы представления информации, тип сортировки). [12]

Самым распространённым действием при работе с таблицами баз данных является выборка интересующей пользователя информации [12]. Такие процедуры проводятся с целью проанализировать определённого рода информацию. Начинается такая команда служебным словом SELECT [12]. Далее идёт перечень колонок, которые нужно выбрать, служебное слово FROM и имя таблицы, из которой нужно сделать выборку данных. Различают две основные схемы запросов к базе данных: [12]

- простые; [12]
- сложные или вложенные запросы. [12]

Простые, или так называемы прямые запросы, строить проще всего [12]. Именно они максимально понятны, точки зрения логики. Как правило, такого рода запросы делаются с одной или нескольких, объединённых таблиц и имеют очень простую структуру [12]. Для примера можно рассмотреть три таблицы, а именно: Counterfoil – условия оплаты, Person – сотрудники и таблица Depart – отделы. Из этих таблиц, скажем в первом случае необходимо просто выбрать всю имеющуюся в них информацию. Для этого нужно выполнить скрипт такого плана: [12]

```
SELECT * FROM COUNTERFOIL
INNER JOIN PERSON ON PE_ID = COU_PERSON
ORDER BY PE_ID;
```

Данный скрипт вернёт пользователю все данные из таблиц упорядоченный по сотрудникам [12]. Это самая простая конструкция, которую, при необходимости можно усложнить условиями. Использование условий обозначается служебным словом WHERE, после которого можно указывать, те поля таблицы, к которым необходимо применить фильтры. [12]

Простые запросы не всегда эффективны [12]. Если в таблицах много полей и записей, то выборка может продолжаться от нескольких секунд, до нескольких минут, что не совсем целесообразно [12]. Задача любого программиста баз данных – это максимальная оптимизация работы запросов, не только для получения максимально корректной информации, но и ускорение времени их выполнения. Как раз для этих целей и существую так называемые вложенные запросы [12].

Как правило, в таких запросах идёт выборка не по всем полям, а только, по тем, которые необходимы [12]. Можно рассмотреть такой пример. Необходимо вывести номер условия оплаты, сотрудника для которого производится оплата и отдел, где сотрудник работает [12]. Для этого можно воспользоваться вложенным скриптом, который вернёт не все, а только нужные поля таблицы [12]. Такой скрипт будет иметь следующий вид: [12]

- номер условия; [12]
- наименование; [12]
- код сотрудника; [12]
- сотрудник; [12]
- код отдела. [12]

В конце скрипта поставлен фильтр, который выведет информацию только о работающих, на данный момент, сотрудниках компании. [12]

Скрипты на выборку данных с базы могут быть ещё сложнее, но общая структура запросов будет полностью похожа на вышеописанные. [12]

Язык Oracle SQL позволяет писать запросы, возвращающие как уникальные наборы строк, так и неуникальные. Например, для удаления повторений могут использоваться такие операторы, как DISTINCT, UNION, INTERSECT и MINUS – при построении результирующего набора с их использованием все повторения будут удалены. Уникальность также обеспечивается в результате операции агрегации с использованием GROUP BY. Если в таблице есть первичный ключ или ограничение уникальности, то ее экземпляр гарантированно содержит уникальные строки.

Обзор технологий проверки правильности запросов

Проектируемая система должна автоматизировать проверку написанных студентами запросов на языке Oracle SQL. Существует несколько подходов к решению этой задачи.

Первый – проводить сравнительный анализ двух текстов кода – студента и «эталонного» кода, который заверен преподавателем и является правильным.

Второй – производить сравнение результатов работы запросов студента и преподавателя.

Первый подход основан на задаче выявления плагиата в исходном программном коде. Однако вывод, который делается в задаче плагиата, прямо противоположен выводам относительно нашей задачи, т.е. то, что в задаче плагиата считается плагиатом, в рамках нашей задачи будет считаться правильным решением. Важно лишь установить порог, называемый процентом оригинальности, в зависимости от которого будет решаться, правилен ли запрос студента, или нет.

Для разбора и анализа кода важно задать его нормированное представление. Наиболее популярный метод – это представление в виде токенов.

При использовании данного подхода исходная программа трансформируется в последовательность токенов. Каждый оператор языка получает свой код, который заранее был определен для данного класса операторов. Далее полученные коды объединяются в строку с сохранением порядка как в исходной программе. Затем происходит сравнение полученных токенизированных представлений двух программ, чтобы определить степень их сходства.

Сама сущность токенизации заключается в том, что сохраняются существенные и игнорируются легко модифицируемые детали кода программы, [1].

