

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

## **ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОЙ РАБОТЕ НА ТЕМУ:**

**«Реализация базы данных ресторана с применением технологии  
резервного копирования и восстановления»**

Выполнил студент Кантарович А.Д  
(Ф.И.О.)

Руководитель работы ст. преп. Нистюк О. А.  
(учен. степень, звание, должность, Ф.И.О., подпись)

И.о. зав. кафедрой ст. преп. Блинова Е.А.  
(учен. степень, звание, должность, Ф.И.О., подпись)

Курсовая работа защищена с оценкой \_\_\_\_\_

Минск 2024

## Содержание

Введение .....	4
1 Постановка задачи .....	5
1.1 Обзор аналогичных решений .....	5
1.1.1 Le Gosse .....	5
1.1.2 KFC .....	6
1.2 Требования к курсовому проекту .....	7
1.3 Вывод по разделу.....	8
2 Проектирование базы данных .....	9
3 Разработка объектов базы данных .....	11
3.1 Таблицы .....	11
3.2 Пакеты процедур .....	14
3.3 Триггеры.....	16
3.4 Планировщики .....	16
3.5 Синонимы.....	18
3.6 Роли.....	19
3.7 Табличные пространства .....	20
3.8 Вывод по разделу.....	21
4 Описание процедур импорта и экспорта.....	22
5 Тестирование производительности .....	23
6 Описание технологии и её применение в базе данных.....	25
7 Руководство пользователя. ....	26
7.1 Сторона системного администратора.....	26
7.2 Сторона администратора. ....	26
7.3 Сторона клиента .....	26
7.4 Вывод по разделу.....	27
Заключение.....	28
Список используемых источников .....	29
Приложение А.....	30
Приложение Б .....	31
Приложение В.....	35
Приложение Г .....	38

## **Введение**

В современном мире эффективное управление ресторанами становится все более актуальным. С увеличением конкуренции и разнообразия услуг, предоставляемых в сфере общественного питания, ресторанный бизнес требует не только качественного обслуживания, но и грамотного учета ресурсов, клиентов и блюд. В этом контексте создание базы данных для ресторана является важным шагом к оптимизации бизнес-процессов.

Данный курсовой проект направлен на разработку базы данных, которая позволит автоматизировать учет информации о меню, заказах, клиентах и поставщиках. Использование базы данных обеспечит быстрый доступ к необходимой информации, сократит время на обработку заказов и повысит качество обслуживания клиентов.

В ходе работы будут рассмотрены основные требования к проектируемой базе данных, а также технологии, которые будут использованы для ее создания. Особое внимание будет уделено структуре базы данных, ее функциональным возможностям и интерфейсу взаимодействия пользователей с системой.

Цель данного проекта — разработать эффективную и удобную в использовании базу данных для ресторана, которая станет основой для успешного управления ресторанным бизнесом.

# 1 Постановка задачи

## 1.1 Обзор аналогичных решений

Для определения требований к курсовому проекту стоит проанализировать схожие решения и аналоги. Ресторанный бизнес далеко не новый, он существовал ещё задолго до появления высокоразвитых технологий и тем более технологий баз данных. Однако, с появлением баз данных, ресторанный бизнес стал более простым в управлении и анализе. Сегодня сложно представить ресторанный бизнес без баз данных.

Рассмотрим два сайта ресторанов или сети ресторанов и попытаемся проанализировать их возможные структуры баз данных, определить основные таблицы и объекты.

### 1.1.1 Le Gosse

Ресторан французской кухни Le Gosse является идеальным примером для анализа его структуры, так как полностью совпадает с темой курсового проекта. Зайдя на веб-сайт legosse.by нас встречает несколько разделов. Самым главным и большим является раздел «Меню».

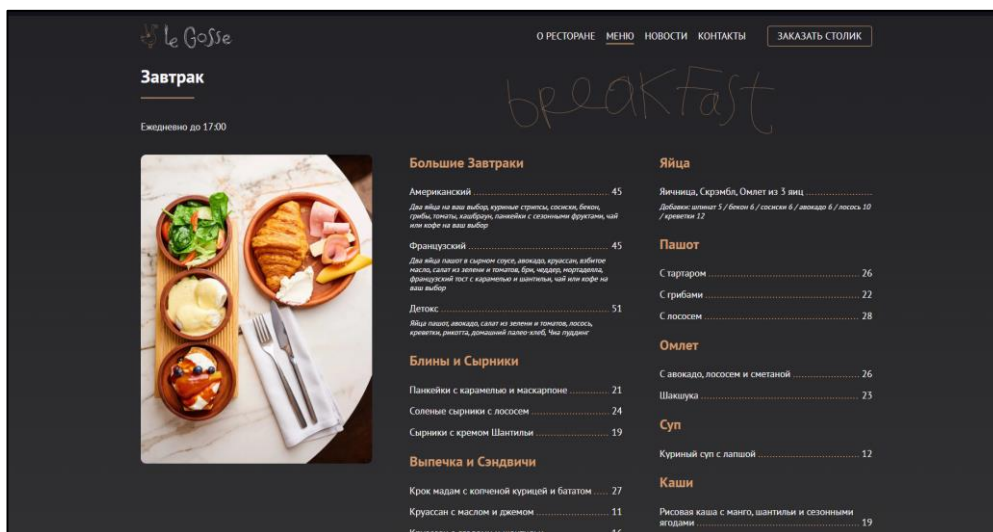


Рисунок 1.1 – Раздел меню на сайте legosse.by

Перейдя на него сразу можно увидеть категории блюд, что означает наличие таблицы категорий, в которой должны быть поля имени и описания. Развернув категорию можно увидеть список блюд с названием, описанием и ценой, что говорит нам о наличии таблицы позиций меню с соответствующими полями названия, описания и цены. В одной категории находится много блюд, что говорит о типе связи между таблицами, а именно “один к многим”.

