МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационные системы и технологии

Специальность 1–40 01 01 Программное обеспечение информационных технологий

Специализация 1–40 01 01 Программное обеспечение информационных технологий (программирование интернет – изданий)

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

«Сервис доставки товаров»

Выполнил студент Трусов Всеволод Сергеевич

(Ф.И.О.)

Руководитель проекта

(учен. степень, звание, должность, Ф.И.О., подпись)

Заведующий кафедрой к.т.н., доц. Смелов В.В .

(учен. степень, звание, должность, Ф.И.О., подпись)

Консультант:

(учен. степень, звание, должность, Ф.И.О., подпись)

Нормоконтролер:

(учен. степень, звание, должность, Ф.И.О., подпись)

Курсовой проект защищен с оценкой

Минск 2022

**Содержание**

[Введение 3](#_Toc121673025)

[Глава 1. Аналитический обзор литературы 4](#_Toc121673026)

[**1.1 Проектирование архитектуры приложения** 4](#_Toc121673027)

[**1.2 Анализ аналогов** 4](#_Toc121673028)

[Глава 2. Архитектура базы данных 6](#_Toc121673029)

[**2.1 Таблицы базы данных** 6](#_Toc121673030)

[**2.2 Представления базы данных** 10](#_Toc121673031)

[**2.3 Индексы базы данных** 11](#_Toc121673032)

[**2.4 Триггеры базы данных** 12](#_Toc121673033)

[**2.5 Процедуры и функции базы данных** 13](#_Toc121673034)

[**2.5.1 Admin\_package** 13](#_Toc121673035)

[**2.5.2 User\_package** 14](#_Toc121673036)

[**2.5.3 Staff\_package** 15](#_Toc121673037)

[**2.5.4 General\_package** 15](#_Toc121673038)

[Глава 3. Описание процедур импорта и экспорта 16](#_Toc121673039)

[Глава 4. Тестирование производительности 16](#_Toc121673040)

[Глава 5. Описание технологии 17](#_Toc121673041)

[Глава 6. Руководство пользователя 18](#_Toc121673042)

[Заключение 19](#_Toc121673043)

[Список литературы 20](#_Toc121673044)

Введение

Анализ пространственных данных на сегодняшний день является одной из распространенных тем, а именно в сфере логистики. Пространственный анализ – это процесс интерпретации данных геоинформационной системы, их изучение и моделирование при помощи компьютерных программ

Данный курсовой проект посвящен разработке реляционной базе данных для сервиса доставки товаров по городу, а также разработки приложения для демонстрации её работы.

База данных — это организованная структура, предназначенная для хранения информации, систематизированная таким образом, чтобы эти материалы могли быть найдены и обработаны с помощью электронной вычислительной машины. Реляционная база данных — база данных, основанная на реляционной модели данных. В качестве СУБД для базы данных была выбрана Oracle 21c, в связи с ее высокой производительностью и надежностью.

Для проверки работоспособности базы данных разработано приложение, написанное на языке Java с использованием такого фреймворка, как Spring.

Для обеспечения безопасности пользователей приложения в данном проекте используется технология хеширования паролей от аккаунта перед записью их в базу данных.

Основные требования к приложению:

* Взаимодействие с базой данных с помощью заранее разработанных процедур.
* Реализация функций заказчика, исполнителя и администратора.
* Поиск ближайших пунктов выдачи товаров относительно заказчика.
* Определение кратчайшего пути доставки товара.
* Определение минимальной стоимости доставки.

В данной документации изложены основные аспекты разработки проекта, а также краткая информация о похожих продуктах, архитектуре, реализации проекта, руководстве пользователя.

Глава 1. Аналитический обзор литературы

## **1**.1 Проектирование архитектуры приложения

Для анализа пространственных данных была использована технология «Spatial And Graph», разработанная компанией Oracle и выпущенная в 2019 году. Информация по применению данной технологии была взята из официальной документации на сайте Oracle.

Приложение для проверки работоспособности базы данных разработано при помощи фреймворка Spring, информация по которому была получена из официальной документации.

## **1**.2 Анализ аналогов

Сервис доставки товаров играет большую роль в жизни людей. По данным 2021 года около 27% людей планеты совершают покупки в сети Интернет. Особенно в период пандемии, когда большинство людей находиться на изоляции, многие используют сервис доставки товаров, еды и т.п. Поэтому многие компании разрабатывают свои сервисы, чтобы привлечь клиентов.

Перед началом разработки, был произведен поиск аналогов данного сервиса, а также выявление сильных и слабых сторон каждого. И основываясь на данном анализе выдвинуты требования к своему сервису доставки товаров. В качестве исследуемых аналогов были выбраны такие сервисы, как «XУТКI», «Boxberry».Основная страница сервиса «ХУТКI» представлена на рисунке 1.1.

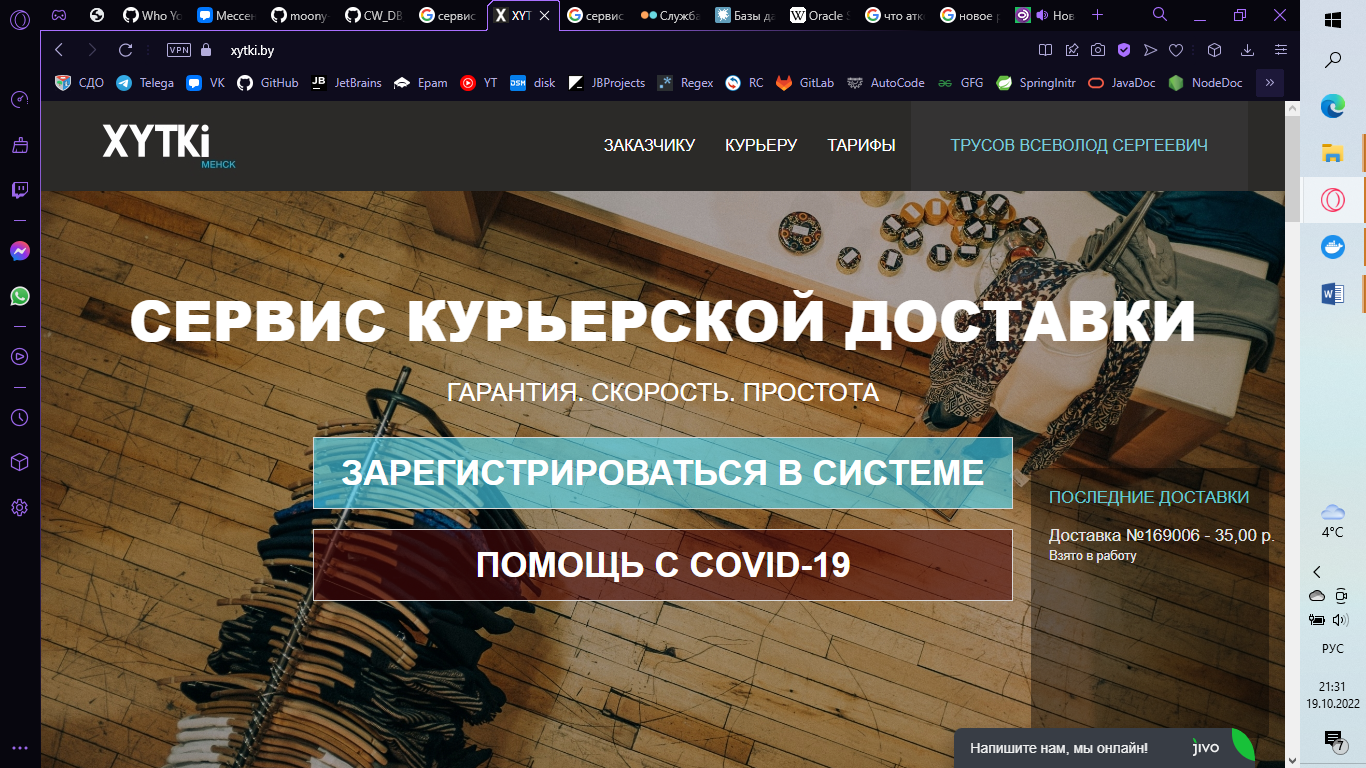


Рисунок 1.1 – Домашняя страница «ХУТКI»

Во вкладке «Новый заказ», представленной на рисунке 1.2, осуществляется оформление заказа. Указывается начальный и конечные пункты доставки, способ оплаты, также описание товара и дополнительные параметры.