Помимо этого подхода также существуют текстовый подход, [2] и представление в виде ориентированного графа или дерева.

Далее необходимо выбрать алгоритм оценки сходства кода. Существуют атрибутивные и структурные методы. Атрибутивные методы поиска основаны на выделении нескольких отличительных особенностей (атрибутов) кода программ и последующем сравнении полученных чисел. Они делают вывод о совпадении кода, если соответствующие атрибуты равны или близки. Такие методы имеют недостаток, поскольку две различные программы на языке SQL могут иметь сходные характеристики.

Структурные методы позволяют рассматривать программы в целом, с учетом контекста задачи, установить логические связи между элементами. Кроме того, они позволят избавиться от ненужных и лишних элементов кода при использовании токенизированного представления. Также структурные методы позволят специализировать анализ кода для различных языков. Это сделает программу неспособной к анализу кода на любом другом языке, помимо SQL, но удовлетворяет вышеизложенным условиям.

Далее будут рассмотрены методы структурного поиска.

– Алгоритм Смита-Ватермана

Алгоритм Смита-Ватермана был разработан в 1981 году для сравнения последовательностей ДНК, [3]. Это алгоритм динамического программирования, который пытается найти оптимальное выравнивание двух строк. Выравнивание двух строк достигается за счет вставки в них пробелов таким образом, чтобы их длины стали одинаковыми.

В 2004 году Роберт Ирвинг адаптировал данный алгоритм для определения сходства программ, [4]. При многократном применении алгоритма к паре входных программ и последующем удалении получившихся наложений, Ирвинг смог добиться обнаружения всех совпадений, длина которых превышает заданный порог. Он также предложил набор оптимизаций, которые позволяют улучшить эффективность алгоритма при его многократном применении.

Недостатком данного алгоритма является необходимость подбирать пороговое значение.

– Алгоритм Хескела

Алгоритм Хескела обрабатывает токенизированное представление программ. Он основан на вычислении наибольшей общей последовательности двух исходных строк, [1].

Данный алгоритм работает за линейное время.

Вначале исходные строки разбиваются на подстроки. Затем осуществляется поиск полученных подстрок в исходных строках.

Первыми ищутся подстроки, встречающиеся в исходных строках по одному разу. Затем проверяются на совпадение элементы строк, находящиеся над ними. Происходит это ровно до тех пор, пока не будет обнаружено первое несовпадение. В итоге получаем набор общих непересекающихся подстрок исходных строк.

Мерой совпадения кодов исходных программ может являться общая длина полученных непересекающихся подстрок.

Надо заметить, что в данном случае возможно совпадение токенизированного представления программ, даже при отсутствии совпадения в исходных программах. Кроме того, небольшие изменения (например, на эквивалентный оператор) будут приводить к игнорированию части блока, не содержащей уникальную подстроку.

Из достоинств алгоритма можно отметить его линейную сложность.

Второй подход основан на сравнении результатов. Сложностью автоматической проверки соответствия ответа студента правильному SQL-запросу преподавателя является существование разных способов написания верного SQL-запроса к приведенному заданию, то есть перевод семантического вопроса в стандартный SQL-запрос не формализован и носит творческий характер. Например, семантически корректный SQL-запрос может иметь несколько способов написания из-за различного порядка наименований столбцов или таблиц, использованных в запросе.

В ходе анализа возможных методов семантического анализа SQL-запросов была сформирована идея создания механизма, который бы сравнивал результаты выполнения правильного SQL-запроса, подготовленного преподавателем, с результатами выполнения SQL-запроса, написанного обучающимся.

Проверка правильности методом сравнения результатов работы запросов студента и преподавателя заключается в сравнении содержимого двух таблиц. Первая таблица будет содержать результат работы запроса студента, вторая – преподавателя соответственно. Таблицы будут генерироваться при помощи команды CREATE TABLE AS SELECT, т.е. на основе запросов.

В первую очередь проверяются таблицы на совпадение количества строк. Если присутствует несовпадение по количеству, значит решение студента неверно. Явная проверка на равенство количества столбцов таблиц не производится, вместо этого происходит обработка исключения, которое может возникнуть при операции MINUS над запросами с разным числом столбцов.

Далее рассматривается ситуация в случае совпадения количества строк.

При принятии решения о том, верен ли запрос, важно знать, указано ли в задании проведение упорядочивания вывода результата. В языке Oracle SQL есть оператор ORDER BY, используемый для сортировки записей для набора результатов SELECT запроса. Если по заданию необходимо проводить упорядочивание записей, проверка будет происходить строго по строкам с одинаковым номером строки (т.е. первую строку с первой и т.д.).