В разделах есть кнопка «Заказать столик», которая позволяет зарезервировать стол.

Рисунок 1.2 – Форма бронирования

Для резервации нужно указать своё имя, телефон, дату бронирования. Этот факт указывает на то, что в базе данных реализовано как минимум ещё две таблицы: клиентов и бронирования. Таблица клиентов должна содержать поля имени и номера телефона. Таблица бронирований должна содержать клиента, который забронировал стол, стол, который данный клиент забронировал, и дата, на которую записана бронь. Раз в таблице бронирований есть стол, который забронировали, должна быть и таблица, которая будет хранить информацию о столах.

Разбор и анализ веб-сайта Le Gosse указали на существование уже пяти разных таблиц: клиентов, столов, бронирований, категорий меню и позиций меню. Эти таблицы являются необходимыми для корректной работы базы данных ресторана

### 1.1.2 KFC

Сеть ресторанов KFC является одним из мастодонтов ресторанного бизнеса в мире. Для управления такой сложной системой ресторанов база данных просто необходима. Рассмотрим сайт сети ресторанов KFC и попытаемся выявить новые структуры, которые могут быть в нашей базе данных.

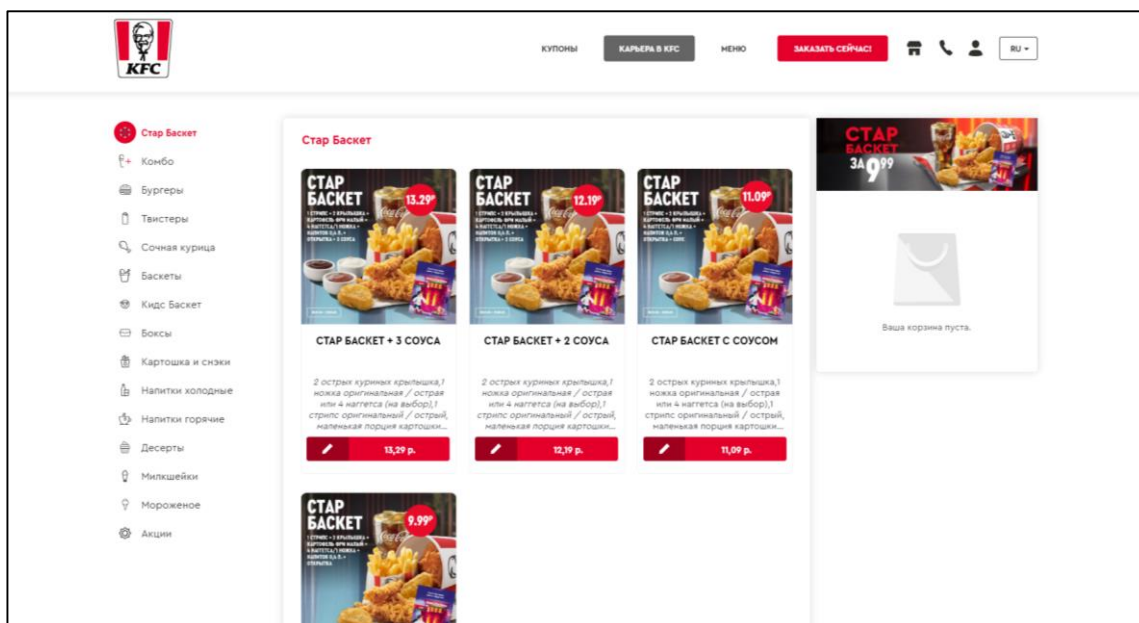


Рисунок 1.3 – Раздел меню на сайте [www.kfc.by](http://www.kfc.by)

Перейдя на сайт можно увидеть уже знакомые категории блюд и сами блюда. Среди разделов сайта можно увидеть опцию «Заказать сейчас!», которая позволяет сделать заказ. Заказы нужно где-то хранить, например, в таблице базы данных. Так же, как и позиции заказа.

Исследуя сайты с аналогичной базой данных было выявлено семь таблиц и несколько процедур к ним, такие, как процедуры добавления записей в таблицы, их изменение и удаление. Процедуры заказов и бронирования также необходимы в базе данных ресторана. Данный анализ будет необходим в дальнейшем проектировании.

## 1.2 Требования к курсовому проекту

Одним из ключевых аспектов успешного функционирования ресторанного бизнеса является эффективное управление данными. Создание базы данных, которая будет хранить информацию о меню, клиентах, заказах и бронировании, позволяет автоматизировать многие процессы и улучшить качество обслуживания. Важной частью данной системы является внедрение технологий резервного копирования и восстановления, что обеспечивает сохранность данных и их доступность в случае непредвиденных ситуаций.

На рынке существует множество систем управления ресторанами, каждая из которых предлагает различные функции, включая управление запасами, заказами и отчетностью. Однако многие из них не обеспечивают достаточного уровня защиты данных или не предлагают удобные механизмы резервного копирования. Это создаёт необходимость в разработке решения,

которое будет сочетать в себе все необходимые функции с акцентом на безопасность данных.

Должны быть выполнены следующие требования:

- база данных должна быть реализована в СУБД Oracle.
- доступ к данным должен осуществляться только через соответствующие процедуры;
- должен быть проведен импорт данных из JSON файлов, экспорт данных в формат JSON;
- 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.

Функционально должны быть реализованы следующие действия:

- управление меню;
- управление заказами;
- управление персоналом;
- резервирование стола;

Проект направлен на разработку архитектуры базы данных для ресторана, которая будет интуитивно понятной для пользователей. Также будет реализована система резервного копирования и восстановления данных, что обеспечит защиту от потери информации.

