ИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПЕУБЛИКИ БЕЛАРУСЬ

Учреждение образования   
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет Информационных технологий

Кафедра Информационные системы и технологии

Специальность 1-40 05 01 Информационные системы и технологии

Направление специальности 1-40 01 02 03 Информационные системы

и технологии (издательско-полиграфический комплекс)

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

«База данных «Кофейня» с применением резервного копирования и восстановления данных»

Выполнил студент Гринцевич Ю.С.

(Ф.И.О.)

Руководитель проекта ассистент Нистюк О.А.

(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)

Курсовой проект защищен с оценкой

Минск 2023

**Оглавление**

[Введение 3](#_Toc134798736)

[1 Постановка задачи 4](#_Toc134798737)

[2 Аналитический обзор литературы 5](#_Toc134798738)

[3 Проектирование базы данных 7](#_Toc134798739)

[4 Разработка объектов базы данных 9](#_Toc134798740)

[5 Импорт и экспорт данных 11](#_Toc134798741)

[6 Тестирование производительности 13](#_Toc134798742)

[7 Описание технологии и ее применения в базе данных 14](#_Toc134798743)

[8 Руководство пользователя 22](#_Toc134798744)

[Заключение 23](#_Toc134798745)

[Cписок источников 24](#_Toc134798746)

[Приложение А 25](#_Toc134798747)

[Приложение Б 27](#_Toc134798748)

[Приложение В 29](#_Toc134798749)

[Приложение Г 33](#_Toc134798750)

# **Введение**

База данных — это организованная структура, предназначенная для хранения информации, систематизированная таким образом, чтобы эти материалы могли быть найдены и обработаны. Реляционная база данных — база данных, основанная на реляционной модели данных. В качестве СУБД для базы данных была выбрана Oracle 12c.

В основной части будут затронуты все аспекты разработки проекта и обоснованы некоторые технические приёмы, к которым приходилось прибегнуть, с целью реализации работы с базой данных.

Любая организация нуждается в своевременном доступе к информации. Ценность информации в современном мире очень высока. Роль распорядителей информации в современном мире чаще всего выполняют базы данных. Базы данных обеспечивают надежное хранение информации, в структурированном виде и своевременный доступ к ней. Практически любая современная организация нуждается в базе данных, удовлетворяющей те или иные потребности по хранению, управлению и администрированию данных.

За последние несколько лет наблюдается тенденция к усложнению структур данных. Простые виды информации, представимой в форме чисел и текстовых строк, не утратив своей значимости, дополняются сегодня многочисленными мультимедийными документами, графическими образами, хронологическими рядами, процедурными, или активными, данными и мириадами прочих сложных информационных форм.

На сегодняшний день на рынке представлено множество технологий доступа к данным и серверов баз данных, каждое, из которых имеет свои отличительные черты. Современные приложения обработки данных ориентированы на работу с большим количеством пользователей, на их удаленность от места расположения основного сервера БД.

Темой данного курсового проекта является разработка базы данных, обеспечивающая работу с базой данных «Кафе».

Автоматизация процессов учета и оформления заказов должна включать в себя разработку системы, обеспечивающую хранение и обработку данных о блюдах, которые предлагает заведение, а также сотрудников, оформляющих заказ.

База данных «Кофейня» существенно упрощает работу сотрудников кофейни, а также предоставляет возможность администратору своевременно вносить необходимые изменения. Данный программный продукт позволяет автоматизировать работу любого заведения ресторанного типа.

Целью данной работы является разработка реляционной базы данных для кофейни: «Кофейня». Эта база данных должна составлялась для обеспечения управления основными задачами кофейни: создавать и удалять заказы, редактировать информацию о товарах, которые предлагает кофейня, а также работников кофейни.

# **1 Постановка задачи**

Целью данной курсовой работы является создание программного обеспечения для автоматизации процесса администрирования процессов приема и увольнения сотрудников и создание, и изменение формации о блюдах, предлагаемых кофейней, кроме этого, приложение позволяет администрировать процессы оформления и изменения заказов.

Задача проекта: совершенствование практических навыков в использовании и разработке базы данных. Создание базы данных и выполнение тестирования базы данных.

Функционально должны быть выполнены следующие задачи:

– редактировать информацию о товарах;

– сотрудник создает и удаляет заказ;

– администратор получает справки о выручке за определенный период;

– администратор получает все заказы за определенный период;

Должны быть выполнены следующие требования:

– База данных должна быть реализована в СУБД Oracle 12c.

– Доступ к данным должен осуществляться только через процедуры.

– Количество объектов БД (таблиц, представлений, индексов, пользователей и пр.) регламентируется задачей.

– Должен быть проведен импорт и экспорт данных.

– Необходимо протестировать производительность базы данных на таблице, содержащей не менее 100 000 строк, и внести изменения в структуру в случае необходимости. Необходимо проанализировать планы запросов к таблице.

– Применить технологию базы данных согласно выбранной теме: подробно описать применяемые системные пакеты, утилиты или технологии; показать применение указанной технологии в базе данных.

– Листинги проекта должны содержать комментарии.

Результатом данной курсовой работы является проект, который будет осуществлять взаимодействие с разработанной базой данных «Coffe».

В процессе выполнения работы были поставлены следующие задачи:

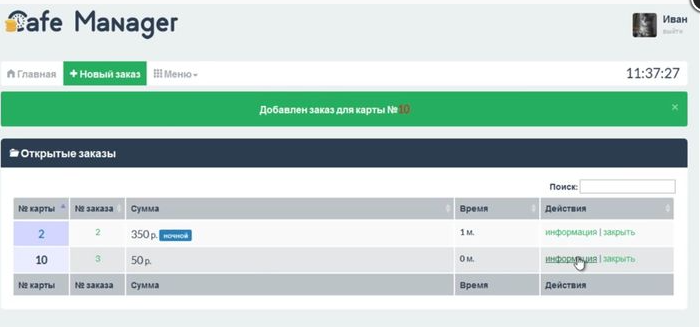
1. Провести аналитический обзор литературы, а также поиск и сравнение аналогов.
2. Спроектировать и разработать базу данных.
3. Провести тестирование производительности базы данных

# **2 Аналитический обзор литературы**

Объектом разработки является база данных для работы кофейни. Кофейня — гастрономическое заведение, место для встреч и общения, где подают напитки и десертные блюда. Заведения подобного плана чаще всего являются местом для деловых встреч, работы, отдыха за счет своей непринужденной обстановки. Поэтому очень важно не допускать ошибок в работе персонала.

При разработке своей базы данных я также обратила внимания на самые успешные примеры популярнейших приложений, которые работают с аналогичными моей базами данных. Рассмотрим их ниже.