Если в задаче нет требования по сортировке, сравнение результатов работы запросов производится с помощью операции MINUS, которая возвращает все строки первого запроса SELECT, не возвращаемые вторым запросом SELECT, [5]. Таким образом, для того, чтобы проверить результаты запросов на равенство, производится две таких операции, первая из которых находит разность между результатом студента и результатом преподавателя, а вторая, наоборот, между результатом преподавателя и студента.

Если операция MINUS в обоих случаях возвращает пустое множество, делается вывод о том, что студент правильно решил задачу. В обратном случае считается, что запрос студента неправильный.

После работы с таблицами производится их удаление, поскольку более они не нужны и только занимают место в базе данных.

В результате анализа двух подходов к решению задачи проверки правильности запросов был выбран метод сравнения результатов работы запросов. Данный подход решает одну из главных проблем, возникающих в процессе автоматизации проверки правильности запросов, а именно то, что таким образом проверяются только ответы, тем самым позволяя иметь разные варианты написания SQL-запроса. Более того, методы, основанные на выявлении плагиата текстов запросов, в рамках нашей задачи являются неоправданно сложными и требуют большого количества «ручного» программирования.

Обзор инструментов разработки системы

В данной главе будут рассмотрены веб-приложения как способ внедрения автоматизированной системы.

В настоящее время существует масса различных способов и средств создать веб-приложение, но любой из этих способов можно отнести к одной из трех категорий:

– разработка в конструкторе сайтов;
– разработка на CMS (Система управления содержимым);
– самостоятельная разработка.

Рассмотрим первый способ.

Конструктор – это программно-реализованная система, позволяющая построить сайт по модульному принципу, когда разработчик

собирает структуру сайта с помощью готовых шаблонов, которые предоставляет конструктор. Конструктор позволяет создать сайт без знаний в области веб-разработки и сопутствующих навыков.

Однако такой подход имеет немало минусов:

– тяжеловесность сайта.

Сайт, сделанный на конструкторе, всегда будет загружаться дольше аналогичного сайта, сделанного на CMS или разработанного самостоятельно. Объясняется это тем, что конструктор содержит в себе огромное количество программного кода, который не относится к сайту, но необходим для построения его итогового внешнего вида.

– отсутствие SEO.

Проведение полноценного SEO сайта, сделанного на конструкторе, не представляется возможным, поскольку SEO – это комплекс мероприятий по работе с кодом, индексации в поисковых системах, построению структуры сайта, и для этого требуется доступ к программному коду и разметке, что невозможно в конструкторе.

– скрытые затраты.

За первичной дешевизной часто скрываются дополнительные, порой значительные траты, такие как размещение на хостинге, домен второго уровня и др.

Перейдем к описанию второго способа.

CMS (Content Management System) – это комплекс программных инструментов для создания, управления и изменения веб-приложения и его содержимого. Система управления контентом представляет собой базовый каркас и набор дополнительных инструментов и надстроек, который позволяет не только создать веб-приложение, но и поддерживать его работу, обновлять контент и взаимодействовать с пользователями, [6]. Любая CMS может позволить реализовать сложные решения, такие как интернет-магазины или корпоративные сайты с глубокой вложенностью страниц. У такого подхода большое количество преимуществ, таких как:

– удобное управление контентом.

Все компоненты CMS, включая ее саму и набор расширений, можно всегда поддерживать в обновленном состоянии без особенных трудозатрат.

– поддержка SEO.

В CMS доступны все необходимые инструменты для продвижения ресурса.

– множество готовых решений.

В сети существует масса модулей, плагинов, дополнений для различных задач.

– бесплатный доступ.

Почти все CMS изначально бесплатны, необходимо только оплатить хостинг.

Стоит отметить следующие минусы:

– уязвимость сайта.

Сайты, использующие популярные CMS, чаще всего становятся объектами атак хакеров.

– сложности с переносом.

Хотя популярные CMS на данный момент имеют автоматизированные средства установки почти на любом хостинге, в случае необходимости переноса сайта или управления его положением, могут возникнуть трудности, так как придется производить всю процедуру установки заново.

– замедление работы сайта.

Страницы загружаются не так быстро, как на веб-сайте с ручным кодированием. Для увеличения времени загрузки страниц нужны дополнительные плагины и расширения.

Последний способ – это самостоятельная разработка.

Написание сайта или веб-приложения вручную – творческий и свободный, но и самый трудоемкий процесс, поскольку требует серьезных знаний не только по самим языкам программирования, но и пониманию архитектуры, бизнес-процессов клиента и многому другому.