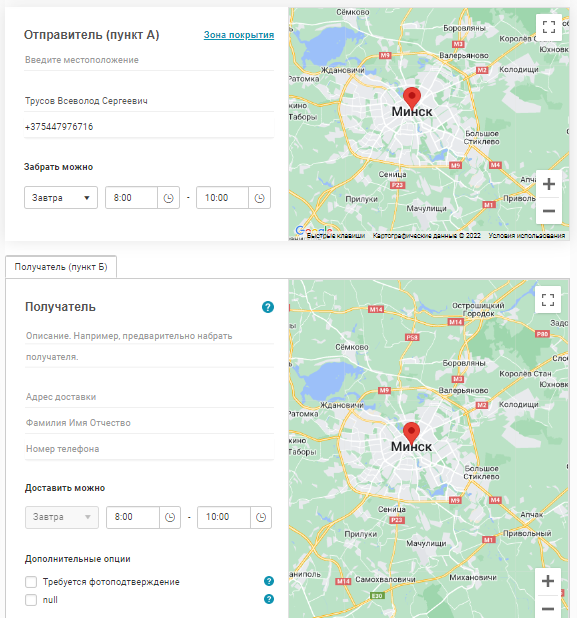


Рисунок 1.2 – Вкладка «Новый заказ»

«Boxberry» – сервис доставки товаров по России и странам СНГ. Страница с расчетом доставки приведена на рисунке 1.3. Данный сервис также позволяет пользователям заказывать перевозку товаров с указанием начального и конечного пунктов. В зависимости от расстояния перевозки, упаковки, ценности товара будет

Рассчитана стоимости доставки.

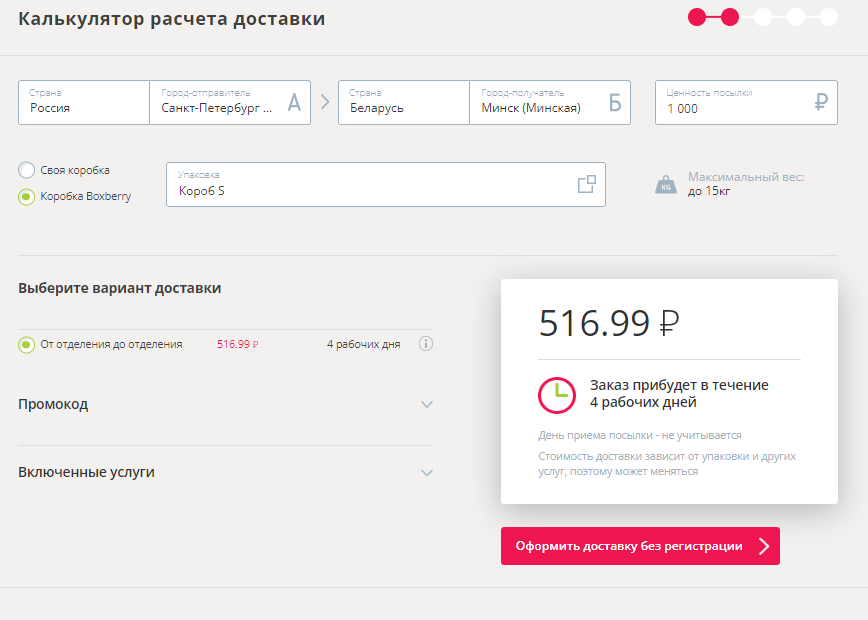


Рисунок 1.3 – Расчет доставки товара

По результатам анализа функциональных возможностей приведенных выше аналогов можно заметить, что данные сервисы доставляют посылки заказчика, у них нет собственных товаров.

Глава 2. Архитектура базы данных

При разработке приложения для курсового проекта была использована база данных Oracle 21c.

## **2.1 Подключаемая база данных**

Перед тем как создать таблицы базы данный и другие объекты нужно определиться куда они будут сохраняться. В данном курсовом проекте была создана подключаемая база данных (Рисунок 2.1):

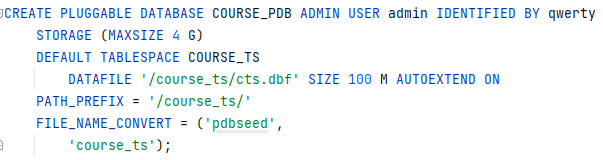


Рисунок 2.1 – Создание подключаемой базы дынных «Course\_pdb»

## **2.2 Пользователи база данных**

Для разработки объектов базы данных было решено создать пользователя-программиста с именем «admin».

Далее под этим пользователем были созданы следующие пользователи базы данных:

* manager;
* customer;
* staff.

Информация о данных пользователях, ролях и профилях безопасности приведена в приложении A.

## **2.3** **Таблицы базы данных**

Для реализации базы данных сервиса доставки товаров было разработано 8 таблиц. Диаграмма базы данных представлена в приложении Б.

Ниже приведено описание каждой из них.

Таблица «UserLogin» представляет собой набор учетных записей. Cостоит из следующих столбцов:

* ID — уникальный идентификатор;
* LOGIN— логин учетной записи;
* PASSWORD — пароль учетной записи;
* ROLE — роль учетной записи.

Скрипт создания таблицы «UserLogin» представлен на рисунке 2.2.

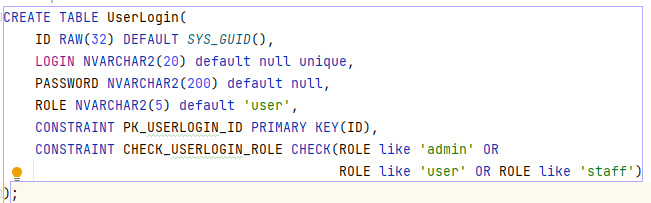


Рисунок 2.2 – Создание таблицы «UserLogin»

Таблица «UserProfile» представляют набор записей, попарно связанных с каждой строкой из таблиц «UserLogin» и «Points». Cостоит из следующих столбцов:

* ID — уникальный идентификатор;
* EMAIL— почта учетной записи;
* USERLOGINID — идентификатор учетной записи в таблице «UserLogin»;
* USERPOINTID — идентификатор точки в таблице «Points».

Скрипт создания таблицы «UserProfile» представлен на рисунке 2.3

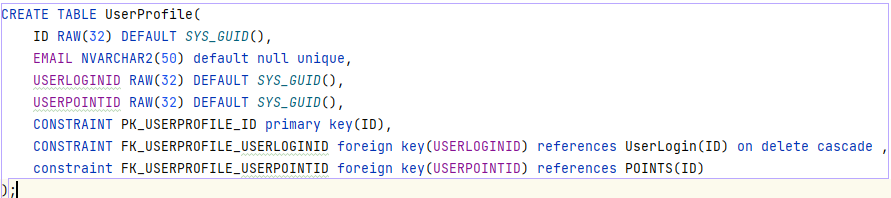


Рисунок 2.3 – Создание таблицы «UserProfile»

Таблица «Goods» содержит информацию о товарах. Cостоит из следующих столбцов:

* ID — уникальный идентификатор;
* NAME — название товара;
* DESCRIPTION — описание товара;
* PRICE — цена товара.

Скрипт создания таблицы «Goods» представлен на рисунке 2.4

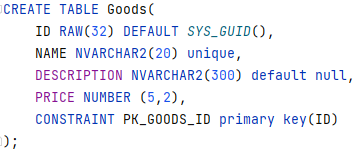


Рисунок 2.4 – Создание таблицы «Goods»

Таблица «Reviews» содержит информацию об отзывах, различной тематики. Cостоит из следующих столбцов:

* ID — уникальный идентификатор;
* CONTENT — содержимое отзыва;
* ESTIMATION — оценка отзыва;
* USERPROFILEID —идентификатор профиля, оставляющего отзыв.