### **1.3 Вывод по разделу**

Исходя из обзора аналогов были составлены основные функциональные требования, к курсовому проекту. Рассмотренные аналоги предоставили полезные ориентиры для разработки базы данных, которая будет сочетать в себе простоту использования, производительность и широкую функциональность. Эти аспекты лягут в основу проектирования и реализации курсового проекта.

## 2 Проектирование базы данных

База данных состоит из пяти таблиц, взаимосвязанных внешними ключами, обеспечивающих эффективное хранение и структурирование данных. Диаграмма структуры полученной базы данных представлена на рисунке 2.1.

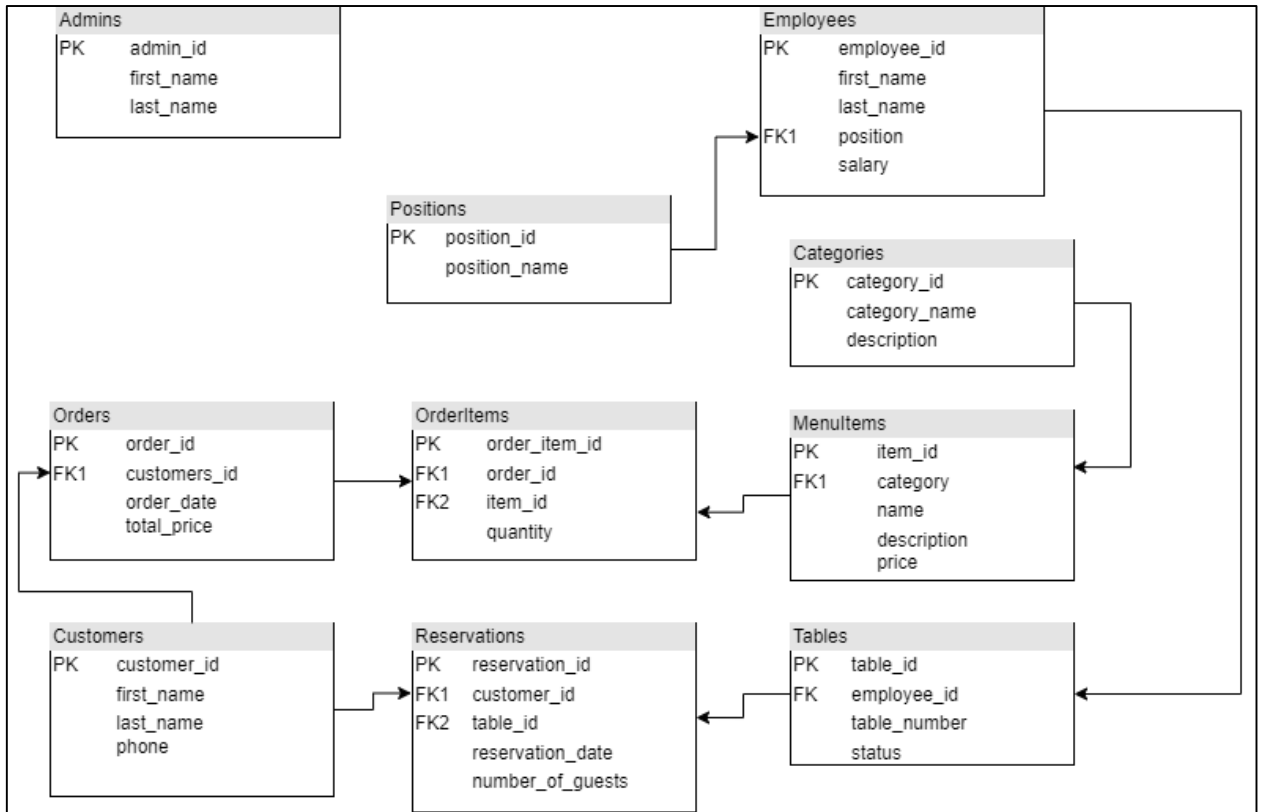


Рисунок 2.1 – UML диаграмма базы данных

Таблица Admins хранит в себе имена администраторов базы данных.

Таблица Positions хранит в себе все возможные должности работников, имя и идентификатор для связи с другими таблицами.

Таблица Employees отвечает за работников заведения, их зарплата и должность.

Таблица Orders содержит в себе информацию о заказе, о клиенте который его заказал, время и цену заказа.

Таблица OrderItems содержит в себе позиции заказа.

Таблица Categories содержит в себе категории меню.

Таблица MenuItemс содержит информацию о подаваемых блюдах: название, описание, цена.

Таблица Customers содержит информацию о клиенте, его имя, фамилию, номер телефона.



Таблица Reservations содержит информацию о брони стола, имеется идентификатор забронированного стола и идентификатор клиента что забронировал и дата брони.

Таблица Tables содержит информацию о столике: статус бронирования, номер стола и официанта, который отвечает за столик.

Для базы данных была создана диаграмма вариантов использования для пользователя, администратора и главного (системного) администратора. Диаграмма представлена в приложении А.

## 3 Разработка объектов базы данных

### 3.1 Таблицы

Таблицы являются неотъемлемой частью любой реляционной базы данных. Краткая характеристика каждой из таблиц была предоставлена в разделе 2. В этом разделе рассмотрим их подробнее.

Таблица клиентов, Customers\_Table, содержит в себе поля идентификатора клиента, имя, фамилия, номер телефона, количество заказов, стоимость проданных этому клиенту блюд, маркер vip клиента.

Таблица 3.1 – Структура таблицы Customers\_Table

Столбец	Тип данных	Ограничение целостности
customer_id	NUMBER	PRIMARY KEY
first_name	VARCHAR2(50)	NOT NULL
last_name	VARCHAR2(50)	NOT NULL
phone	VARCHAR2(20)	NOT NULL
order_amount	NUMBER	DEFAULT 0
order_price	NUMBER	DEFAULT 0
vip	NUMBER(1)	DEFAULT 0 CHECK (vip IN (0, 1))

Поля количества заказов и стоимости проданных блюд влияют на то, идентифицируется ли клиент как vip клиент или нет.