Cafe Manager это удобная система для автоматизации и управления любым заведением, как ориентированное на оплату по времени, так и классическое кафе без оплаты времени. Интерфейс этого сервиса можно увидеть на рисунке 2.1



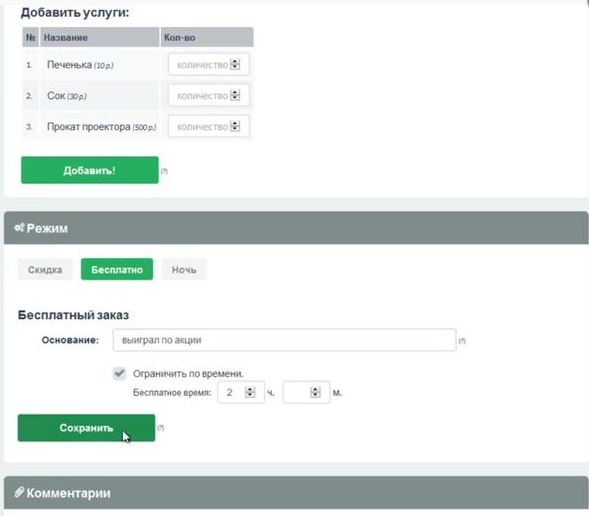


Рисунок 2.1 – Главная страница приложения «Cafe Manager»

Quick Resto – eдиная система для управления и развития ресторанных проектов всех форматов: онлайн-касса, сайт, мобильное приложение, доставка, CRM, склад и аналитика продаж в облаке. Интерфейс этого сервиса можно увидеть на рисунке 2.1



Рисунок 2.1 – Главная страница приложения «Quick Resto»

Разобрав три, приведенных выше, сервиса для администрирования заведений ресторанного типа, можно сформировать общий функционал необходимый для работы приложения.

Прежде всего, естественно, база данных должна корректно работать с большим количеством данных, при этом не быть перегружена. Обязательно должна быть функция отображения всего списка блюд в заказе.

Также стоит ввести возможность просмотра простейшей статистики заказов за определенные промежутки времени для администратора.

# **3 Проектирование базы данных**

Важным этапом курсовой работы является создание логически взаимосвязанных таблиц. Чтобы составить визуальную взаимосвязанную структуру нашей базы данных, нам необходимо продумать, какая информация будет храниться в этих таблицах, после этого создать связи с помощью первичных и внешних ключей.

Логическое описание базы данных:

1. Таблица "Товары" должна содержать информацию о товарах, которые продаются в кофейне:

* Id товара (ключевое поле);
* название товара;
* цену товара;
* индикатор, указывающий, удален ли товар.

2. Таблица "Товары в заказе" должна содержать информацию о товарах, которые были заказаны посетителями кофейни:

* Id товара в заказе (ключевое поле);
* Id заказа, к которому относится данный товар;
* Id товара;
* количество заказанных товаров.

3. Таблица "Заказы" должна содержать информацию о заказах, которые были сделаны посетителями кофейни:

* Id заказа (ключевое поле);
* Id пользователя, который создал или изменил заказ;
* дата создания или изменения заказа;
* размер скидки на заказ;
* индикатор, указывающий, удален ли заказ.

4. Таблица "Пользователи" должна содержать информацию о пользователях, которые имеют доступ к базе данных:

* Id пользователя (ключевое поле);
* имя пользователя;
* телефон пользователя;
* дата создания учетной записи;
* дата рождения пользователя;
* индикатор, указывающий, уволен ли пользователь;
* логин пользователя;
* пароль пользователя;
* идентификатор статуса пользователя.

5. Таблица "Статусы" должна содержать информацию о статусах пользователей, которые имеют доступ к базе данных кофейни:

* Id статуса (ключевое поле);
* название статуса;

Диаграмма базы данных, спроектированной в ходе анализа информации представлена на рисунке 3.1.

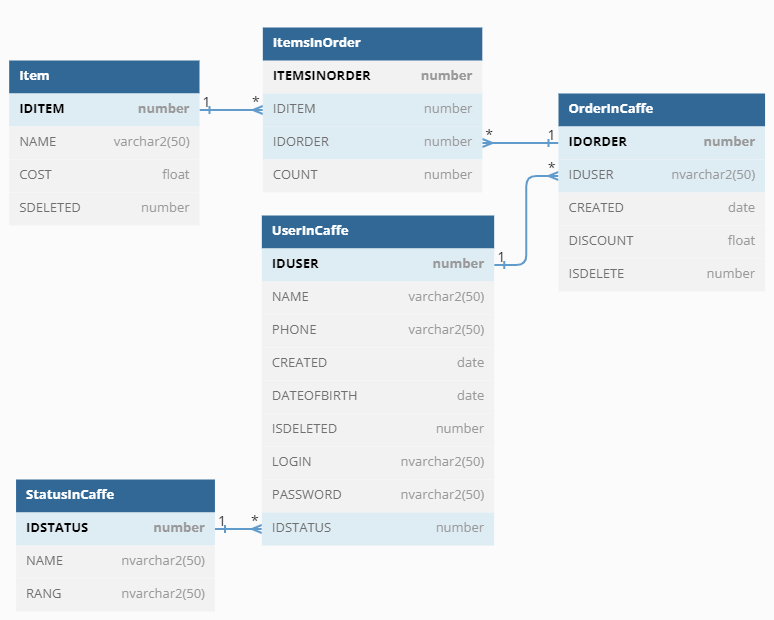


Рисунок 3.1 – Диаграмма базы данных

По результатам анализа данных база данных будет состоять из следующих таблиц:

* Item;
* ItemsInOrder;
* OrderInCaffe;
* UserInCaffe;
* StatusInCaffe;

# **4 Разработка объектов базы данных**

В ходе проектирования базы данных было создано 5 таблиц, связанных между собой внешними ключами. Скрипты создания таблиц хранятся в приложении А, скрипты создания пользователей хранятся в приложении Б.

Таблица «Item» хранит список блюд, которые кофейня предлагает посетителям. Поля таблицы представлены в таблице 4.1.

Таблица 4.1 – Столбцы таблицы Item

|  |  |  |
| --- | --- | --- |
| Наименование | Тип | Описание |
| IDITEM | int | идентификатор товара, первичный ключ |
| NAME | varchar(50) | название товара |
| COST | float | стоимость товара |
| ISDELETED | CHAR(1) | индикатор, означающий удален ли товар |

Таблица «ItemsInOrder» содержит список всех товаров, которые были заказаны посетителями, а также их количество. Поля таблицы представлены в таблице 4.2.

Таблица 4.2 – Столбцы таблицы ItemsInOrder

|  |  |  |
| --- | --- | --- |
| Наименование | Тип | Описание |
| ITEMSINORDER | int | идентификатор объекта, первичный ключ |
| IDORDER | int | идентификатор заказа, которому принадлежит данный заказ, внешний ключ к таблице «OrderInCaffe» |
| IDITEM | int | идентификатор блюда, которое заказал посетитель, внешний ключ к таблице «Item» |
| COUNT | Number(10,2) | количество блюд в заказе |

Таблица «OrderInCaffe» содержит список заказов. Важно понимать, что в данной таблице нет списка блюд, которые заказал гость, они находятся в таблице «ItemsInOrder» Поля таблицы представлены в таблице 4.3.

Таблица 4.3 – Столбцы таблицы OrderInCaffe

|  |  |  |
| --- | --- | --- |
| Наименование | Тип | Описание |
| IDORDER | int | идентификатор заказа, первичный ключ |
| IDUSER | int | идентификатор пользователя, который создал/изменил заказ, внешний ключ к таблице «UsersInCaffe» |
| CREATED | date | дата создания или изменения заказа |
| DISCOUNT | float | размер скидки на заказ |
| ISDELETED | CHAR(1) | индикатор, означающий удален ли заказ |

Таблица «StatusInCaffe» содержит в себе список статусов, а также уровень привилегий статуса. Такая таблица позволит динамически расширять количество статусов в базе данных, что позволяет модифицировать полномочия пользователей приложения. Поля таблицы представлены в таблице 4.4.