Для веб-приложения на стороне сервера можно применять различные технологии и такие языки программирования, как PHP, Python, Ruby, C# и другие.

Плюсы такого подхода к разработке веб-приложений состоят в следующих аспектах:

– свобода выбора.

Весь функционал будет написан под конкретные нужды, а не адаптирован из какого-либо шаблона.

– широкие возможности продвижения.

В отличие от CMS и конструкторов, продвинуть в естественном поиске самостоятельно разработанный сайт намного легче.

– высокая скорость загрузки.

Веб-продукт мгновенно загружается в браузере и не нагружает сервер.

– высокая безопасность.

Повышенная безопасность за счет индивидуального подхода и отсутствия кода сомнительного качества.

Говоря о средствах разработки приложений от корпорации Oracle, она предлагает целый ряд средств разработки, которые позволяют создавать надежные, работающие приложения, способные пережить не одну смену компьютерных технологий, аппаратно-программных платформ и групп разработчиков.

Инструментами разработки веб-приложений являются:

– Oracle Forms.
– Oracle Application Express (сокращенно APEX).

Oracle Forms – программное обеспечение для создания экранов управления базой данных Oracle. Это интегрированная среда разработки, включающая навигатор объектов, список свойств и редактор кода, который использует язык PL/SQL. Изначально ПО разрабатывалось для запуска программ на стороне сервера в формате текстового терминала. Сейчас оно может портироваться на разные платформы, включая Windows, Java, и работать в режиме клиент-сервер, [7].

Главное назначение системы – создание систем с доступом к базе данных Oracle.

Oracle Forms обращается к базе данных Oracle и создает экран, на котором представлены данные. Форма в исходнике (*.fmb) компилируется в выполнимую форму (*.fmx) которая может запускаться независимо из модуля запуска форм. Форма используется для просмотра и редактирования данных в приложениях, управляемых базами данных. В форму можно поместить различные элементы GUI, такие как кнопки, меню, полосы прокрутки и графику. Исходный код также может быть помещен в файлы библиотек (*.pll), которые скомпилированы в исполняемые файлы библиотеки (*.plx), используемые во время выполнения.

В результате применения подобного подхода можно создать несколько шаблонов формы по умолчанию, которые обладают полной функциональностью базы данных, но не содержат никакого программного кода.

Второй вариант разработки веб-приложений от Oracle – Oracle Application Express.

Oracle APEX – это low-code платформа для разработки и развертывания современных и безопасных приложений, [8]. Данная платформа содержит удобные средства декларативной разработки и может использоваться для разработки различных приложений для любой предметной области – от простых приложений типа электронной таблицы в браузере до сложных многофункциональных корпоративных приложений.

Платформа APEX доступна как часть СУБД Oracle Database и обеспечивает тесную интеграцию с базой данных, в 10 раз ускоряя обмен данными между приложением и базой данных. Как результат, сокращается время отклика для конечных пользователей приложений.

Oracle APEX использует трехуровневую архитектуру, в которой запросы отправляются из браузера через веб-сервер в базу данных, [9].

Доступ к APEX можно организовать через несколько разных серверов:

– Встроенный PL/SQL шлюз (Embedded PL/SQL Gateway – EPG);
– Oracle HTTP Server с mod_plsql;
– Oracle RESTful Data Services (ORDS).

Последний вариант наиболее предпочтительный, поскольку обеспечивает высокую производительность и безопасность.

Обработка данных и бизнес-логика выполняются в базе данных. Эта архитектура гарантирует доступ к данным с нулевой задержкой, максимальную производительность и масштабируемость из коробки. Веб-запрос из веб-браузера отправляется в ORDS, где он передается в базу данных Oracle для выполнения действий. В базе данных запрос обрабатывается Oracle APEX. Как только обработка завершена, результат отправляется обратно через ORDS в браузер. (рис. 1) APEX позволяет генерировать из данных, хранящихся в БД Oracle, динамические страницы программ и обрабатывать их в режиме реального времени. [13]

При создании или расширении программ, написанных в APEX, метаданные, которые хранятся в таблицах БД Oracle, модифицируются. [13]. Во время работы программы APEX считывает метаданные и отображает программу в браузере. Образно говоря, APEX «живет» в базе данных Oracle и представляет собой набор данных, структурированных в таблицах с большим количеством PL/SQL кода.[13]

В силу описанных характеристик и того факта, что Oracle APEX – это не самая распространенная технология, что повышает интерес в ее изучении, выбор был сделан в пользу именно этой платформы.