Таблица администраторов базы данных, Admins\_Table, содержит идентификатор администратора, его имя и фамилию.

Таблица 3.2 – Структура таблицы Admins\_Table

Столбец	Тип данных	Ограничение целостности
admin_id	NUMBER	PRIMARY KEY
first_name	VARCHAR2(50)	NOT NULL
last_name	VARCHAR2(50)	NOT NULL

Таблица категорий, Category\_Table, содержит поля идентификатора категории, имени категории и описания категории.

Таблица 3.3 – Структура таблицы Category\_Table

Столбец	Тип данных	Ограничение целостности
category_id	NUMBER	PRIMARY KEY
category_name	VARCHAR2(50)	NOT NULL UNIQUE
category_description	VARCHAR2(255)	

Таблица позиций меню, MenuItems\_Table, содержит поле идентификатора позиции, названия блюда, описания, цены, идентификатора категории к которому принадлежит блюдо, статистика продаж за день, месяц, год и за всё время.

Таблица 3.4 – Структура таблицы MenuItems\_Table

Столбец	Тип данных	Ограничение целостности
item_id	NUMBER	PRIMARY KEY
name	VARCHAR2(100)	NOT NULL
description	VARCHAR2(255)	
price	NUMBER(10, 2)	NOT NULL
category_id	NUMBER	FOREIGN KEY
daily_buy	NUMBER	DEFAULT 0 CHECK (daily_buy >= 0)
monthly_buy	NUMBER	DEFAULT 0 CHECK (monthly_buy >= 0)
yearly_buy	NUMBER	DEFAULT 0 CHECK (yearly_buy >= 0)
total_buy	NUMBER	DEFAULT 0 CHECK (total_buy >= 0)

Category\_Table и MenuItems\_Table связаны между собой связью “один к многим”, так как в одной категории может быть много позиций меню.

Таблица заказов, Orders\_Table, содержит в себе идентификатор заказа, идентификатор сделавшего заказ пользователя, время заказа, общая цена заказа и количество блюд в заказе.

Таблица 3.5 – Структура таблицы Orders\_Table

Столбец	Тип данных	Ограничение целостности
order_id	NUMBER	PRIMARY KEY
customer_id	NUMBER	FOREIGN KEY
order_date	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP
total_price	NUMBER	DEFAULT 0
item_quantity	NUMBER	DEFAULT 0

Orders\_Table связана с Customers\_Table связью “один к многим”, так как один пользователь может сделать много заказов.

Таблица позиций заказа, OrderItems\_Table, содержит в себе идентификатор позиции, идентификатор заказа, идентификатор блюда, количество порций, общая цена позиции заказа.

Таблица 3.6 – Структура таблицы OrderItems\_Table

Столбец	Тип данных	Ограничение целостности
order_item_id	NUMBER	PRIMARY KEY
order_id	NUMBER	FOREIGN KEY
item_id	NUMBER	FOREIGN KEY
quantity	NUMBER	NOT NULL
pos_price	NUMBER	NOT NULL

OrderItems\_Table связана с Orders\_Table связью “один к многим”, так как у заказа есть несколько позиций заказа, а MenuItems\_Table связана с

OrderItems\_Table связью “один к одному”, так как на одной позиции заказа может быть только одна позиция меню.

Таблица должностей, Positions\_Table, содержит поля идентификатора должности и названия должности.

Таблица 3.7 – Структура таблицы Positions\_Table

Столбец	Тип данных	Ограничение целостности
position_id	NUMBER	PRIMARY KEY
position_name	VARCHAR2(50)	NOT NULL UNIQUE

Таблица сотрудников, Employees\_Table, имеет поля идентификатора сотрудника, имени, фамилии, идентификатора должности, зарплаты.

Таблица 3.8 – Структура таблицы Employees\_Table

Столбец	Тип данных	Ограничение целостности
employee_id	NUMBER	PRIMARY KEY
first_name	VARCHAR2(50)	NOT NULL
last_name	VARCHAR2(50)	NOT NULL
position_id	NUMBER	FOREIGN KEY
salary	NUMBER	NOT NULL

Employees\_Table связана с Positions\_Table связью “один к многим”, так как одна должность может быть у неопределенного числа сотрудников.

Таблица столов, Tables\_Table, содержит идентификатор стола, вместимость, идентификатор официанта, статус бронирования.

Таблица 3.9 – Структура таблицы Tables\_Table

Столбец	Тип данных	Ограничение целостности
table_id	NUMBER	PRIMARY KEY
capacity	NUMBER	NOT NULL
waiter	NUMBER	FOREIGN KEY
status	VARCHAR2(20)	DEFAULT 'Available'

Таблица бронирований, Reservations\_Table, содержит в себе идентификатор брони, идентификатор клиента, стола, дата бронирования

Таблица 3.10 – Структура таблицы Reservations\_Table

Столбец	Тип данных	Ограничение целостности
reservation_id	NUMBER	PRIMARY KEY
customer_id	NUMBER	FOREIGN KEY
table_id	NUMBER	FOREIGN KEY
reservation_date	TIMESTAMP	NOT NULL

Reservations\_Table связана с Tables\_Table связью “один к одному” так как на один стол можно навесить только одну бронь, а с Customers\_Table Reservations\_Table связана типом “один к многим”, так как клиент может забронировать несколько столов.

Представленная структура базы данных обеспечивает эффективное и надежное хранение информации, необходимой для функционирования системы управления кинотеатром. Предусмотренные ограничения целостности и связи между таблицами гарантируют целостность данных и согласованность информации в системе.