Таблица 4.4 – Столбцы таблицы StatusInCaffe

|  |  |  |
| --- | --- | --- |
| Наименование | Тип | Описание |
| STATUS | int | идентификатор статуса, первичный ключ |
| NAME | nvarchar(50) | название статуса |

Таблица «UserInCaffe» содержит данные о сотрудниках, которые имеют доступ к системе. Поля таблицы представлены в таблице 4.5.

Таблица 4.5 – Столбцы таблицы UserInCaffe

|  |  |  |
| --- | --- | --- |
| Наименование | Тип | Описание |
| IDUSER | int | идентификатор аккаунта, первичный ключ |
| NAME | varchar(50) | имя сотрудника |
| PHONE | varchar(10) | телефон сотрудника |
| CREATED | date | дата создания учетной записи |
| DATEOFBIRTH | date | дата рождения сотрудника |
| ISDELETED | CHAR(1) | индикатор, означающий уволен ли сотрудник |
| LOGIN | nvarchar(50) | логин сотрудника |
| PASSWORD | nvarchar(50) | пароль сотрудника |
| STATUS | nvarchar(50) | идентификатор статуса пользователя, который является сотрудником кофейни, внешний ключ к таблице «StatusInCaffe» |

Следует отметить, что поля «ISDELETED» являются обязательными и неизбежными, исходя из налогового кодекса Республики Беларусь, а также некоторых правил бухгалтерии, из баз данных нельзя удалять блюда, заказы, а также уволенных сотрудников по причине того, что данные таблицы используются в ежеквартальных отчетах, а значит влияют на количество выручки или качество отчетности по оформленным заказам за квартал.

Помимо таблиц в базе данных были созданы объекты: процедуры для выполнения основных действий с данными в базе данных, функции, последовательности и пользователи.

Все процедуры, связанные с таблицами и пользователями хранятся в отдельных пакетах процедур. Все select запросы хранятся в функциях, которые создают pipeline. Скрипты функций и процедур хранятся в приложении В.

В каждой из процедур, для избежания исключительных ситуаций, происходит обработка исключений с выводом сообщения об ошибке.

Для разграничения прав доступа было создана два пользователя. Один является администратором базы данных, а второй – обычный пользователь. Администратор имеет возможность добавлять сотрудников, блюда и заказы. Второй пользователь имеет ограниченные права на выполнение процедур.

Также была создана последовательность для нумерации заказов, блюд и пользователей, которая каждому новому объекту ставит идентификатор на единицу больше предыдущего.

# **5 Импорт и экспорт данных**

Импорт и экспорт данных — это автоматический или полуавтоматический ввод и вывод наборов данных между различными программными приложениями. В курсовой работе импорт и экспорт данных был произведен как с помощью процедур вставки, так и с помощью XML файлов.

Скрипт создания процедуры импорта и экспорта можно посмотреть в приложении Г. Скриншот XML файла для импорта изображен на рисунке 5.1

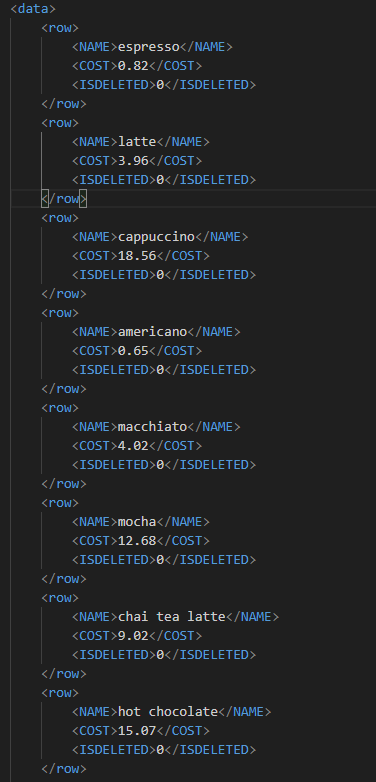


Рисунок 5.1 – Наполнение XML файла

Для импорта была разработана функция ReadFileForImport(), которая читает содержимое файла и возвращает его в виде объекта XMLTYPE. Затем определяется процедура ImportXMLItems(), которая вызывает функцию ReadFileForImport для чтения содержимого XML файла, затем извлекает данные из XML файла с помощью функции EXTRACTVALUE, и вставляет их в таблицу ITEM с помощью оператора INSERT INTO. В качестве параметра она принимает имя xml-файла.

Для экспорта была разработана процедура Export\_Items(). Она выполняет экспорт данных из таблицы Item в XML файл items\_export.xml с помощью пакета DBMS\_XMLDOM. Для этого сначала создается каталог EXSPORTFILEXML, затем определяется процедура export\_data, которая открывает курсор для выборки данных из таблицы Products, создает XML документ с помощью функции DBMS\_XMLDOM.NewDomDocument, и записывает его в файл items\_export.xml с помощью функции DBMS\_XMLDOM.WRITETOFILE. Содержимое файла items\_export.xml представлено на рисунке 5.2.

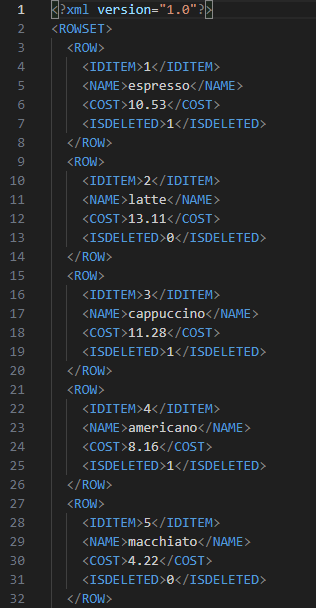


Рисунок 5.1 – Наполнение файла items\_export.xml

# **6 Тестирование производительности**

Для тестирования производительности заполним таблицу ItemsInOrder на 100000 строк при помощи анонимного блока, представленного на рисунке 6.1.

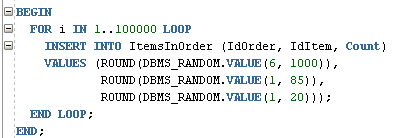


Рисунок 6.1 – анонимный блок заполнения таблицы 100000 строками

Для увеличения скорости select-запроса нужно создать индекс на таблицу, которую заполнили 1000000 строками. Цена запроса во вкладке «План запроса» до создания индекса представлена на рисунке 6.2.