Скрипт создания таблицы «Review» представлен на рисунке 2.5

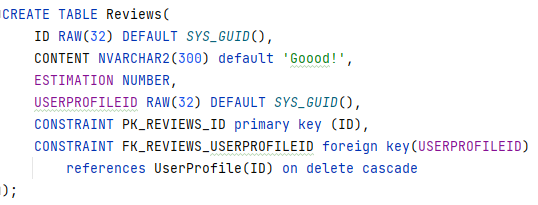


Рисунок 2.5 – Создание таблицы «Reviews»

Таблица «Points» содержит информацию о расположении пользователей и сотрудников сервиса. Cостоит из следующих столбцов:

* ID — уникальный идентификатор;
* LOCATION — геоданные о положении пользователя или сотрудника;
* TYPE — тип точки, может быть пользовательский или для сотрудников;

Скрипт создания таблицы «Points» представлен на рисунке 2.6

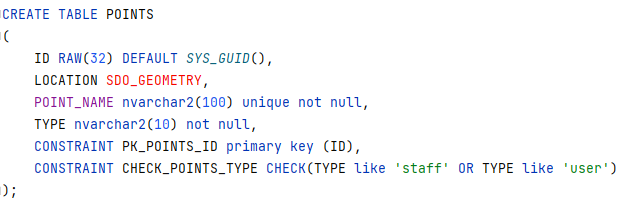


Рисунок 2.6 – Создание таблицы «Points»

Таблица «History» содержит историю заказов пользователей. Cостоит из следующих столбцов:

* ID — уникальный идентификатор;
* ORDERID — идентификатор заказа;

Скрипт создания таблицы «History» представлен на рисунке 2.6

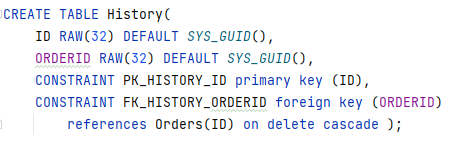


Рисунок 2.6 – Создание таблицы «History»

Таблица «Orders» содержит информацию о заказах. Cостоит из следующих столбцов:

* ID — уникальный идентификатор;
* ORDERNAME — уникальное имя заказа;
* CUSTOMERPROFILEID — идентификатор заказчика;
* EXCECUTORPROFILEID — идентификатор исполнителя;
* STATUS — статус заказа;
* USERLOCATIONID — идентификатор местоположения заказчика;
* DELIVERYLOCATIONID — идентификатор местоположения сотрудника;
* ORDERDATE — дата заказа;
* DELIVERYDATE — дата доставки заказа;
* DELIVERYTYPE — тип доставки заказа;
* Price — цена заказа

Скрипт создания таблицы «Orders» представлен на рисунке 2.7

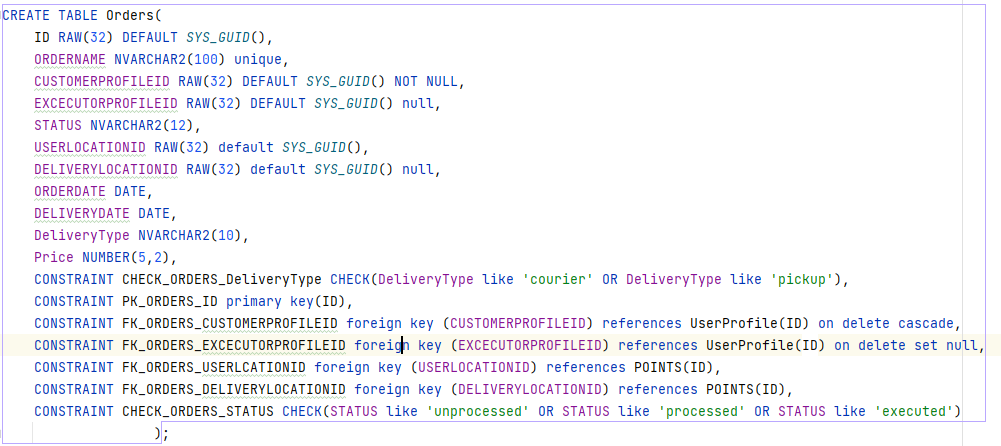
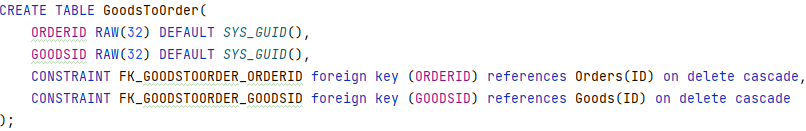


Рисунок 2.7 – Создание таблицы «Orders»

Таблица «GoodsToOrder» содержит информацию о том, к какому заказу относится тот или иной товар. Данная таблица была создана для реализации связи «многие-ко-многим». Cостоит из следующих столбцов:

* ORDERID — идентификатор заказа;
* GOODSID — идентификатор товара;

Скрипт создания таблицы «GoodsToOrder» представлен на рисунке 2.8

Рисунок 2.8 – Создание таблицы «History»

## **2.2 Представления базы данных**

Представление(view), или виртуальная таблица в базе данных — это специфический образ таблицы или набора таблиц, определенный оператором «SELECT». Представление не существует физически как обычная таблица, являющаяся частью табличного пространства. Фактически представление создает виртуальную таблицу или подтаблицу только с теми строками и/или столбцами, которые нужно показать пользователю.

Представление — результат выполнения хранимого запроса, поэтому в словаре данных сохраняется только определение представления.

Представления используются по следующим причинам, включая перечисленные ниже:

* уменьшение сложности;
* повышение безопасности;
* повышение удобства;
* переименование столбцов таблицы;
* настройка данных для пользователей.

В процессе выполнения курсового проекта были созданы следующие представления:

* Unprocessed\_orders\_view для получения необработанных заказов;
* All\_points\_names\_view для получения адреса местоположения;
* Processed\_orders\_view для получения обработанных заказов;
* Orders\_not\_executed\_views для получения невыполненных заказов;
* Reviews\_view для получения всех отзывов;
* History\_view для получения истории всех заказов;
* Orders\_executed\_view для получения заказов и названия всех товаров;
* Orders\_executed\_grouped\_view для получения заказов сгруппированных по его имени.

Листинг создания представлений представлен в приложении B.

## **2.3 Индексы базы данных**

Индекс — объект базы данных, создаваемый с целью повышения производительности поиска данных.

Таблицы в базе данных могут иметь большое количество строк, которые хранятся в произвольном порядке, и их поиск по заданному критерию путём последовательного просмотра таблицы строка за строкой может занимать много времени.

В базе данных были созданы следующие индексы:

* USERPROFILE\_USERLOGINID\_INDEX
* USERPROFILE\_USERPOINTID\_INDEX
* ORDERS\_CUSTOMERPROFILEID\_INDEX
* ORDERS\_EXCECUTORPROFILEID\_INDEX
* ORDERS\_USERLCATIONID\_INDEX
* ORDERS\_DELIVERYLOCATIONID\_INDEX
* ORDERS\_STATUS\_INDEX
* GOODSTOORDER\_ORDERID\_INDEX
* GOODSTOORDER\_GOODSID\_INDEX
* REVIEWS\_USERPROFILEID\_INDEX
* HISTORY\_ORDERID\_INDEX
* HISTORY\_ORDERNAME\_INDEX
* POINTS\_POINT\_NAME\_INDEX
* POINTS\_TYPE\_INDEX
* USERLOGIN\_LOGIN\_INDEX
* USERLOGIN\_ROLE\_INDEX
* GOODS\_NAME\_INDEX

Индексы были созданы для всех полей, к которым применялся оператор «join», также к полям, которые использовались в секциях «where» и «group by».

Для улучшения производительности получения пространственных данных

был создан пространственный индекс «Points\_SP\_Index» к таблице «Points» полю «Location». Листинг создания индексов приведен в приложении C.

Примечание – в данной главе описаны индексы, которые создавались вручную, индексы, которые создавались автоматически из-за таких ограничений, как «primary key» или «unique» и др. здесь не указаны

## **2.4 Триггеры базы данных**

Триггер — [хранимая процедура](https://ru.wikipedia.org/wiki/%D0%A5%D1%80%D0%B0%D0%BD%D0%B8%D0%BC%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D0%B4%D1%83%D1%80%D0%B0) особого типа, которую пользователь не вызывает непосредственно, а исполнение которой обусловлено действием по модификации данных. Триггеры применяются для обеспечения целостности данных и реализации сложной [бизнес-логики](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B7%D0%BD%D0%B5%D1%81-%D0%BB%D0%BE%D0%B3%D0%B8%D0%BA%D0%B0). Триггер запускается сервером автоматически при попытке изменения данных в таблице, с которой он связан. Все производимые им модификации данных рассматриваются как выполняемые в [транзакции](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D0%B7%D0%B0%D0%BA%D1%86%D0%B8%D1%8F_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)), в которой выполнено действие, вызвавшее срабатывание триггера. Соответственно, в случае обнаружения ошибки или нарушения целостности данных может произойти откат этой транзакции.