### 3.2 Пакеты процедур

Для упрощения и структурирования процессов управления данными были созданы три пакета процедур: admin\_package, user\_package и sys\_admin\_package. Каждый из этих пакетов выполняет функции определенной роли. Пакеты направлены на улучшение управляемости, безопасности и удобства работы с базой данных.

Пакет admin\_package содержит в себе процедуры администратора для модерирования основных таблиц базы данных. Процедуры пакета admin\_package представлены в таблице 3.11.

Таблица 3.11 – Процедуры пакета admin\_package

Процедура	Назначение
AddCustomer	Добавление клиента в таблицу Customers_Table
DeleteCustomer	Удаление клиента из таблицы
UpdateCustomer	Изменение записи о клиенте
AddPosition	Добавление должности в Positions_Table
UpdatePosition	Изменение записи о должности
DeletePosition	Удаление записи о должности
AddEmployee	Добавление сотрудника в Employees_Table
DeleteEmployee	Удаление сотрудника из таблицы
UpdateEmployee	Изменение записи о сотруднике
AddTable	Добавление стола в Tables_Table
UpdateTable	Изменение записи о столе
DeleteTable	Удаление записи о столе
AssignWaiterToTable	Привязать официанта к столу
ClearWaiterFromTable	Отвязать официанта от стола
VIPCustomersPrice	Вывод vip клиентов по стоимости заказов
VIPCustomersOrderItems	Вывод vip клиентов по количеству купленных блюд
DailyBuyStatistics	Выводит статистику по продажам блюд за день
MonthlyBuyStatistics	Выводит статистику по продажам блюд за месяц
YearlyBuyStatistics	Выводит статистику по продажам блюд за год

Продолжение таблицы 3.11

Процедура	Назначение
TotalBuyStatistics	Выводит статистику по продажам блюд за всё время

Пакет `user_package` содержит процедуры клиента, направленные на совершение заказов, их изменение и отмену. Так же в пакете описаны процедуры бронирования стола и отмены бронирования. Процедуры пакета `user_package` представлены в таблице 3.12.

Таблица 3.12 – Процедуры пакета `user_package`

Процедура	Назначение
AddOrder	Сделать заказ. Заказ заносится в <code>Orders_Table</code>
CancelOrder	Отменить заказ. Заказ и его позиции удаляются из таблиц
AddItemToOrder	Добавить позицию заказа. Добавляет позицию в <code>OrderItems_Table</code>
DeleteItemFromOrder	Удалить позицию заказа.
CreateReservation	Забронировать столик. Заносит запись о бронировании в <code>Reservations_Table</code>
DeleteReservation	Отмена брони.

Пакет `sys_admin_package` содержит процедуры для работы с таблицей админов. В нем определены процедуры для добавления администратора базы данных, изменения информации о нем и удаления записи о нем. Главный администратор также может экспортировать таблицу в JSON формат и импортировать из него. Процедуры, определенные в пакете `sys_admin_package`, представлены в таблице 3.13.

Таблица 3.13 – Процедуры пакета `sys_admin_package`

Процедура	Назначение
AddAdmin	Добавляет администратора в таблицу <code>Admins_Table</code>
DeleteAdmin	Удаляет администратора из таблицы по его <code>id</code>
UpdateAdmin	Изменяет существующую запись в таблице <code>Admins_Table</code>
ExportAdminsJSON	Экспортирует таблицу админов в JSON файл
ImportAdminsJSON	Импортирует данные по админам из JSON файла

Таким образом в базе данных ресторана Гюсто создано три пакета под три роли, что помогает разделить задачи между этими ролями и обезопасить доступ к данным. Спецификации пакетов представлены в приложении Б.

### 3.3 Триггеры

Триггеры являются очень удобным и полезным инструментом БД, которые позволяют облегчить и частично автоматизировать управление БД. В базе данных ресторана французской кухни Гюсто создан триггер, который реагирует, когда у одного из клиентов в таблице Customers\_Table значение заказанных блюд достигнет значения 100, или цена всех заказанных блюд примет значение 1000. Код, создающий триггер для присвоения клиенту vip статуса, представлен в листинге 3.14.

```
CREATE OR REPLACE TRIGGER trg_update_vip
BEFORE UPDATE ON Customers
FOR EACH ROW
BEGIN
    IF :NEW.orders_price >= 1000 OR :NEW.order_amount >= 100
THEN
        :NEW.vip := 1;
    END IF;
END;
```

Листинг 3.14 – Код создания триггера trg\_update\_vip

Как только триггер сработает, он установит значение vip у пользователя на 1, помечая его как важного клиента.

В базе данных ресторана также создан триггер, который устанавливает значение vip у клиента на 0, что снимает с него статус важного для ресторана клиента. Триггер, отнимающий vip статус клиента, представлен в листинге 3.15.

```
CREATE OR REPLACE TRIGGER trg_update_novip
BEFORE UPDATE ON Customers
FOR EACH ROW
BEGIN
    IF :NEW.orders_price < 1000 AND :NEW.order_amount < 100 THEN
        :NEW.vip := 0;
    END IF;
END;
```

Листинг 3.15 – Код создания trg\_update\_novip

Данные триггеры очень сильно помогают автоматизировать процесс выдачи vip статуса клиентам.

### 3.4 Планировщики

Планировщик — это мощный компонент Oracle Database, предназначенный для автоматизации выполнения задач и управления

расписанием этих задач. В базе данных GUSTO\_DB планировщики работают с таблицами MenuItems\_Table, Orders\_Table и Reservations\_Table.