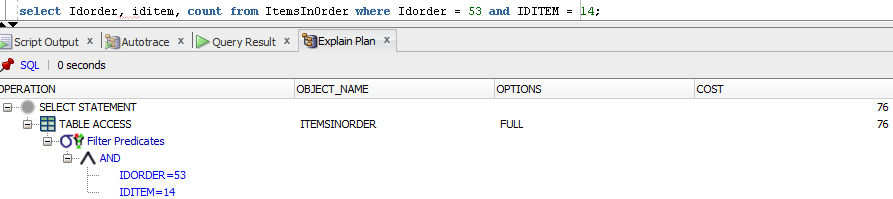


Рисунок 6.2 – план запроса до создания индекса

После создания индекса цена запроса уменьшилась, что можно наблюдать на рисунке 6.3

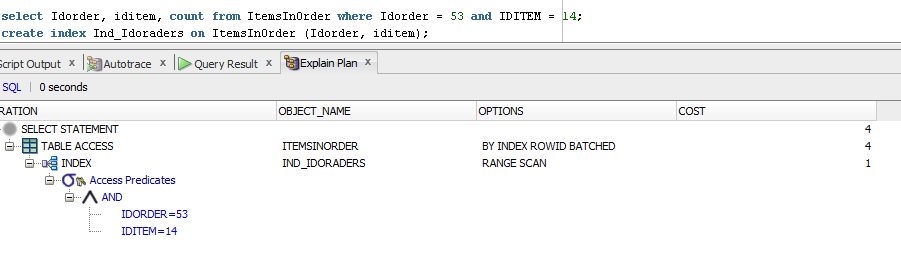


Рисунок 6.3 – план запроса после создания индекса

# **7 Описание технологии и ее применения в базе данных**

В курсовом проекте для реализации была выбрана технология: резервное копирование и восстановление данных.

Резервное копирование – процесс создания копии данных на носителе, предназначенном для восстановления данных в оригинальном или новом месте их расположения в случае их повреждения или разрушения. В случае применения утилиты RMAN резервное копирование файлов базы данных Oracle выполняться внутри базы данных посредством самого сервера баз данных.

RMAN – это инструмент, предоставляемый Oracle для резервного копирования и восстановления базы данных. Он позволяет создавать резервные копии базы данных, а также восстанавливать ее в случае сбоя.

RMAN умеет делать резервные копии и копии образов файлов данных, управляющих файлов, архивных журналов повторного выполнения, файлов SPFILE. Утилита RMAN восстанавливает файлы данных из резервных копий (backup) и применяет необходимые архивные журналы повторного выполнения для приведения базы данных в актуальное состояние.

Утилита RMAN упрощает процедуры резервного копирования, позволяя использовать мощные и в то же время легкие в написании сценарии резервного копирования и восстановления.

В курсовой работе был осуществлен следующие команды для демонстрации восстановления удаленной базы данных из резервной копии:

* Создаем папку для резервной копии. Демонстрация этого действия представлена на рисунке 7.1.



Рисунок 7.1 – Создание папки

* Открываем командную строку с правами администратора.
* Остановим все службы Oracle 12c, используя команду. Демонстрация этого действия представлена на рисунке 7.2.



Рисунок 7.2 – Демонстрация остановки всех служб oracle 12c

* Проверяем состояние нашей базы данных. База данных должна создавать архивные журналы и находиться в режиме чтения и записи. Демонстрация этого действия представлена на рисунке 7.3.

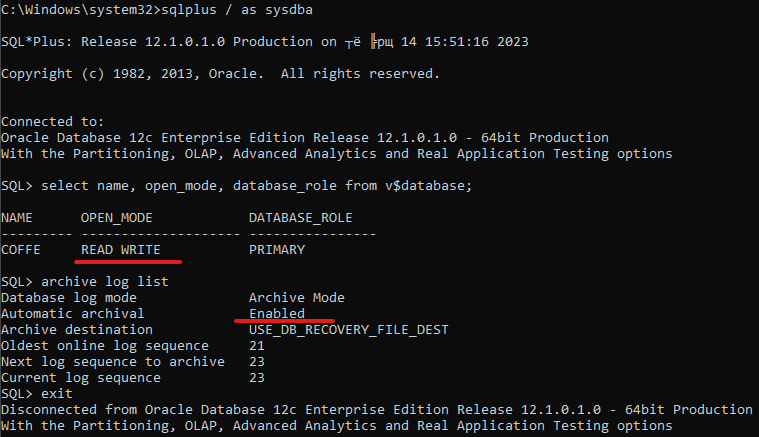
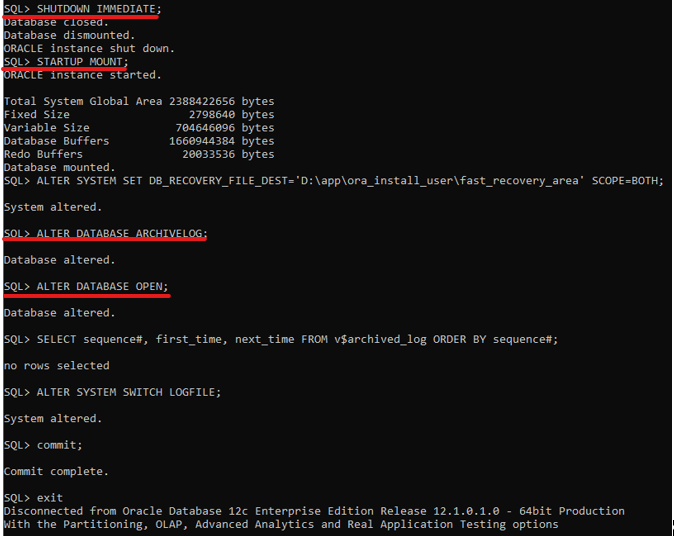


Рисунок 7.3 – Проверка состояния базы данных

* Если состояние базы данных NOARCHIVELOG то переводим ее в режим ARCHIVELOG. Демонстрация этого действия представлена на рисунке 7.4.



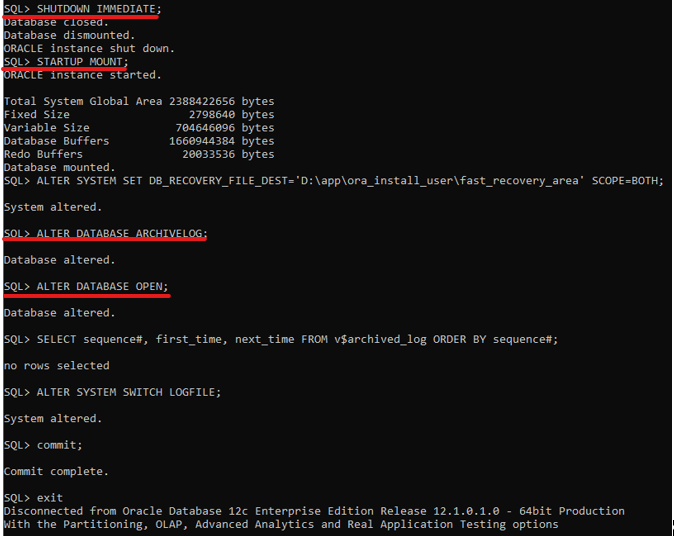


Рисунок 7.4 – Перевод базы данных в режим ARCHIVELOG

* Перейдем в RMAN и создадим сжатую резервную копию с журналом архивов в созданной директории. Демонстрация этого действия представлена на рисунке 7.5.

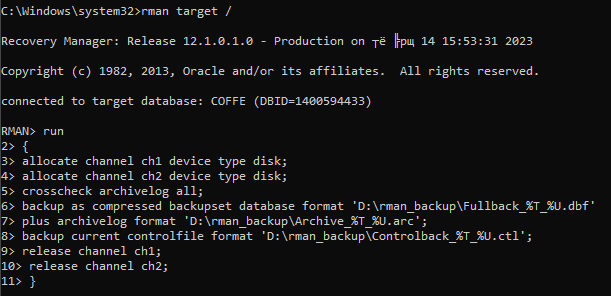


Рисунок 7.5 – Создание резервной копии

* Проверяем успешность завершения резервного копирования. Демонстрация этого действия представлена на рисунке 7.6.