Существует 3 способа работы с данной платформой:

– установка на локальный компьютер;
– использование бесплатного сервиса Always Free Service облака Oracle;
– запрос и настройка своего рабочего пространства в бесплатном облачном сервисе apex.oracle.com.

Первый способ имеет несколько важных преимуществ по сравнению с остальными двумя:

– При установке APEX создается администратор со специальным рабочим пространством, который может управлять рабочими пространствами других пользователей, а также выдавать привычные в контексте баз данных привилегии, например привилегию CREATE DATABASE LINK, которая понадобится нам в процессе организации соединения к учебной базе данных.
– Приложения не будут удаляться из рабочих пространств, даже если они будут неактивны в течение долгого времени, в отличие от ситуации, когда используется запрошенное рабочее пространство на облачном сервисе

apex.oracle.com. В этом случае приходится постоянно поддерживать

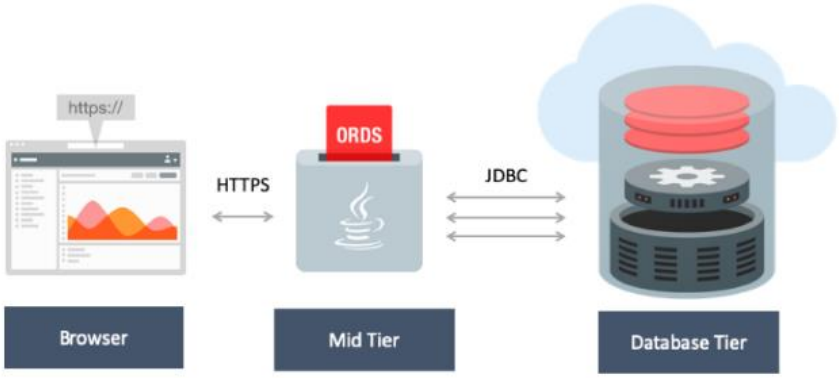

Рисунок 1 - Архитектура Oracle APEX

актуальность приложения во избежание его удаления.

Исходя из вышеперечисленных факторов, выбор был сделан в пользу скачивания Oracle APEX на локальный компьютер.

Организация хранения данных

Прежде чем перейти к организации хранения данных, необходимо описать возможности системы.

Система предназначена для работы с 3 типами ролей:

- администраторы системы, которые имеют доступ ко всем страницам системы, а также обладают наибольшим количеством привилегий.
- преподаватели, которые могут загружать в систему новые задачи, оценивать решение студентов, а также следить за их успеваемостью.
- студенты, которые могут загружать свои решения и следить за своими успехами.

Таким образом, необходимо определить возможности системы для всех ролей.

1. Студент заходит в систему.
   - Отправка задания на проверку
     Студент выбирает из списка курс, нужную лабораторную работу и номер задачи, загружает код задачи в форму и отправляет на проверку.
   - Изменение задания
     При необходимости студент может изменить код и сохранить изменения. При этом, номера курса, лабораторной работы и задания изменить нельзя. Также он может удалить свои решения.
   - Просмотр результатов оценивания
     При попадании на страницу с результатами, студент видит только свои задания с проставленной преподавателем оценкой.
2. Преподаватель заходит в систему.
   - Загрузка заданий
     Преподаватель может загружать новые задачи в систему, а также удалять и изменять загруженные задачи.
   - Проверка работ
     Преподавателю открывается страница со всеми непроверенными заданиями, присланными студентами. При выборе определенной задачи открывается страница с решением студента. Также есть кнопка «Проверить задачу», нажимая на которую преподавателя переносит на страницу, на которой отображается результат проверки данной задачи. Помимо результатов имеется сообщение об ошибке, которая могла возникнуть, если решение оказалось неправильным. В этом же случае отображаются лишние и недостающие строки запроса студента. Далее имеются две кнопки – «ОК» и «Изменить результат», первая из которых подразумевает, что преподаватель согласен с вердиктом системы. При нажатии на вторую кнопку результат инвертируется, что означает, что преподаватель не согласен с системой и желает изменить результат.
   - Просмотр результатов оценивания решений студентов
     Преподаватель видит текущую успеваемость всех студентов, чьи работы были проверены.
3. Администратор заходит в систему.
   - Загрузка заданий
     Администратор также может загружать новые задания, удалять и изменять существующие.
   - Проверка работ
     При необходимости администратору также доступна возможность оценивания работ студентов.

– Просмотр результатов оценивания студентов
Администратор видит текущую успеваемость всех студентов, чьи работы были проверены.
– Работа с пользователями, зарегистрированными в системе
Администратор может просматривать пользователей системы, добавлять новых, а также изменять их данные. В частности, в обязанности администратора входит регистрация преподавателей.