Планировщики таблицы MenuItems\_Table сбрасывают статистику заказов блюд раз в день, месяц или год. Планировщик по сбросу статистики за день представлен в листинге 3.16.

```
BEGIN
  DBMS_SCHEDULER.create_job (
    job_name          => 'reset_daily_buy',
    job_type          => 'PLSQL_BLOCK',
    job_action        => 'BEGIN Clear_Day; END;',
    start_date        => SYSTIMESTAMP,
    repeat_interval   => 'FREQ=DAILY; BYHOUR=0; BYMINUTE=0;
BYSECOND=0',
    enabled           => TRUE
  );
END;
```

Листинг 3.16 – Создание планировщика reset\_daily\_buy

Планировщик таблицы Orders\_Table очищает таблицу от неактуальных заказов раз в день. Код создания планировщика clear\_orders\_sched представлен в листинге 3.17.

```
DBMS_SCHEDULER.create_job (
  job_name          => 'clear_orders_sched',
  job_type          => 'PLSQL_BLOCK',
  job_action        => 'BEGIN Clear_Orders; END;',
  start_date        => SYSTIMESTAMP,
  repeat_interval   => 'FREQ=DAILY; BYHOUR=5; BYMINUTE=0;
BYSECOND=0',
  enabled           => TRUE
);
```

Листинг 3.17 – Создание планировщика clear\_orders\_sched

Планировщик таблицы Reservations\_Table очищает таблицу от неактуальных бронирований, чей срок превышает два дня. Код создания планировщика по сбросу брони представлен в листинге 3.18.

```
DBMS_SCHEDULER.create_job (
  job_name          => 'clear_old_reservs',
  job_type          => 'PLSQL_BLOCK',
  job_action        => 'BEGIN Clear_Old_Reservation; END;',
  start_date        => SYSTIMESTAMP,
```



```
repeat_interval => 'FREQ=DAILY; BYHOUR=5; BYMINUTE=0;
BYSECOND=0',
enabled         => TRUE
);
```

### Листинг 3.18 – Создание планировщика clear\_old\_reservs

Таким образом в базе данных Gusto планировщики созданы для очистки таблиц от неактуальной и устаревшей информации. Процедуры, выполняющиеся планировщиками, представлены в листингах в приложении В.

## 3.5 Синонимы

Создание синонима в базе данных Oracle выполняется с целью предоставления альтернативного и удобного имени для объекта базы данных. Синоним может использоваться для скрывтия сложных или длинных имен объектов, что повышает читаемость кода и облегчает его поддержку. В базе данных ресторана Gusto созданы синонимы таблиц для удобного их использования как администраторам, так и пользователям. Синонимы таблиц представлены в листинге 3.19.

```
CREATE PUBLIC SYNONYM Customers FOR
ANDREW_KANTAROVICH.Customers_Table;
CREATE PUBLIC SYNONYM Admins FOR
ANDREW_KANTAROVICH.Admins_Table;
CREATE PUBLIC SYNONYM Category FOR
ANDREW_KANTAROVICH.Category_Table;
CREATE PUBLIC SYNONYM MenuItems FOR
ANDREW_KANTAROVICH.MenuItems_Table;
CREATE PUBLIC SYNONYM Orders FOR
ANDREW_KANTAROVICH.Orders_Table;
CREATE PUBLIC SYNONYM OrderItems FOR
ANDREW_KANTAROVICH.OrderItems_Table;
CREATE PUBLIC SYNONYM Positions FOR
ANDREW_KANTAROVICH.Positions_Table;
CREATE PUBLIC SYNONYM Employees FOR
ANDREW_KANTAROVICH.Employees_Table;
CREATE PUBLIC SYNONYM Tables FOR
ANDREW_KANTAROVICH.Tables_Table;
CREATE PUBLIC SYNONYM Reservations FOR
ANDREW_KANTAROVICH.Reservations_Table;
```

### Листинг 3.19 – Создание синонимов

Также синонимы созданы для админского, пользовательского пакетов для укорочения названия пакета при вызове процедур пакета. Синонимы для пакетов представлены в листинге 3.20.

```
create public synonym user_action for
ANDREW_KANTAROVICH.user_package;

create public synonym admin_action for
ANDREW_KANTAROVICH.admin_package;
```

Листинг 3.20 – Синонимы пакетов процедур

Использование синонимов способствует более гибкому взаимодействию с объектами базы данных, а также улучшает общую структуру базы данных.

### 3.6 Роли

Для определения функциональных возможностей пользователей были создана роль `user_role`. Данная роль назначается каждому зарегистрированному пользователю и содержит права на создание сессии и выполнения `user_package`. Код создания роли и выдачи привилегий представлен в листинге 3.21.

```
create role gusto_user_role;
GRANT CREATE SESSION TO gusto_user_role;
GRANT SELECT, INSERT, UPDATE, DELETE ON ORDERS TO
gusto_user_role;
GRANT SELECT, INSERT, UPDATE, DELETE ON ORDERITEMS TO
gusto_user_role;
GRANT SELECT, UPDATE ON CUSTOMERS TO gusto_user_role;
GRANT SELECT, UPDATE ON MENUITEMS TO gusto_user_role;
GRANT SELECT on TABLES to gusto_user_role;
GRANT SELECT on RESERVATIONS to gusto_user_role;
GRANT EXECUTE ON ANDREW_KANTAROVICH.user_package TO
gusto_user_role;
```

Листинг 3.21 – Создание роли `gusto_user_role`

Для определения функциональных возможностей администратора был создан пользователь базы данных `admin_database` и роль `admin role`. При назначении администратору роли, он получает расширенные привилегии, такие как возможность создания процедур, пользователей, сессий и таблиц, а также высокие административные права, включая DBA. Роль `admin_role` группирует эти привилегии, обеспечивая централизованное управление доступом к базе данных. Код создания роли админа представлен в листинге 3.22.

```
create role gusto_admin_role;  
GRANT DBA TO gusto_admin_role;  
grant execute on admin_package to gusto_admin_role;
```

### Листинг 3.22 – Создание роли gusto\_admin\_role

Такая структура ролей и их назначение пользователю спроектированы для обеспечения эффективного и безопасного управления базой данных, предоставляя администратору только необходимые привилегии для выполнения своих обязанностей без предоставления излишних прав, которые могли бы представлять потенциальные угрозы безопасности.