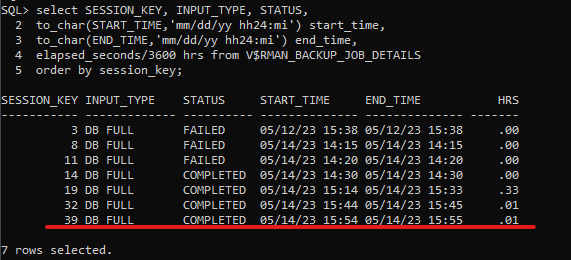


Рисунок 7.6 – Создание резервной копии

* Создаем копию файла spfileCOFFE.ora в файле spfileCOFFE.ora\_bkp. Демонстрация этого действия представлена на рисунке 7.7.

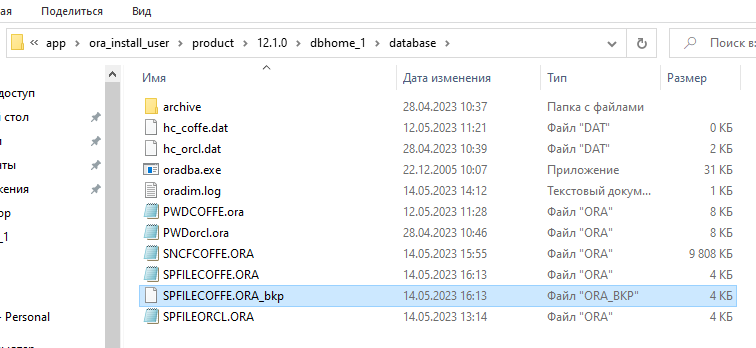


Рисунок 7.7 – Создаем копию файла spfileCOFFE.ora

* Останавливаем базу данных. Демонстрация этого действия представлена на рисунке 7.8.

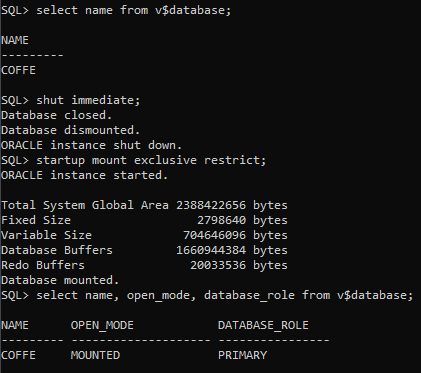


Рисунок 7.8 – Останавливаем базу данных

* Удаляем базу данных. Демонстрация этого представлена на рисунке 7.9.

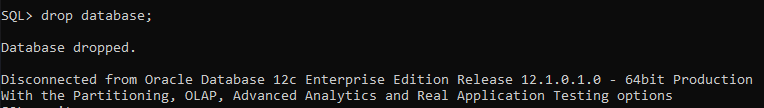


Рисунок 7.9 – Удаляем базу данных

* Проверяем, что база данных была удалена. Демонстрация этого действия представлена на рисунке 7.10.

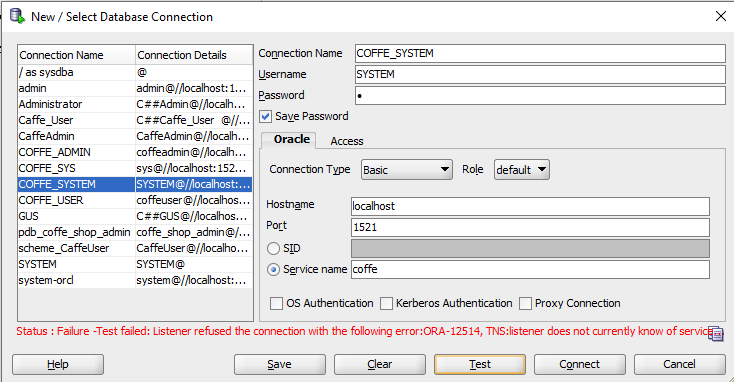
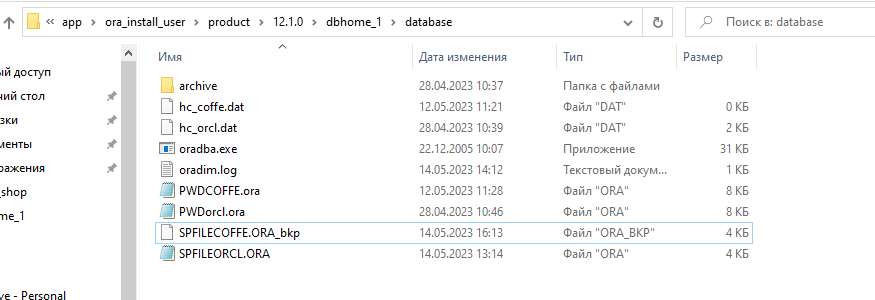


Рисунок 7.10 – Проверка удаленной базы данных

* Восстанавливаем файл spfileCOFFE.ora. Демонстрация этого действия представлена на рисунке 7.11.

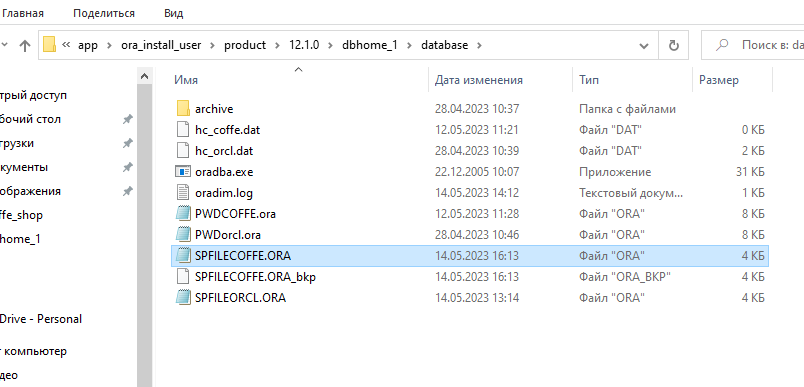


Рисунок 7.11 – Восстанавливаем файл spfileCOFFE.ora

* Запускаем экземпляр базы данных без монтирования данных. Демонстрация этого действия представлена на рисунке 7.12.

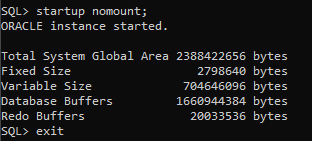


Рисунок 7.12 – Запускаем экземпляр базы данных без монтирования

* Указываем путь к контрольным файлам для восстановления. Демонстрация этого действия представлена на рисунке 7.13.

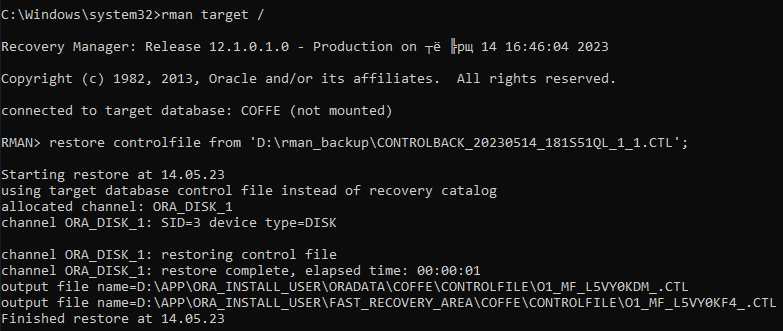


Рисунок 7.13 – Указываем контрольные файлы

* Загружаем управление файлами управления базы данных в память и проверяем на наличие ошибок. Демонстрация этого действия представлена на рисунке 7.14.