Для базы данных был создан «add\_in\_history\_table\_after\_adding\_in\_orders\_trigger», добавляющий в таблицу «History» данные после создания заказа или изменяющий запись в таблице при изменении статуса заказа. Ниже приведено создание триггера (Рисунок 2.9):

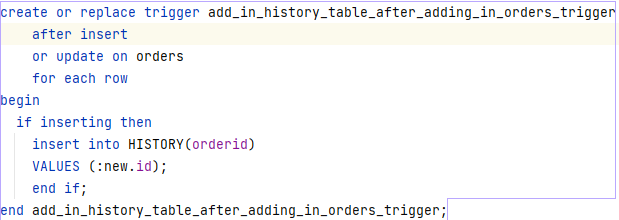


Рисунок 2.9 – Создание триггера

## **2.5 Процедуры и функции базы данных**

Для управления данными через приложение администратор, пользователь или исполнитель использует хранимые процедуры. Хранимая процедура – объект базы данных, представляющий собой набор SQL-инструкций, который компилируется один раз и хранится на сервере.

В данном курсовом проекте написанные процедуры и функции разбиты на следующие пакеты:

* Admin\_package;
* User\_package;
* Staff\_package;
* General\_package.

Таблицы, содержащие сигнатуры процедур и функций, приведены в приложении Д.

## **2.5.1 Admin\_package**

Процедра «delete\_reviews» предназначена для очищения таблицы «Reviews».

Процедура «update\_order\_executor\_deliverypoint» предназначена для изменения строки из таблицы «Orders» а именно полей, содержащих идентификатор исполнителя, идентификатор пункта выдачи товара, статус и цену товара.

Функция «get\_unprocessed\_orders» возвращает курсор, содержащий записи из представления «unprocessed\_orders\_view», которое содержит записи о заказах, статус которых «unprocessed».

Процедура «delete\_good\_by\_name» удаляет запись из таблицы «Goods». Параметром принимает имя товара.

Процедура «add\_good» добавляет строку в таблицу «Goods». Параметрами принимает название товара(строка), описание товара(строка) и цену(число).

Функция «get\_executors\_by\_point\_name» возвращает курсор, содержащий логин исполнителя и количество текущих заказов у данного исполнителя. Параметром принимает строку, содержащую адрес из таблицы «Points».

Функция «get\_all\_points\_name» возвращает курсор, содержащий записи из представления «all\_points\_names\_view».

Функция «get\_all\_persons\_by\_role» возвращает курсор, содержащий записи из таблицы «UserLogin». Параметром принимает строку, содержащую роль.

Функция «get\_customerlogin\_by\_order\_id» возвращает строку, содержащую логин из таблицы «UserLogin». Параметром принимает идентификатор из таблицы «Orders».

Функция «get\_userlocation\_by\_order\_id» возвращает курсор, содержащий адрес из таблицы «Points». Параметром принимает идентификатор из таблицы «Orders».

Функция «get\_count\_orders\_by\_executor\_login\_id» возвращает число заказов, не имеющих статус «executed». Параметрами принимает идентификатор из таблицы «UserLogin».

Функция «get\_count\_unprocessed\_orders» возвращает число строк из таблицы «Orders» со статусом «unprocessed».

Функция «get\_count\_staff» возвращает число строк из таблицы «UserLogin», значение в поле «role» которых равно «staff». Тем самым данная функции подсчитывает количество доставщиков товаров.

Функция «get\_user\_count» подсчитывает количество пользователей. Возвращает число строк.

Функция «get\_admin\_info» возвращает курсор содержащий один элемент со значениями полей, являющимися результатами функция «get\_count\_staff», «get\_user\_count», «get\_count\_unprocessed\_orders».

## **2.5.2 User\_package**

Функция «encrypt\_password» предназначена для шифрования паролей. Она принимает параметром строку, являющуюся незашифрованным паролем, который ввел пользователь при регистрации и возвращает хэш переданной строки.

Функция «dencrypt\_password» предназначена для расшифрования хэша. Принимает параметром строку, содержащую хэш и возвращает расшифрованную строку.

Процедура «register\_user» используется для регистрации пользователя в базе данных. Данная процедура добавляет запись в таблицу «UserLogin» и «UserProfile». Имеет следующие параметры логин, название улицы, к которой будет относиться данная учетная запись, пароль, роль и почта.

Функция «authorisation» используется для авторизация учетной записи. Принимает параметрами логин и пароль. Функция возвращает курсор содержащий логин и роль из таблицы «UserLogin».

Процедура «add\_review» добавляет запись в таблицу «Reviews». Параметрами принимает текст отзыва, оценку и логин ученой записи, оставившей отзыв.

Процедура «delete\_order\_by\_name» предназначена для удаления заказа из таблицы «Orders». Параметром принимает имя заказа.

Функция «get\_good\_by\_name» возвращает курсор содержащий строку из таблицы «Goods». Параметром принимает имя товара.

Функция «get\_history\_by\_login» возвращает курсор, содержащий строки из представления «history\_view». Параметром принимает логин учетной записи, историю заказов которой нужно получить.

Функция «get\_orders\_not\_executed\_by\_login» возвращает курсор, содержащий записи из представления «orders\_not\_executed\_view». Параметром принимает логин учетной записи, для которой нужно получить информацию о заказах.

Функция «add\_order» возвращает имя заказа. Параметрами принимает логин учетной записи, создавшей заказ, дату заказа, дату доставки заказа, тип заказа, начальную цену заказа, адрес пункта выдачи заказа.

Функция «get\_routes\_by\_user\_login» курсор, содержащий записи, содержащие поля адрес пункта доставки, расстояние между адресом заказчика и пунктом выдачи товара, а также количество сотрудников в данном пункте доставки. Данная функция предназначена для анализа пользователем, в какой пункт выдачи товара он пойдет, чтобы забрать товар.

Функция «insert\_into\_orders» возвращает идентификатор заказа. Предназначена для вставки в таблицу «Orders». Параметрами принимает идентификатор заказчика, идентификатор исполнителя, начальный пункт выдачи товара, статус заказа, адрес заказчика, дата заказа, дата доставки, тип заказа, цена заказа.

Процедура «add\_goods\_to\_order» добаляет запись в таблицу «GoodsToOrder». Параметрами принимает идентификатор заказа и идентификатор товара.

Процедура «add\_good\_to\_order» параметрами принимает имя заказа и название товара, который нужно добавить в заказ. В реализации данной процедуры получается идентификатор заказа и идентификатор товара. Затем вызывается процедура «add\_goods\_to\_order» в которую передаются найденные идентификаторы.

Функция «get\_orders\_count\_by\_status\_or\_all» возвращает число строк из таблицы «Orders». Параметрами принимает логин учетной записи, статус заказа, символ, сигнализирующий о том, чтобы получить все заказы к данному логину или только с конкретным статусом.

Функция «get\_customer\_info» возвращает курсор, содержащий поля количество необработанных заказов, количество обработанных заказов и количество всех заказов. Параметром принимает логин пользователя.

## **2.5.3 Staff\_package**

Функция «get\_processed\_orders\_count\_by\_login» возвращает число строк из таблицы «Orders» Параметром принимает логин учетной записи.

Функция «get\_processed\_order\_to\_staff\_by\_login» возвращает курсор, содержащий записи из представления «processed\_orders\_view». Параметром принимает логин учетной записи.

## **2.5.4 General\_package**

Функция «count\_rows\_of\_goods» возвращает число строк из таблицы «Goods».

Функция «get\_pagination\_goods» возвращает курсор, содержащий записи из таблицы «Goods». Параметрами принимает стартовый номер строки и конечный номер строки, таким образом, предоставляется возможность получать строки из таблицы частями.

Функция «get\_reviews» возвращает курсор, содержащий строки из представления «reviews\_view»

Процедура «change\_order\_status\_by\_name» изменяет запись в таблице «Orders», а именно поле «status». Параметром принимает имя заказа и новый статус заказа.

Функция «get\_distance\_between\_deliverypoint\_customer» возвращает число, равное расстоянию между двумя точками из таблицы «Points». Параметрами принимает адрес точки пункта выдачи товара и адрес точки заказчика.

Функция «get\_route\_length\_analysis» возвращает курсор, содержащий записи с полями адрес пункта выдачи, значение функции «get\_distance\_between\_deliverypoint\_customer» и количество сотрудников в данном пункте выдачи товаров.