## 3.7 Табличные пространства

Табличные пространства в базах данных играют ключевую роль в управлении физическим пространством хранения данных. Они позволяют группировать объекты базы данных, такие как таблицы и индексы, в логические структуры, что упрощает администрирование и управление. Табличные пространства дают возможность контролировать, где хранятся данные, распределяя их по различным физическим устройствам, что может улучшать производительность и упрощать резервное копирование. Разделение данных по табличным пространствам уменьшает конкуренцию за ресурсы, повышая производительность при одновременной работе с несколькими пользователями.

В базе данных ресторана Гюсто создано два табличных пространства. Первое, Restaurant\_Data, хранит таблицы, непосредственно относящиеся к данным, которые заполняют администраторы. Код создания табличного пространства Restaurant\_Data представлен в листинге 3.23.

```
CREATE TABLESPACE RESTAURANT_DATA  
DATAFILE  
'/opt/oracle/oradata/ORCLCDB/GUSTO_PDB/restaurant_data.dbf'  
SIZE 100M  
AUTOEXTEND ON  
NEXT 10M  
MAXSIZE UNLIMITED  
LOGGING  
EXTENT MANAGEMENT LOCAL;
```

### Листинг 3.23 – Создание Restaurant\_Data

Табличное пространство Restaurant\_Data хранит данные, необходимые для корректной работы базы данных, такие, как категории меню, позиции меню, данные о сотрудниках, столах и т.д.

Табличное пространство, отвечающее за данные добавленные пользователем, создано под именем User\_Data. Код создания табличного пространства User\_Data представлен в листинге 3.24.

```
CREATE TABLESPACE RESTAURANT_DATA
DATAFILE
'/opt/oracle/oradata/ORCLCDB/GUSTO_PDB/user_data.dbf'
SIZE 100M
AUTOEXTEND ON
NEXT 10M
MAXSIZE UNLIMITED
LOGGING
EXTENT MANAGEMENT LOCAL;
```

### Листинг 3.24 – Создание User\_Data

Табличное пространство User\_Data хранит в себе таблицы заказов, позиций заказов и таблицы бронирования, то есть таблицы, непосредственно относящиеся к функционалу со стороны клиента.

## 3.8 Вывод по разделу

Все рассмотренные компоненты формируют сложную, но гибкую и эффективную систему, которая обеспечивает корректное функционирование и управление данными. Благодаря тщательно спроектированным объектам базы данных система способна гарантировать надежность, оперативность и целостность данных.

Следует подчеркнуть, что объекты базы данных разработаны с учетом разных ролей пользователей, таких как обычный пользователь, администратор и главный администратор. Каждому типу пользователя предоставлены соответствующие права для выполнения своих задач в системе.

Система охватывает не только основные операции управления данными, но и предлагает дополнительные функции, такие как анализ пользовательской активности и выдача рекомендаций.

Таким образом, разработанные объекты базы данных служат не только для хранения данных, но и обеспечивают функциональность, способствующую эффективному управлению и взаимодействию с системой.

## 4 Описание процедур импорта и экспорта

В данном проекте реализованы процедуры импорта и экспорта данных в формате JSON для таблицы Admins\_Table. Формат JSON обеспечивает удобочитаемое и структурированное представление информации, что облегчает обмен данными между различными системами и платформами. Кроме того, использование JSON способствует улучшению резервного копирования и миграции данных, обеспечивая надежность и сохранность информации.

Для реализации экспорта данных в JSON, была разработана процедура ExportAdminsJSON, в свою очередь, для импорта данных была разработана процедура ImportAdminsJSON. Процедура экспорта таблицы админов в формат JSON представлена в листинге 4.1.

```
PROCEDURE ExportAdminsJSON AS
    l_file UTL_FILE.FILE_TYPE;
    l_json_data CLOB;
    v_ErrorMessage VARCHAR2(4000);
BEGIN
    l_file := UTL_FILE.FOPEN('JSON_DIR', 'customers.json',
    'w', 32767);
    FOR rec IN (SELECT admin_id, first_name, last_name FROM
Admins) LOOP
        l_json_data := '{"admin_id": ' || rec.admin_id || ',
"first_name": "' || rec.first_name || "', "last_name": "' ||
rec.last_name || '"}';
        UTL_FILE.PUT_LINE(l_file, l_json_data);
    END LOOP;
    UTL_FILE.FCLOSE(l_file);
    DBMS_OUTPUT.PUT_LINE('Экспорт      таблицы      завершен
успешно. ');
    EXCEPTION
        WHEN OTHERS THEN
            v_ErrorMessage := SQLERRM;
            DBMS_OUTPUT.PUT_LINE('Ошибка      при      экспорте' ||
sqlerrm);
            IF      UTL_FILE.IS_OPEN(l_file)      THEN
                UTL_FILE.FCLOSE(l_file);
            END IF;
END;
```

Листинг 4.1 – Процедура ExportAdminsJSON

JSON файл с записями из таблицы админов хранится в директории базы данных. Чтобы импортировать данные используется процедура ImportAdminsJson, представленная в приложении Г.

## 5 Тестирование производительности

Одной из основных задач при разработке является тестирование производительности базы данных. Это тестирование помогает определить, насколько оперативно база данных обрабатывает запросы и возвращает результаты. Оценка производительности критична для понимания эффективности системы и выявления потенциальных недочетов, которые могут приводить к задержкам или проблемам при использовании приложения.