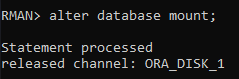


Рисунок 7.14 – Монтируем базу данных

* Сканируем каталог и добавляем информацию о всех найденных там резервных копиях и архивных журналах. Демонстрация этого действия представлена на рисунке 7.15.

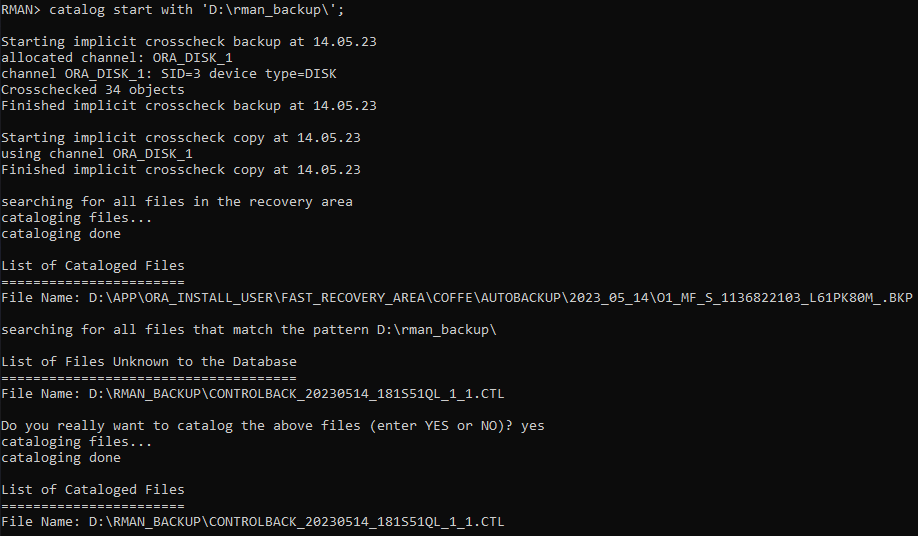


Рисунок 7.15 – Сканируем каталог

* Запускаем восстановление базы данных. Демонстрация этого действия представлена на рисунке 7.16.

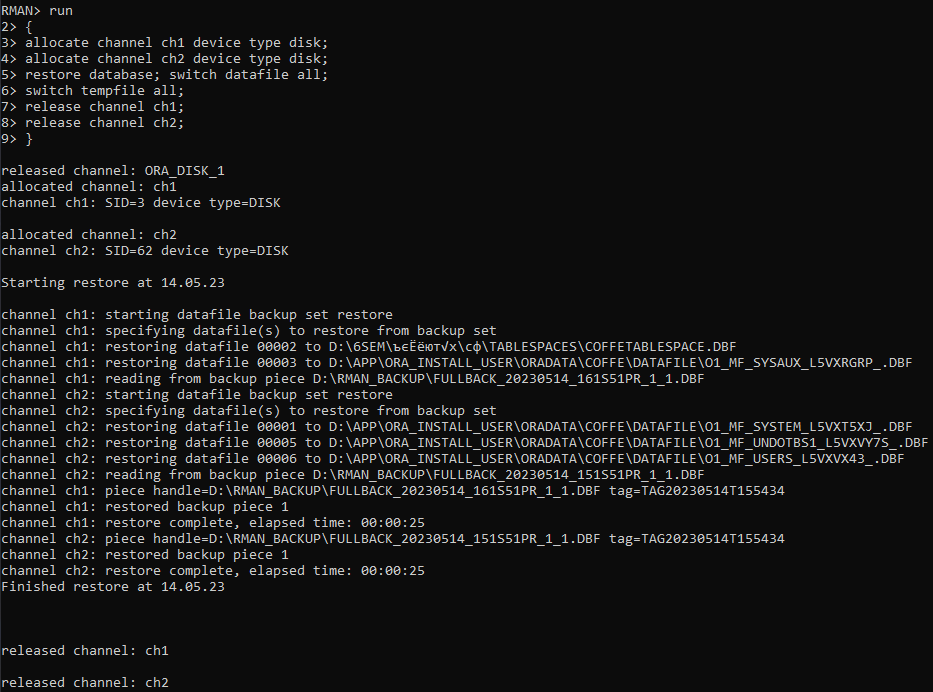


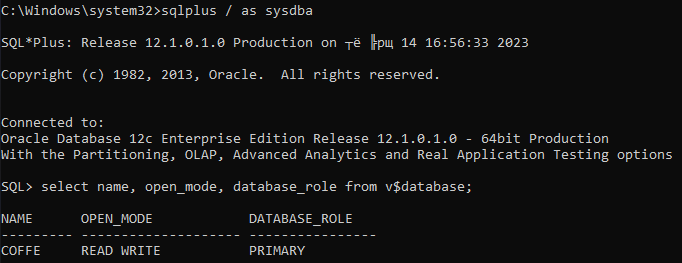
Рисунок 7.16 – Запускаем восстановление

* Открываем базу данных после восстановления. Демонстрация этого действия представлена на рисунке 7.17.



Рисунок 7.17 – Открываем базу данных

* Проверяем работоспособность базы данных. Демонстрация этого действия представлена на рисунке 7.18.



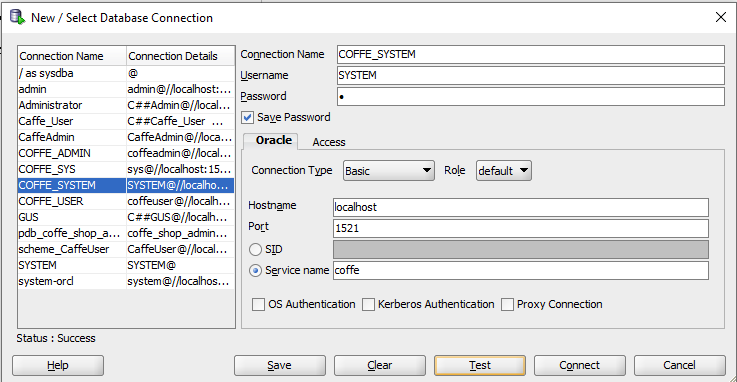


Рисунок 7.18 – Проверяем работоспособность базы данных

# **8 Руководство пользователя**

В данной курсовой работе функционально должны были выполнены следующие задачи:

* редактировать информацию о товарах;
* сотрудник создает и удаляет заказ;
* администратор получает справки о выручке за определенный период;
* администратор получает все заказы за определенный период;

Для того чтобы редактировать информацию о товарах с обычного пользователя и с администратора были разработаны следующие процедуры:

* DeleteItem([ID удаляемого блюда]) – удаление блюда;
* UpdateItemName([ID изменяемого блюда], [новое название блюда]) – обновление названия блюда;
* UpdateItemCost([ID изменяемого блюда], [обновленная цена]) – обновление стоимости блюда;

Для создания и удаления заказа сотрудником были разработаны следующие процедуры:

* CreateOrder([ID юзера, создающего заказ],[дата создания], [скидка], [идентификатор существования заказа], [ID блюда], [количество]) – создания заказа;
* DeleteOrder([ID удаляемого заказа]) – удаление заказа;

Для просмотра администратором выручки за определенный период была разработана процедура:

* GetSumPrice([дата начала периода],[ дата оканчания периода]);

Для получения администратором перечня заказов за определенный период была разработана следующая процедура:

* GetOrderByPeriod([дата начала периода],[ дата оканчания периода]);

Для воспроизведения этих функций в SQL Dqveloper нужно открыть окно Dbms Output и подключиться к нужному пользователю. Далее следует выделить анонимный блок с нужной нам функцией, предварительно вписав туда нужные параметры. Для запуска анонимного блока следует нажать клавишу F5 либо сочетание клавиш CTRL+Enter. В окно Dbms Output будет выведен результат работы анонимного блока.