Глава 3. Описание процедур импорта и экспорта

В рамках задания нужно было разработать процедуру, экспортирующую данные из представления «orders\_executed\_grouped\_view» в XML документ. С этой целью разработана процедура orders\_export. Реализация которой приведена на рисунке 3.1



Рисунок 3.1 – Процедура «orders\_export»

Также нужно было написать процедуру, которая бы, осуществляла импорт данных из XML документа в таблицу базы данных. В этом случае данные вставляются в таблицу «Goods». Процедура goods\_import приведена на рисунке 3.2.



Рисунок 2.9 – Процедура «goods\_import»t

Глава 4. Тестирование производительности

Для тестирования производительности была выполнена череда select-запросов, один из которых это запрос на получения списка пунктов выдачи товара и расстояния от каждого пункта до точки с адресом заказчика. Был просмотрен план данного запроса, его стоимость составляла 3.0.

Для улучшения производительности был создан индекс на таблицу «Points», а именно индекс к полю «type». После этого стоимость select-запроса стала равна 2.0.

Далее для оценки производительности API были проведены тесты в такой программе, как «Postman». Был вызван метод, который в свою очередь вызывает раннее исследуемый select-запрос. На момент существования таблицы «Points» без индексов, время, затрачиваемое на выполнение данного запроса к API, составляли 1.1 секунд. После добавления индекса время, затрачиваемое на выполнение запроса, стало составлять 661 миллисекунд (Рисунок 4.1).

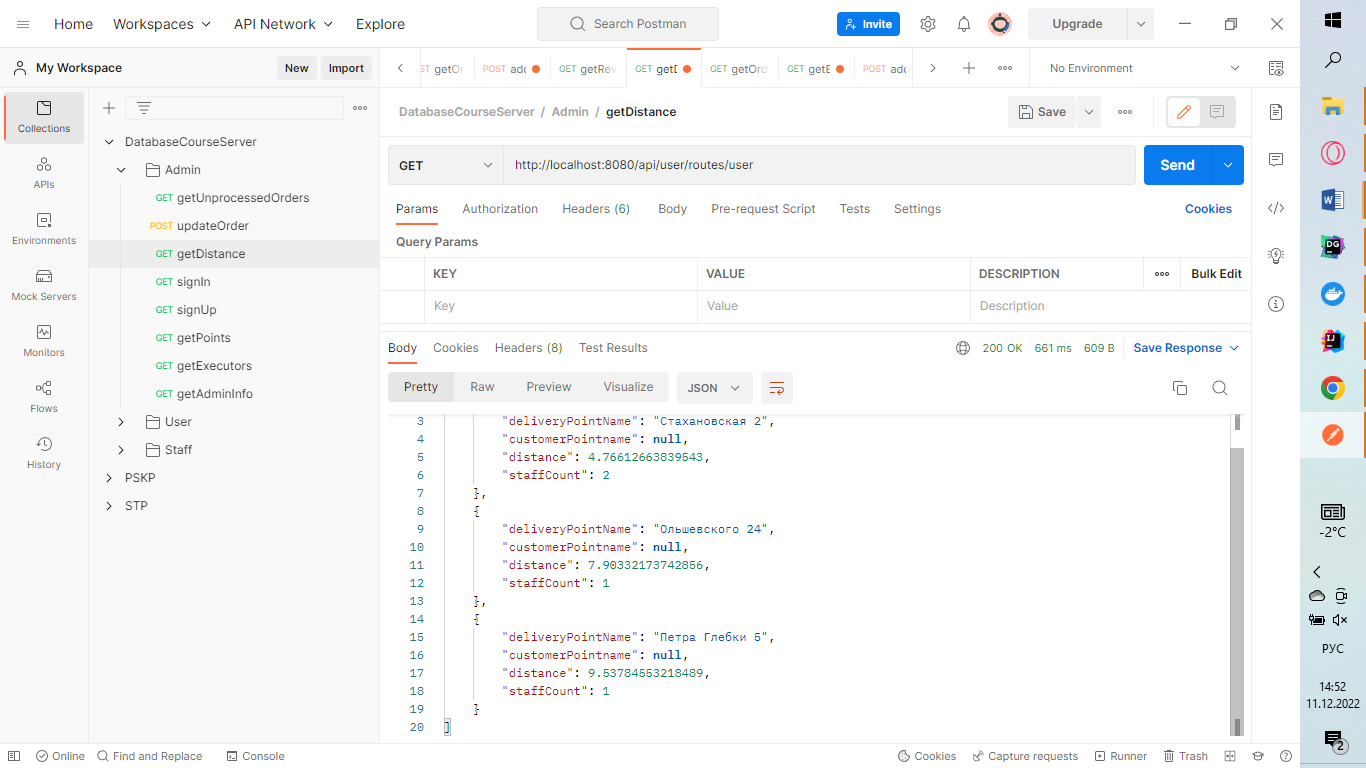


Рисунок 4.1 – Анализ времени выполнения запроса к API

Глава 5. Описание технологии

В данном курсовом проекте была применена технология под название «Spatial and Graph» для реализации анализа расстояния между точками в пространстве. Данная технология предоставляет удобный функционал для пространственных данных, а также их анализа. Пространственные объекты предоставляют схему и функции, облегчающие хранение, поиск, обновление и запрос коллекций пространственных объектов в базе данных Oracle. Spatial and Graph состоит из следующего:

* Схема (MDSYS), предписывающая хранение, синтаксис и семантику поддерживаемых геометрических типов данных.
* Механизм пространственной индексации
* Операторы, функции и процедуры для выполнения запросов области интереса, запросов пространственного соединения и других операций пространственного анализа
* Функции и процедуры для утилит и операций настройки
* Модель данных топологии для работы с данными об узлах, ребрах и гранях в топологии;
* Сетевая модель данных для представления возможностей или объектов, которые моделируются как узлы и связи (вершины и ребра) в графе;
* GeoRaster — функция, позволяющая хранить, индексировать, запрашивать, анализировать и доставлять данные GeoRaster, т. е. растровые изображения и данные с координатной сеткой, а также связанные с ними метаданные.

Для нахождения расстояния между точками использовалась такая функция, как «sdo\_geom.sdo\_distance», которая принимает первым параметром точку, вторым параметром также точку, третьим – точность, до которой нужно получить результат, четвертым – единицы измерения. Ниже приведен пример использования данной функции (Рисунок 5.1):