В результате анализа возможностей системы было разработано 5 таблиц, отвечающих за организацию хранения данных:

– USER_ROLE – информация о ролях в системе.
– USER_REPOSITORY – информация о пользователях системы.
– STUDENT_CODE – информация о текстах запросов студентов.
– CORRECT_CODE – информация о текстах запросов преподавателя.
– STUDENT_MARK – информация об успехах студентов.
– PAGE_ACCESS – информация о том, какие страницы приложения доступны определенной роли.

Была составлена ER-диаграмма отношений между таблицами. (рис. 2)

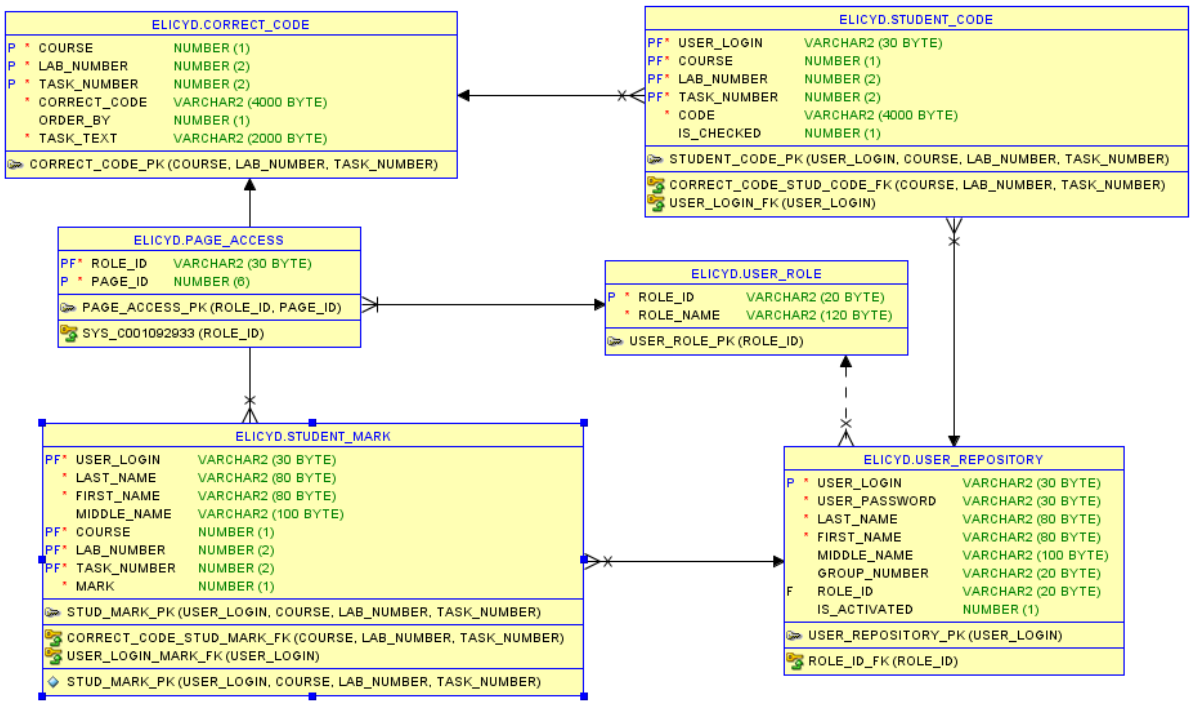

Рисунок 2 - ER-диаграмма связи между таблицами

Данные таблицы были созданы в удаленной базе данных. Чтобы получить к ней доступ из нашего веб-приложения в APEX, необходимо создать с ней связь. В контексте баз данных Oracle для этого существует специальный объект – Database Link.

Связь базы данных Oracle (Database Link) – это одностороннее соединение локальной базы данных с удаленной базой данных. Связь всегда односторонняя. Пользователи удаленной базы не могут применять ее для подключения к локальной базе – вместо этого они должны создать отдельную связь базы данных. Связь базы данных позволяет получать доступ к разным базам данных через учетную запись пользователя удаленной базы. Привилегии в этой базе данных будут идентичны привилегиям пользовательской учетной записи, которая применялась для создания связи. Связи баз данных удобны, когда необходимо запросить таблицу в распределенной базе данных, или вставить данные из таблицы другой базы в собственную локальную таблицу. Связи позволяют пользователям обращаться к множеству баз данных как к единой логической базе данных, [10]. Была использована следующая SQL-команда на создание связи:

CREATE DATABASE LINK "DBLINK2"

```
CONNECT TO "ELICYD" IDENTIFIED BY
VALUES ':1'
USING '(DESCRIPTION =
  (ADDRESS_LIST =
   (ADDRESS = (PROTOCOL =
TCP)(HOST = oraclebi.avalon.ru)(PORT = 1521))
  )
  (CONNECT_DATA =
   (SERVICE_NAME = orcl12)
  )
 )'
```

Реализация автоматизированной системы в Oracle Application Express

В результате разработки была осуществлена модуляризация кода, упростившая процесс отладки программы, оптимизировавшая код, а также сделавшая его более читабельным и надежным, [11]. Таким образом, было создано 5 процедур, участвующих в проверке правильности запросов студентов:

- QUERY_IS_CORRECT1.
- COMPARE_QUERY_RESULTS.
- COMPARE_COL_NAMES.
- RESET_SEQ.
- DROP_STUD_COR_TABLES.

Рассмотрим каждую процедуру в отдельности.

1. QUERY_IS_CORRECT1

Данная процедура принимает на вход текст запроса студента и преподавателя и на их основе создает две таблицы, содержимое которых затем сравнивается в процедуре COMPARE_QUERY_RESULTS.

2. COMPARE_QUERY_RESULTS

Данная процедура производит непосредственное сравнение содержимого двух таблиц, созданных в первой процедуре.

3. COPMARE_COL_NAMES

Данная процедура проверяет имена столбцов, записанных в запросах студента и преподавателя, на соответствие.

4. RESET_SEQ

Данная процедура нумерует строки таблиц, содержащих результаты запросов студента и преподавателя. Используется в случае, когда важен порядок вывода строк в запросе.

5. DROP_STUD_COR_TABLES

Данная процедура удаляет таблицы, содержащие результаты запросов. Вызывается по окончании проверки правильности запроса.

Разработка и оформление страниц В Oracle Application Express начались со стартовой страницы, на которую попадают в первую очередь – это страница входа в систему. Эта страница автоматически формируется при создании приложения в APEX по стандартному шаблону. Имеются поля для ввода логина и пароля. Также, помимо кнопки для входа в систему, есть кнопка регистрации. Она необходима для студентов, которые, в отличие от преподавателей, должны регистрироваться самостоятельно. Переходя по ней, студент попадает в форму регистрации и заполняет поля, аналогичные полям таблицы USER_REPOSITORY, поскольку после успешной регистрации, данные о новом пользователе фиксируются в вышеуказанной таблице.

Следующей была создана страница, отображающая информацию по пользователям системы. Информация берется из таблицы USER_REPOSITORY, хранящейся в учебной базе данных.

Данная страница доступна только администраторам системы. Администратор может добавить нового пользователя, удалить или изменить данные существующих пользователей.

Далее были создана страница, содержащая информацию о задачах. Она связана с таблицей CORRECT_CODE. Эта страница видна администратору и преподавателю. Пользователи с этими ролями могут добавить новую задачу, изменить или удалить существующую.

Аналогично предыдущей была создана страница с информацией по загруженным студентами задачам. Она ассоциирована с таблицей STUDENT_CODE. Для студента доступны только загруженные им задачи. Студент может загрузить новую задачу, изменить код или удалить загруженные задачи. Администратор и преподаватель такими правами не обладают, однако им видны загруженные задачи всех студентов. Помимо этого, преподаватель, нажав на «карандаш» слева от записи, попадает на страницу с информацией по этой задаче. Отсюда он может начать проверку задачи, нажав на кнопку «Проверить задачу».

Одна из главных страниц в системе – та, которая отображает результат проверки. Нажимая на кнопку «Проверить задачу», преподавателя перебрасывает на эту страницу, отображающую результаты проверки, сообщение об ошибке, если таковая есть, а также запросы студента и преподавателя для наглядности. Дополнительно, если студент неправильно написал запрос, появляются две таблицы, хранящие недостающие и лишние по сравнению с результатом преподавателя результаты запроса студента. Если преподаватель согласен с системой, он нажимает на кнопку «ОК», иначе – на кнопку «Поменять оценку».

Последняя созданная страница отображает результаты успеваемости студентов, помещенные в таблицу STUDENT_MARK. Для студентов отображаются только их задачи с проставленной оценкой, в то время как для администратора и преподавателя – задачи всех студентов.

В каждой форме определена валидация для изменяющихся полей. Типовой валидацией является проверка на NOT NULL значение,

UNIQUE значение, а также проверка на числовое значение. Также для каждой такой страницы определены процессы, определяющие логику взаимодействия с базой данных. С их помощью производится вставка новых записей, обновление или удаление записей таблиц.