Скрипты создания всех процедур представлены в приложении В.

# **Заключение**

В процессе решения поставленной задачи была достигнута поставленная цель по созданию базы данных «Coffe Shop» для автоматизированной системы, которая управляет рабочим процессом кофейни. В данной работе использовалось СУБД Oracle DataBase 12c. При разработке курсового проекта использовались объекты: таблицы, хранимые процедуры, индексы, последовательности, пакеты.

Основной целью курсового проекта стало проектирование базы данных, которая помогла облегчить процессы администрирования в кофейне.

Приложение прошло тестирование при использовании в БД большого количество данных. Также были реализованы процедуры для импорта, экспорта данных.

Была реализована технология «Резервное копирование и восстановление данных».

В соответствии с полученным результатом работы программы можно сделать вывод, что разработанная программа работает верно, а требования технического задания выполнены.

# **Cписок источников**

1. Документация Oracle [Электронный ресурс] / Foundation, Inc. https://docs.oracle.com/cd/B28359\_01/server.111/b31222/toc.htm
2. Официальный сайт Oracle [Электронный ресурс] / Foundation, Inc. https://www.oracle.com/database/database-vault/index.html
3. Продукты Oracle [Электронный ресурс] [Электронный ресурс] / Foundation, Inc. http://www.interface.ru/home.asp?artId=24678
4. Документы Oracle [Электронный ресурс] / Foundation, Inc. http://www.oracle.com/technetwork/database/security/database-vault-ds-12c-1898877.pdf

# **Приложение А**

//Создание таблиц

create table Item

(

IdItem int generated by default as identity start with 1

minvalue 1

maxvalue 1000000

increment by 1

cache 20

primary key,

Name varchar(50) not null,

Cost float not null CHECK (Cost >= 0),

IsDeleted CHAR(1) CHECK (IsDeleted IN ('1','0'))

);

create table UserInCaffe

(

IdUser int generated by default as identity start with 1

minvalue 1

maxvalue 1000000

increment by 1

cache 20

primary key,

Name varchar(50) not null,

Login varchar(50) not null,

Password varchar(50) not null,

Phone varchar(10) not null,

Created date not null,

DateOfBirth date not null,

Status varchar(50) default 'user' not null,

IsDeleted CHAR(1) CHECK (IsDeleted IN ('1','0'))

);

CREATE TABLE OrderInCaffe

(

IdOrder int generated by default as identity start with 1

minvalue 1

maxvalue 1000000

increment by 1

cache 20

primary key,

IdUser int,

Created date not null,

Discount float not null,

IsDeleted CHAR(1) CHECK (IsDeleted IN ('1','0')),

CONSTRAINT FK\_IdUser

FOREIGN KEY (IdUser)

REFERENCES UserInCaffe(IdUser)

);

create table ItemsInOrder

(

ItemsInOrder int generated by default as identity start with 1

minvalue 1

maxvalue 1000000

increment by 1

cache 20

primary key,

IdOrder int,

IdItem int,

Count NUMBER(10,2) NOT NULL CHECK (Count >= 0),

CONSTRAINT FK\_IdOrder

FOREIGN KEY (IdOrder)

REFERENCES OrderInCaffe(IdOrder),

CONSTRAINT FK\_IdItem

FOREIGN KEY (IdItem)

REFERENCES Item(IdItem)

);

create table StatusInCaffe

(

Status int primary key,

name varchar(50)

);

# **Приложение Б**

--создание пользователей и придание им необходимых прав

create public synonym Item for SYSTEM.Item;

create public synonym UserInCaffe for SYSTEM.UserInCaffe;

create public synonym StatusInCaffe for SYSTEM.StatusInCaffe;

create public synonym ItemsInOrder for SYSTEM.ItemsInOrder;

create public synonym OrderInCaffe for SYSTEM.OrderInCaffe;

create public synonym GetOrderByPeriod for SYSTEM.GetOrderByPeriod;

create public synonym GetSumPrice for SYSTEM.GetSumPrice;

create public synonym CreateOrder for SYSTEM.CreateOrder;

create public synonym DeleteOrder for SYSTEM.DeleteOrder;

create public synonym DeleteItem for SYSTEM.DeleteItem;

create public synonym UpdateItemName for SYSTEM.UpdateItemName;

create public synonym UpdateItemCost for SYSTEM.UpdateItemCost;

--создание табличных пространств

create tablespace CoffeTablespace

datafile 'D:\6sem\курсовые\бд\TABLESPACES\CoffeTablespace.dbf'

size 100M

autoextend on next 10M

MAXSIZE UNLIMITED;

create temporary tablespace TempCoffeTablespace

tempfile 'D:\6sem\курсовые\бд\TABLESPACES\TempCoffeTablespace.dbf'

size 100M

autoextend on next 10M

MAXSIZE UNLIMITED;

-- создание роли администратору

create role coffe\_admin\_role;

grant all privileges to coffe\_admin\_role;

-- создание обычного пользователя (работника)

create user coffeadmin identified by 1

default tablespace CoffeTablespace

quota unlimited on CoffeTablespace

temporary tablespace TempCoffeTablespace;

create user coffeuser identified by 1

default tablespace CoffeTablespace

quota unlimited on CoffeTablespace

temporary tablespace TempCoffeTablespace;

grant coffe\_user\_role to coffeuser;

-- выдача роли администратору

grant coffe\_admin\_role to coffeadmin;

-- создание роли пользователя

create role coffe\_user\_role;

grant connect to coffe\_user\_role;

-- выдача прав на запуск процедур и селект к таблицам пользователю

grant execute on CreateOrder to coffe\_user\_role;

grant execute on DeleteOrder to coffe\_user\_role;

grant execute on DeleteItem to coffe\_user\_role;

grant execute on UpdateItemName to coffe\_user\_role;

grant execute on UpdateItemCost to coffe\_user\_role;

grant select on Item to coffe\_user\_role;

grant select on ItemsInOrder to coffe\_user\_role;

grant select on OrderInCaffe to coffe\_user\_role;