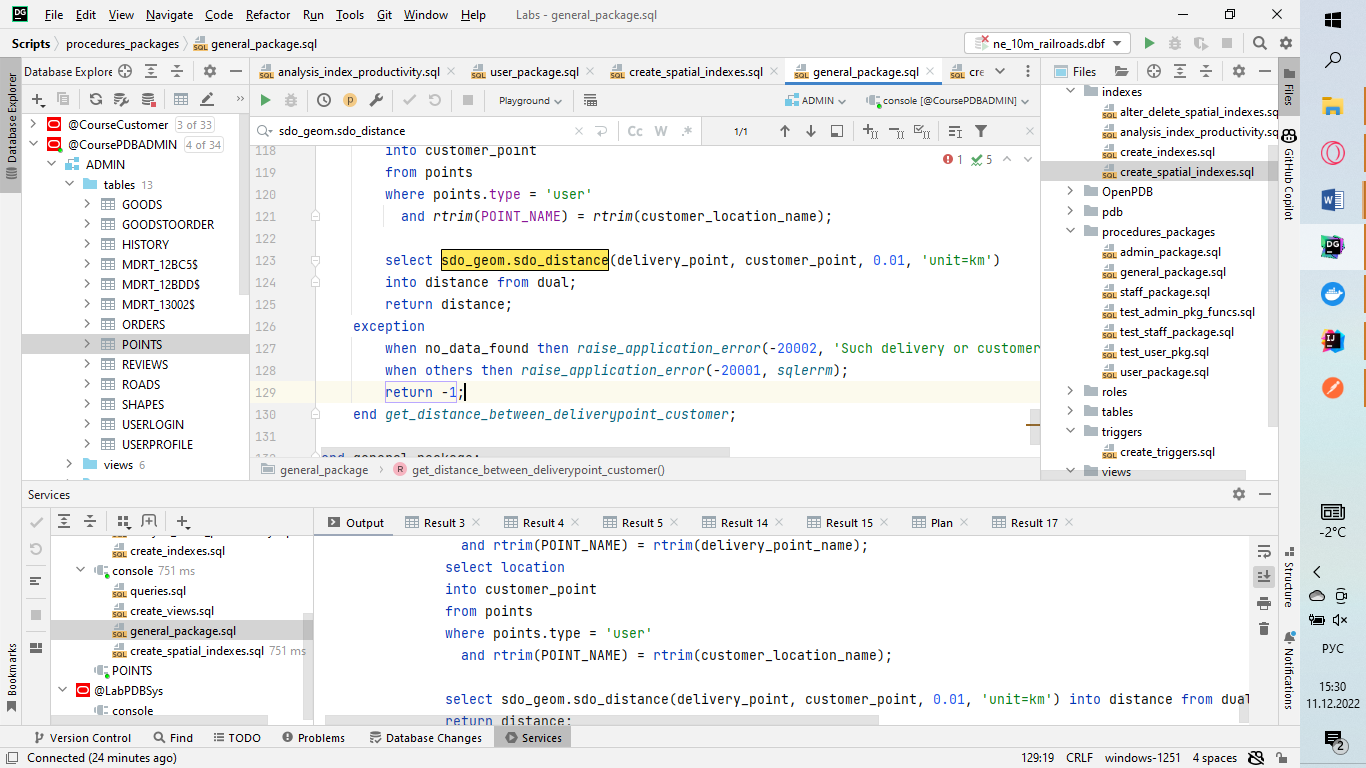


Рисунок 5.1 – Пример использования функции

Глава 6. Руководство пользователя

Для демонстрации работы базы данных было разработано приложение, написанное на таких языках, как «Java» и «JavaScript».

Программа, написанная на языке «Java», представляет собой серверную часть приложения, реализованная с использование такого фрэймворка, как «Spring». В базе данных существуют три пользователя(«manager», «user», «staff»). Данное приложение подключается к базе данных используя этих пользователей, чтобы в дальнейшем выполнять запросы под соответствующим пользователем и ролью. Так, например, разработан контроллер с названием «CustomerController», который принимает запросы с клиентской части приложения, описание которой будет представлено ниже, и выполняет их в базе данных под пользователем «customer». Например, такой метод как «getReviews» выполняет запрос к базе данных на получение всех записей из таблицы «Reviews» (Рисунок 6.1):

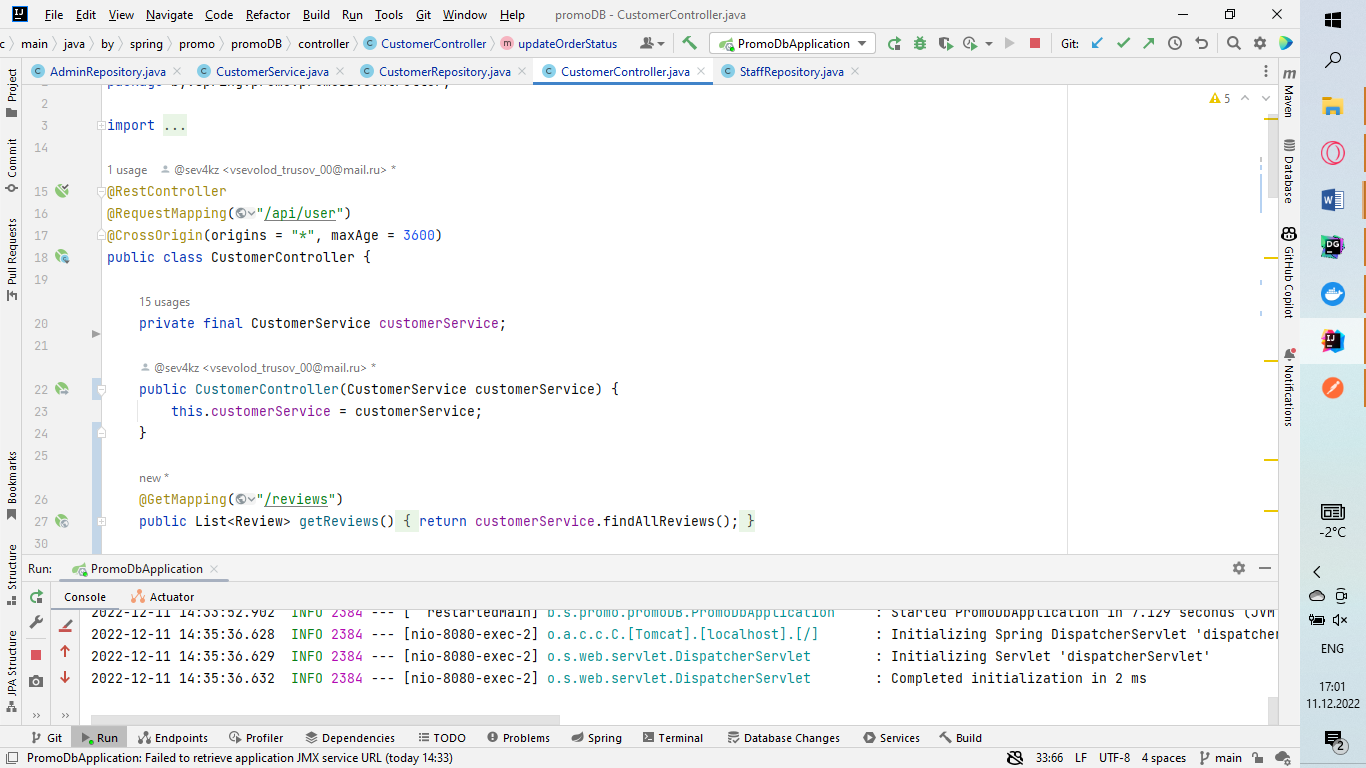


Рисунок 6.1 – Метод контроллера «CustomerController»

Второй составляющей, для демонстрации работы базы данных является программа, написанная на «JavaScript» с использование такой библиотеки, как «React». Данная программа представляет собой клиентскую часть приложения. С нее, используя различные url отправляются запросы к серверной части приложения. Например, на странице регистрации после нажатия на кнопку «Sign Up» происходит проверка корректности введенных данных и отправляется post-запрос к серверной части приложения с url «http://localhost:8080/api/user/registration», в теле данного запроса передаются данные, которые будут добавлены в базу данных (Рисунок 6.2):

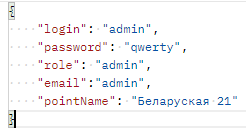


Рисунок 6.2 – Данные в теле post-запроса

Заключение

В данном курсовом проекте была разработана база данных по теме «Cервис доставки товаров» с использованием технологии «Spatial and Graph» для анализа пространственных данных. Также для демонстрации работы базы данных было разработано приложение, серверная часть которого написана на таком языке, как «Java» с использование фрэймворка «Spring». Клиентская часть приложения написана на языке «JavaScript» с использованием библиотеки «React».

Был реализован функционал ролей, описанных в листе задания:

* Пользователь
  1. оформление заказа,
  2. отмена заказа,
  3. нахождение ближайших пунктов выдачи товара,
  4. оставление оценки и отзыва по полученному товару
* Исполнитель
  1. выставление отметки по выполнению работы,
  2. просмотр отзывов
* Администратор
  1. назначение исполнителя полученному заказу,
  2. определение минимальной стоимости доставки,
  3. определение кратчайшего пути

А также были реализованы следующие функции:

* импорт и экспорт данных таблиц в XML формат;
* заполнение таблицы на 100 000 записей (таблица товаров).

Протестировав все компоненты приложения, можно прийти к заключению, что все основные требования выполнены и приложение работает исправно.

Список литературы

1. METANIT.COM Сайт о программировании [Электронный ресурс] / Режим доступа: https://metanit.com – Дата доступа: 15.10.2021.

2. Oracle-dba.ru [Электронный ресурс] / Режим доступа: https://oracle-dba.ru – Дата доступа: 27.10.2021.

3. Шифрование в PL/SQL: уроки программирования [Электронный ресурс] / Режим доступа: <https://oracle-patches.com/db/sql/3939-shifrovanie-v-pl-sql-uroki-programmirovaniya> – Дата доступа: 16.10.2021.

4. DBMS\_XMLDOM [Электронный ресурс] / Режим доступа: <https://docs.oracle.com/cd/B1930601/appdev.102/b14258/dxmldom.htm#i1076719> – Дата доступа: 20.10.2021.