Следующим этапом было определение схем аутентификации и авторизации пользователей системы.

Для схемы аутентификации была создана функция MY_AUTHEN_SCHEME, принимающая логин и пароль и возвращающая BOOLEAN-значение. В ней происходит проверка наличия учетных данных, полученных на входе, в таблице USER_REPOSITORY. В результате проверки возвращается то или иное логическое значение. Если функция вернула TRUE, пользователь аутентифицирован в системе, иначе – нет.

Для схемы авторизации важна таблица PAGE_ACCESS, содержащая два столбца, определяющих роль и страницу приложения, которая должна быть доступна пользователю с этой ролью. Далее была создана функция PAGE_AUTH, принимающая на вход имя пользователя и идентификатор страницы, и возвращающая BOOLEAN-значение. Функция определяет роль пользователя, имя которого определено во входном параметре, и ищет полученный на входе ID страницы в таблице PAGE_ACCESS среди значений, определенных для этой роли. В зависимости от успеха или неуспеха поиска возвращается то или иное логическое значение. Если функция вернула TRUE, пользователю доступна данная страница системы, иначе – нет.

Немаловажным шагом является необходимость скрыть те страницы, которые пользователю недоступны, из навигационного меню. Для этого было создано динамическое навигационное меню. Сперва было создано представление ROLE_LIST_ENTRY, ассоциирующее каждую роль с идентификатором доступной ей страницы системы. Динамическое навигационное меню представляет из себя SQL-запрос, обращающийся к специальному представлению APEX_APPLICATION_LIST_ENTRIES, представлению ROLE_LIST_ENTRY и таблице USER_REPOSITORY, которые связаны друг с другом по ID страницы системы, ID роли и логину пользователя. Таким образом, для каждой роли определяется свой список страниц, отображающихся в навигационном меню.

Заключение

В данной статье был проведен краткий анализ языка Oracle, рассмотрены существующие технологии проверки правильности запросов, а также приведен собственный алгоритм проверки. Помимо этого, были рассмотрены инструменты разработки систем и выбран Oracle Application Express в качестве итоговой среды разработки. Также было приведено описание реализации автоматизированной системы проверки правильности запросов студентов в среде Oracle APEX.

References:

1. Krass, A. (2006). *Obzor algoritmov obnaruzheniya plagiata v iskhodnyh kodah programm*.
2. Roy, C.K., & Cordy, J.R. (2007). *A Survey on Software Clone Detection Research.* School of Computing Technical Report.
3. Heon, M., & Murvihill, D. (2015). *Program Similarity Detection with Checksims.* A Major Qualifying Project Report submitted to the faculty of Worcester Polytechnic Institute.
4. Irving, R.W. (2004). *Plagiarism and Collusion Detection using the Smith-Waterman Algorithm.* Department of Computing Science, University of Glasgow.
5. (n.d.). *MINUS OPERATOR*. Retrieved 03.05.2022 from https://oracleplsql.ru/minus.html
6. (n.d.). *3 sposoba razrabotki veb-sajta.* Retrieved from 03.05.2022 https://vc.ru/dev/78714-3-sposoba-razrabotki-veb-sayta
7. (n.d.). *Oracle Forms.* Retrieved 04.05.2022 from https://ru.bmstu.wiki/index.php?title=Oracle_Forms&mobileaction=toggle_view_desktop
8. Scott, J. E., & Spendolini, S. (n.d.). *Pro Oracle Application Express.*
9. (n.d.). *Architecture – Oracle APEX*. Retrieved 05.05.2022 from https://apex.oracle.com/en/platform/architecture/#:~:text=Oracle%20APEX%20uses%20a%20simple,scalability%2C%20out%20of%20the%20box
10. (n.d.). *Oracle database link – upravlenie svyazyami baz dannyh*. Retrieved 05.05.2022 from https://oracle-patches.com/oracle/prof/oracle-database-link-%D1%83%D0%BF%D1%80%D0%B0%D0%

B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5-%D1%81%D0%B2%D1%8F%D0%B7%D1%8F%D0%BC%D0%B8-%D0%B1%D0%B0%D0%B7-%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85

11. Andreeva, N. V., Kozhevnikov, V. A., & Sabinin, O. YU. (2019). *Programmirovanie baz dannyh: osnovy PL/SQL:* uchebnik. (p.183). SPb. : Izd-vo Politekhn. un-ta.
12. (n.d.). Retrieved from https://hsbi.hse.ru/articles/vidy-i-tipy-sql-zaprosov/
13. (n.d.). Retrieved from http://enisey.name/umk/upr21/ch13s04.html