# **Приложение В**

--Procedures

-- все заказы за определенный период

create or replace procedure GetOrderByPeriod (startdate date, enddate date)

as

cursor orders\_cur is select \* from OrderInCaffe where CREATED between startdate and enddate order by CREATED;

orderid OrderInCaffe.IDORDER%TYPE;

userid OrderInCaffe.IDUSER%TYPE;

crdt OrderInCaffe.CREATED%TYPE;

disc OrderInCaffe.DISCOUNT%TYPE;

isdel OrderInCaffe.ISDELETED%TYPE;

begin

open orders\_cur;

loop

fetch orders\_cur into orderid, userid, crdt, disc, isdel;

dbms\_output.put\_line(orderid || ' ' || userid || ' ' || crdt || ' ' || disc || ' ' || isdel);

exit when orders\_cur%NOTFOUND;

end loop;

close orders\_cur;

exception

when others then dbms\_output.put\_line('Ошибка : ' || sqlerrm);

end;

begin

GetOrderByPeriod(TO\_DATE('2022-06-01', 'YYYY-MM-DD'), TO\_DATE('2022-12-01', 'YYYY-MM-DD'));

end;

-- справка о выручке за определенный период

create or replace function GetSumPrice(startdate date, enddate date) return number

as

summa number;

begin

select SUM(COST\*COUNT) into summa from Item

inner join ItemsInOrder

on Item.IDITEM = ItemsInOrder.IDITEM

inner join OrderInCaffe on OrderInCaffe.IDORDER = ItemsInOrder.IDORDER

where OrderInCaffe.CREATED between startdate and enddate;

return summa;

end;

declare

totalsum number;

begin

totalsum := GetSumPrice(TO\_DATE('2022-08-01', 'YYYY-MM-DD'), TO\_DATE('2022-12-01', 'YYYY-MM-DD'));

dbms\_output.put\_line('Выручка за период: ' || to\_char(totalsum));

end;

-- создание заказа

create or replace procedure CreateOrder (userid OrderInCaffe.IDUSER%TYPE,

dtcreate OrderInCaffe.CREATED%TYPE,

disc OrderInCaffe.DISCOUNT%TYPE,

isdel OrderInCaffe.ISDELETED%TYPE,

itemid ItemsInOrder.IDITEM%TYPE,

qtty ItemsInOrder.COUNT%TYPE)

as

orderid number;

begin

insert into OrderInCaffe (IDUSER, CREATED, DISCOUNT, ISDELETED)

values (userid, dtcreate, disc, isdel)

returning IDORDER into orderid;

insert into ItemsInOrder (IDORDER, IDITEM, COUNT)

values (orderid, itemid, qtty);

dbms\_output.put\_line('Добавлен заказ');

exception

when others then dbms\_output.put\_line('Ошибка : ' || sqlerrm);

end;

begin

CreateOrder(1, TO\_DATE('2023-05-12', 'YYYY-MM-DD'), 0.2, 0, 1, 10);

end;

commit;

-------------------------------------------------------------

-- удаление заказа

create or replace procedure DeleteOrder(idtodelete OrderInCaffe.IDORDER%TYPE)

as

begin

update OrderInCaffe set ISDELETED = 1 where IDORDER = idtodelete;

dbms\_output.put\_line('Статус заказа изменен на УДАЛЕН');

exception

when others then dbms\_output.put\_line('Ошибка : ' || sqlerrm);

end;

begin

DeleteOrder(1008);

end;

commit;

----------------------------------------

-- удаление блюда

create or replace procedure DeleteItem(itemtodelete Item.IDITEM%TYPE)

as

begin

update Item set ISDELETED = 1 where IDITEM = itemtodelete;

dbms\_output.put\_line('Статус блюда изменен на УДАЛЕН из меню');

exception

when others then dbms\_output.put\_line('Ошибка : ' || sqlerrm);

end;

begin

DeleteItem(1);

end;

-- обновление названия блюда

create or replace procedure UpdateItemName(itemtoupdate Item.IDITEM%TYPE,

newname Item.NAME%TYPE)

as

oldname Item.NAME%TYPE;

begin

select NAME into oldname from Item where IDITEM = itemtoupdate;

update Item set NAME = newname where IDITEM = itemtoupdate;

dbms\_output.put\_line('Имя блюда блюда изменено с ' || oldname || ' на ' || newname);

exception

when others then dbms\_output.put\_line('Ошибка : ' || sqlerrm);

end;

begin

UpdateItemName(1, 'espresso');

end;

commit;

-- обновление стоимости блюда

create or replace procedure UpdateItemCost(itemtoupdate Item.IDITEM%TYPE,

newcost Item.COST%TYPE)

as

oldcost Item.COST%TYPE;

begin

select COST into oldcost from Item where IDITEM = itemtoupdate;

update Item set COST = newcost where IDITEM = itemtoupdate;

dbms\_output.put\_line('Стоимость блюда изменена с ' || oldcost || ' на ' || newcost);

exception

when others then dbms\_output.put\_line('Ошибка : ' || sqlerrm);

end;

begin

UpdateItemCost(1, 10.53);

end;

commit;

# **Приложение Г**

--XML импорт

drop DIRECTORY INFILEXML;

CREATE DIRECTORY INFILEXML AS 'D:\6sem\курсовые\бд\COFFE'

CREATE OR REPLACE FUNCTION ReadFileForImport(FileName NVARCHAR2)

RETURN XMLTYPE

AS

FileContent CLOB;

Buffer CLOB;

FileDescriptor UTL\_FILE.FILE\_TYPE;

BEGIN

FileDescriptor := UTL\_FILE.FOPEN('INFILEXML', FileName, 'R');

LOOP

BEGIN

UTL\_FILE.GET\_LINE(FileDescriptor, Buffer);

FileContent := FileContent || Buffer;

EXCEPTION WHEN NO\_DATA\_FOUND THEN EXIT;

END;

END LOOP;

RETURN XMLTYPE(FileContent);

END;

CREATE OR REPLACE PROCEDURE ImportXMLItems(FileName VARCHAR2)

AS

FileContent XMLTYPE;

Name NVARCHAR2(50);

Cost NUMBER(6,2);

IsDeleted CHAR(1);

i INT DEFAULT 1;

BEGIN

FileContent := ReadFileForImport(FileName);

LOOP

BEGIN

SELECT EXTRACTVALUE(FileContent, '/data/row['||i||']/NAME') INTO Name FROM DUAL;

SELECT EXTRACTVALUE(FileContent, '/data/row['||i||']/COST') INTO Cost FROM DUAL;

SELECT EXTRACTVALUE(FileContent, '/data/row['||i||']/ISDELETED') INTO IsDeleted FROM DUAL;

IF Name IS NULL THEN EXIT;

END IF;

insert into Item(Name, Cost, IsDeleted)

values (Name, Cost, IsDeleted);

i := i + 1;

END;

END LOOP;

exception

when others then dbms\_output.put\_line('Error');

END;

begin

ImportXMLItems('items.xml');

end;

select \* from Item;

delete item where Name like 'Item %';

----------------------

-- XML экспорт

DROP DIRECTORY EXSPORTFILEXML;

CREATE DIRECTORY EXSPORTFILEXML AS 'D:\6sem\курсовые\бд\COFFE';

create or replace procedure export\_items

is

rc sys\_refcursor;

doc DBMS\_XMLDOM.DOMDocument;

begin

open rc for select \* from Item;

doc := DBMS\_XMLDOM.NewDomDocument(XMLTYPE(rc));

DBMS\_XMLDOM.WRITETOFILE(doc, 'EXSPORTFILEXML/items\_export.xml');

end export\_items;

BEGIN

export\_items;

END;