5. Developing and Using Stored Procedures [Электронный ресурс] / Режим доступа: <https://docs.oracle.com/cd/B2835901/appdev.111/b28843/tdddgprocedures.htm> – Дата доступа: 26.10.2021.

6. Stackoverflow.com [Электронный ресурс] / Режим доступа: https://stackoverflow.com – Дата доступа: 08.12.2021

Приложение А

Таблица – Информация о пользователях и их профилях

|  |  |
| --- | --- |
| Имя пользователя | Профиль |
| customer | customer\_profile |
| staff | staff\_profile |
| manager | manager\_profile |
| admin | admin\_profile |

Таблица – Описание привилегий, выданных ролям

|  |  |
| --- | --- |
| Название роли | Привилегия |
| admin\_role | create indextype |
| execute any operator |
| create any directory |
| drop profile |
| create profile |
| create trigger |
| execute any procedure |
| create procedure |
| create role |
| create any sequence |
| create view |
| create any index |
| create table |
| drop user |
| create user |
| create session |
| manager\_role | create session |
| execute admin\_package,  general\_package |
| customer\_role | create session |
| execute user\_package,  general\_package |
| staff\_role | create session |
| execute staff\_package,  general\_package |

Приложение Б

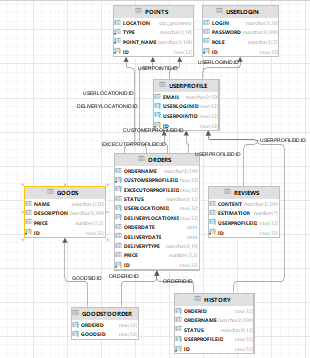


Схема базы данных

Приложение В

create or replace view unprocessed\_orders\_view  
as  
select o1.ordername as order\_name,  
 o1.orderdate as order\_date,  
 o1.deliverydate as delivery\_date,  
 o1.status as order\_status,  
 o1.price as order\_price,  
 o1.DELIVERYTYPE as delivery\_type,  
 g.name as good\_name,  
 userlogin.login as executor\_login,  
  
 admin\_package.*get\_customerlogin\_by\_order\_id*(o1.id) as customer\_login,  
  
 points.point\_name as delivery\_point,  
 admin\_package.*get\_userlocation\_by\_order\_id*(o1.id) as user\_point  
  
from orders o1  
 join points on o1.deliverylocationid = points.id  
 join goodstoorder gto on o1.id = gto.orderid  
 join goods g on gto.goodsid = g.id  
 join USERPROFILE on o1.excecutorprofileid = userprofile.id  
 join userlogin  
 on userprofile.userloginid = userlogin.id  
where o1.status = 'unprocessed';

create or replace view all\_points\_names\_view  
as  
select point\_name, type  
from points;

create or replace view processed\_orders\_view  
as  
select o1.ordername as order\_name,  
 o1.orderdate as order\_date,  
 o1.deliverydate as delivery\_date,  
 o1.price as order\_price,  
 o1.DELIVERYTYPE as delivery\_type,  
 o1.status as order\_status,  
 g.name as good\_name,  
 userlogin.login as executor\_login,  
 points.point\_name as delivery\_point,

Листинг – Cоздание представлений

admin\_package.*get\_customerlogin\_by\_order\_id*(o1.id) as customer\_login,  
 admin\_package.*get\_userlocation\_by\_order\_id*(o1.id) as user\_point  
from orders o1  
 join points on o1.deliverylocationid = points.id  
 join goodstoorder gto on o1.id = gto.orderid  
 join goods g on gto.goodsid = g.id  
 join USERPROFILE on o1.excecutorprofileid = userprofile.id  
 join userlogin  
 on userprofile.userloginid = userlogin.id;  
*--orders\_not\_executed\_view*create or replace view orders\_not\_executed\_view as  
select o1.ordername as order\_name,  
 o1.orderdate as order\_date,  
 o1.deliverydate as delivery\_date,  
 o1.status as order\_status,  
 o1.price as order\_price, o1.DELIVERYTYPE as delivery\_type,  
 goods.name as good\_name,  
 userlogin.login as customer\_login,  
 user\_package.*get\_executor\_login\_by\_order\_id*(o1.id) as executor\_login,  
 points.point\_name as delivery\_point,  
 admin\_package.*get\_userlocation\_by\_order\_id*(o1.id) as user\_point  
from orders o1  
 join points on o1.deliverylocationid = points.id  
 join goodstoorder on o1.id = goodstoorder.orderid  
 join goods on goodstoorder.goodsid = goods.id  
 join userprofile on o1.CUSTOMERPROFILEID = userprofile.id  
 join userlogin on userprofile.userloginid = userlogin.ID;create or replace view reviews\_view as  
select content, estimation, userlogin.login as login  
from reviews  
 join userprofile  
 on reviews.userprofileid = userprofile.id  
 join USERLOGIN  
 on userprofile.userloginid = userlogin.id;

create or replace view history\_view as  
select g.name as name,  
 h.status as status,  
 o.orderdate as order\_date,  
 o.deliverydate as delivery\_date,  
 ul.login as user\_login  
from orders o  
 join history h on h.orderid = o.id  
 join userprofile u on o.customerprofileid = u.id  
 join userlogin userloginid = ul.id  
 join goodstoorder gto on o.id = gto.orderid  
 join goods g on gto.goodsid = g.id;ul on u.

Листинг – Cоздание представлений

*--orders\_not\_executed\_view*create or replace view orders\_executed\_view  
as  
select o1.ordername as order\_name,  
 o1.orderdate as order\_date,  
 o1.deliverydate as delivery\_date,  
 o1.status as order\_status,  
 o1.price as order\_price,  
 o1.DELIVERYTYPE as delivery\_type,  
 goods.name as good\_name,  
 userlogin.login as customer\_login,  
 user\_package.*get\_executor\_login\_by\_order\_id*(o1.id) as executor\_login,  
 points.point\_name as delivery\_point,  
  
 admin\_package.*get\_userlocation\_by\_order\_id*(o1.id) as user\_point  
from orders o1  
 join points on o1.deliverylocationid = points.id  
 join goodstoorder on o1.id = goodstoorder.orderid  
 join goods on goodstoorder.goodsid = goods.id  
 join userprofile on o1.CUSTOMERPROFILEID = userprofile.id  
 join userlogin on userprofile.userloginid = userlogin.ID  
where o1.status = 'executed';  
create or replace view orders\_executed\_grouped\_view as  
select o1.ordername as order\_name,  
 o1.orderdate as order\_date,  
 o1.deliverydate as delivery\_date,  
 o1.status as order\_status,  
 o1.price as order\_price,  
 o1.DELIVERYTYPE as delivery\_type,  
 userlogin.login as customer\_login,  
 user\_package.*get\_executor\_login\_by\_order\_id*(o1.id) as executor\_login  
from orders o1  
 join points on o1.deliverylocationid = points.id  
 join goodstoorder on o1.id = goodstoorder.orderid  
 join goods on goodstoorder.goodsid = goods.id  
 join userprofile on o1.CUSTOMERPROFILEID = userprofile.id  
 join userlogin on userprofile.userloginid = userlogin.ID  
where o1.status = 'executed'  
group by o1.ordername, o1.orderdate, o1.deliverydate, o1.status, o1.price,  
o1.deliverytype, userlogin.login, o1.id ;

Листинг – Cоздание представлений

Приложение С

create index USERPROFILE\_USERLOGINID\_INDEX on USERPROFILE(USERLOGINID);  
create index USERPROFILE\_USERPOINTID\_INDEX on USERPROFILE(USERPOINTID);  
  
*--Orders table*drop index ORDERS\_CUSTOMERPROFILEID\_INDEX ;  
drop index ORDERS\_EXCECUTORPROFILEID\_INDEX;  
drop index ORDERS\_USERLCATIONID\_INDEX;  
drop index ORDERS\_DELIVERYLOCATIONID\_INDEX;  
drop index ORDERS\_STATUS\_INDEX;  
*---------------------*create index ORDERS\_CUSTOMERPROFILEID\_INDEX on ORDERS(CUSTOMERPROFILEID);  
create index ORDERS\_EXCECUTORPROFILEID\_INDEX on ORDERS(EXCECUTORPROFILEID);  
create index ORDERS\_USERLCATIONID\_INDEX on ORDERS(USERLOCATIONID);  
create index ORDERS\_DELIVERYLOCATIONID\_INDEX on ORDERS(DELIVERYLOCATIONID);  
create index ORDERS\_STATUS\_INDEX on ORDERS(STATUS);

create index GOODSTOORDER\_ORDERID\_INDEX on GOODSTOORDER(ORDERID);  
create index GOODSTOORDER\_GOODSID\_INDEX on GOODSTOORDER(GOODSID);  
  
*--Reviews table*drop index REVIEWS\_USERPROFILEID\_INDEX;  
*--------------------*create index REVIEWS\_USERPROFILEID\_INDEX on REVIEWS(USERPROFILEID);  
  
  
*--History table*drop index HISTORY\_ORDERID\_INDEX;  
*-----------------------------*create index HISTORY\_ORDERID\_INDEX on HISTORY(ORDERID);  
  
*--POINTS*  
create index POINTS\_TYPE\_INDEX on POINTS(TYPE);  
*--USERLOGIN*drop index USERLOGIN\_ROLE\_INDEX;  
*------------------*create index USERLOGIN\_ROLE\_INDEX on USERLOGIN(ROLE);  
*--GOODS*drop index GOODS\_NAME\_INDEX;  
*------------------*create index GOODS\_NAME\_INDEX on GOODS(NAME);

create index POINTS\_SP\_INDEX on POINTS(LOCATION) indextype is mdsys.spatial\_index;

Листинг – Cоздание индексов

Приложение Д

Таблица – Описание процедур и функций пакета «admin\_package»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя функции(процедуры) | Название параметра | Тип данных | Тип параметра |
| delete\_reviews | - | - | - |
| update\_order\_executor\_deliverypoint | order\_name | nvarchar2 | in |
| order\_executor\_login | nvarchar2 | in |
| deliverypoint\_name | nvarchar2 | in |
| get\_order\_price | number | in |
| get\_unprocessed\_orders | - | ref cursor | out |
| delete\_good\_by\_name | good\_name | nvarchar2 | in |
| add\_good | good\_name | nvarchar2 | in |
| good\_description | nvarchar2 | in |
| good\_price | number | in |
| get\_executors\_by\_point\_name | - | ref cursor | out |
| get\_point\_name | nvarchar2 | in |
| get\_all points\_name | - | ref cursor | out |
| get all persons by role | - | ref cursor | out |
| role | nvarchar2 | in |
| get\_customerlogin\_by\_order\_id | - | nvarchar2 | out |
| order\_id | raw | in |
| get\_userlocation\_by\_order\_id | - | nvarchar2 | out |
| order\_id | raw | in |
| get\_count\_orders\_by\_executor\_login\_id | - | number | out |
| userlogin\_id | raw | in |
| get\_count\_unprocessed\_orders | - | number | out |
| get\_count\_staff | - | number | out |
| get\_user\_count | - | number | out |
| get\_admin\_info | - | ref cursor | out |

Таблица – Описание процедур и функций пакета «general\_package»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя функции(процедуры) | Название параметра | Тип данных | Тип параметра |
| get\_distance\_between\_deliverypoint\_customer |  | number | out |
| delivery\_point\_name | nvarchar2 | in |
| customer\_location\_name | nvarchar2 | in |
| get\_route\_length\_analysis | - | ref cursor | out |
| customer\_point\_name | nvarchar2 | in |
| change\_order\_status\_by\_name | order\_name | nvarchar2 | in |
| get\_status | nvarchar2 | in |
| get\_reviews | - | ref cursor | out |
| get\_pagination\_goods | - | ref cursor | out |
| start\_value | number | in |
| count\_rows\_of\_goods | - | number | out |

Таблица – Описание процедур и функций пакета «staff\_package»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя функции(процедуры) | Название параметра | Тип данных | Тип параметра |
| get\_processed\_order\_to\_staff\_by\_login | - | ref cursor | out |
| staff\_login | nvarchar2 | in |
| get\_processed\_orders\_count\_by\_\_login |  | number | out |
| staff\_login | nvarchar2 | in |

Таблица – Описание процедур и функций пакета «staff\_package»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя функции(процедуры) | Название параметра | Тип данных | Тип параметра |
| get\_processed\_order\_to\_staff\_by\_login | - | ref cursor | out |
| staff\_login | nvarchar2 | in |
| get\_processed\_orders\_count\_by\_\_login | - | number | out |
| staff\_login | nvarchar2 | in |

Таблица – Описание процедур и функций пакета «user\_package»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя функции(процедуры) | Название параметра | Тип данных | Тип параметра |
| dencrypt\_password | - | varchar2 | out |
| password\_hash | varchar2 | in |
| encrypt\_password | - | nvarchar2 | out |
| password | varchar2 | in |
| add\_review | get\_content | nvarchar2 | in |
| get\_estimation | number | in |
| get\_login | nvarchar2 | in |
| delete\_order\_by\_name | order\_name | nvarchar2 | in |
| get\_good\_by\_name | - | ref cursor | out |
| good\_name | nvarchar2 | in |
| get\_all\_goods | - | ref cursor | out |
| get\_history\_by\_login | - | ref cursor | out |
| customer\_login | nvarchar2 | in |
| get\_orders\_not\_executed\_by\_login | - | ref cursor | out |
| user\_login | nvarchar2 | in |
| add\_order | - | nvarchar2 | out |
| customer\_login | nvarchar2 | in |
| good\_name | nvarchar2 | in |
| get\_data\_order | date | in |
| get\_delivery\_date | date | in |
| get\_delivery\_type | nvarchar2 | in |
| get\_order\_price | number | in |
| get\_delivery\_point\_pickup | nvarchar2 | in |
| get\_routes\_by\_user\_login | - | ref cursor | out |
| user\_login | nvarchar2 | in |
| get\_executor\_login\_by\_order\_id | - | nvarchar2 | out |
| order\_id | raw | in |
| insert\_into\_orders | - | raw | out |
| customer\_profile\_id | raw | in |
| executor\_profile\_id | raw | in |
| start\_deliverylocation | raw | in |
| order\_status | nvarchar2 | in |
| userlocation\_id | raw | in |
| get\_data\_order | date | in |
| get\_delivery\_date | date | in |
| get\_delivery\_type | nvarchar2 | in |
| get\_order\_price | number | in |
| add\_goods\_to\_order | order\_id | raw | in |
| good\_id | raw | in |
| add\_good\_to\_order | order\_name | nvarchar2 | in |
| good\_name | nvarchar2 | in |

|  |  |  |  |
| --- | --- | --- | --- |
| Имя функции(процедуры) | Название параметра | Тип данных | Тип параметра |
| get\_orders\_count\_by\_status\_or\_all | - | number | Out |
| user\_login | nvarchar2 | in |
| get\_status | nvarchar2 | in |
| get\_all | nvarchar2 | in |
| get\_customer\_info | - | ref cursor | out |
| user\_login | nvarchar2 | in |
| authorisation | - | ref cursor | out |
| get\_login | nvarchar2 | in |
| get\_password | nvarchar2 | in |
| register\_user | get\_login | nvarchar2 | in |
| get\_userpoint\_name | nvarchar2 | in |
| password | nvarchar2 | in |
| get\_role | nvarchar2 | in |
| get\_email | nvarchar2 | in |

Таблица – Информация о таблицах

|  |  |  |
| --- | --- | --- |
| Название таблицы | Название колонки | Тип данных |
| orders | id | raw |
| ordername | nvarchar2 |
| customerprofileid | raw |
| excecutorprofileid | raw |
| status | nvarchar2 |
| userlocationid | raw |
| deliverylocationid | raw |
| orderdate | date |
| deliverydate | date |
| deliverytype | nvarchar2 |
| price | number |
| points | id | raw |
| location | sdo\_geometry |
| point\_name | nvarchar2 |
| type | nvarchar2 |
| roads | id | raw |
| road\_name | nvarchar2 |
| road | sdo\_geometry |
| goodstoorder | orderid | raw |
| goodsid | raw |
| history | id | raw |
| orderid | raw |
| goods | id | raw |
| name | nvarchar2 |
| description | nvarchar2 |
| price | number |
| userlogin | id | raw |
| login | nvarchar2 |
| password | nvarchar2 |
| role | nvarchar2 |
| userprofile | id | raw |
| email | nvarchar2 |
| userloginid | raw |
| userpointid | raw |
| reviews | id | raw |
| content | nvarchar2 |
| estimation | number |
| userprofileid | raw |