Для тестирования производительности была выбрана таблица клиентов (Customers\_Table). Таблица клиентов была выбрана для тестирования так как имеет связи с другими неотъемлемыми для базы данных ресторана таблицами, такими как Orders\_Table и Reservation\_Table. На таблицу клиентов так же поступает много запросов со стороны процедур пакета клиента. Поэтому быстроту работы базы данных стоит протестировать именно на этой таблице.

Для проверки производительности в таблицу клиентов, с помощью цикла for, было занесено сто тысяч строк. Цикл ввода сто тысяч строк приведен в листинге 5.1.

```
begin
    FOR i IN 1..100000 LOOP
        INSERT INTO Customers_Table (first_name, last_name,
phone) VALUES ('Имя', 'Фамилия', 375331000000 + i);
        IF MOD(i, 1000) = 0 THEN COMMIT; END IF;
    END LOOP;
    COMMIT;
end;
```

Листинг 5.1 – Цикл добавления строк в Customers\_Table

В ходе тестирования производительности путем выполнения процедур для работы с таблицей клиентов выяснилось, что скорость выполнения процедур достаточно высока для комфортной и быстрой работы, не превышает ста миллисекунд. Достигается это тем, что все процедуры происходят с поиском записей по первичным ключам. А как известно, индексы для первичных ключей создаются автоматически, делая работу с ними намного быстрее. Тестирование производительности на поиск строки представлен на рисунке 5.2.

select first_name, last_name from Customers where customer_id = '85000';		
Script Output x Query Result x Explain Plan x		
SQL 0,041 seconds		
OPERATION	OBJECT_NAME	OPTIONS
SELECT STATEMENT		
TABLE ACCESS	CUSTOMERS_TABLE	BY INDEX ROWID
INDEX	SYS_C008225	UNIQUE SCAN
Access Predicates CUSTOMER_ID=85000		

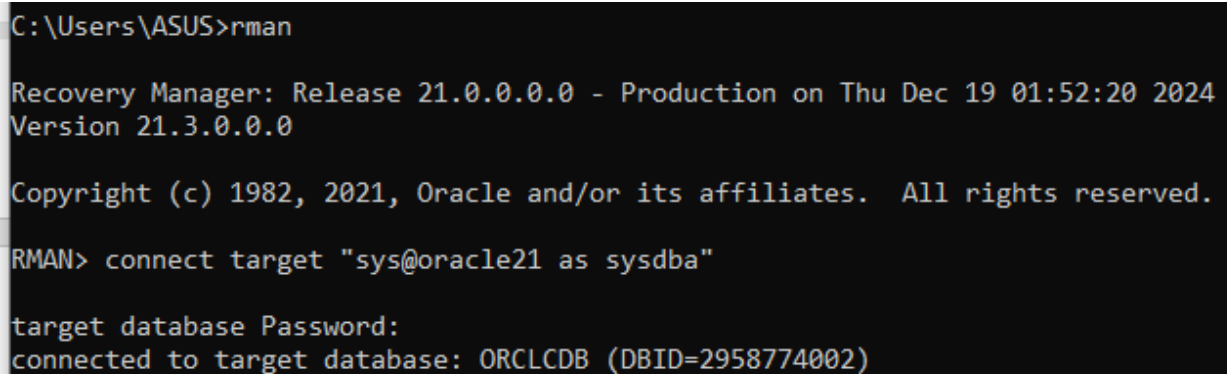
Рисунок 5.2 – Результат тестирования

В ходе тестирования производительности выяснилось, что база данных имеет хорошую скорость выполнения процедур и запросов и выполнения процедур в связи с тем, что все операции выполняются, основываясь на полях с первичным ключом, к которому уже прикреплен индекс.

## 6 Описание технологии и её применение в базе данных

В базе данных ресторана Гюсто была использована технология резервного копирования и восстановления. Резервное копирование и восстановление в базах данных представляет собой систему методов и инструментов, предназначенных для сохранения и восстановления данных в случае их потери или повреждения. В данной базе данных используется технология резервного копирования для обеспечения надежности и безопасности хранения информации.

Для работы с резервными копиями и их восстановлением была выбрана утилита RMAN. RMAN (Recovery Manager) – это утилита для управления резервным копированием и восстановлением баз данных Oracle. Она предоставляет мощные средства для защиты данных и упрощает процессы резервного копирования. Процесс подключения к базе данных с помощью утилиты RMAN представлена на рисунке 6.1.



```
C:\Users\ASUS>rman

Recovery Manager: Release 21.0.0.0.0 - Production on Thu Dec 19 01:52:20 2024
Version 21.3.0.0.0

Copyright (c) 1982, 2021, Oracle and/or its affiliates. All rights reserved.

RMAN> connect target "sys@oracle21 as sysdba"

target database Password:
connected to target database: ORCLCDB (DBID=2958774002)
```

Рисунок 6.1 – Подключение RMAN к базе данных

Технология резервного копирования позволяет обеспечить сохранность и доступность данных, играя ключевую роль в обеспечении непрерывности работы базы данных и предоставлении средств для восстановления информации в случае необходимости.



## **7 Руководство пользователя.**

В базе данных реализовано 3 роли пользователей: системный администратор (главный администратор), администратор и клиент.

### **7.1 Сторона системного администратора.**

В задачи системного администратора входит управление базой данных, создание ролей и присвоение им привилегий. Резервное копирование и восстановление также происходит со стороны системного администратора. Системный администратор создает таблицу Admins\_Table, в которую с помощью процедур пакета sys\_admin\_package добавляет, изменяет и удаляет информацию об администраторах базы данных.

### **7.2 Сторона администратора.**

В задачи администратора базы данных входит основное управление таблицами, их создание и наполнение. От имени администратора создаются триггеры, планировщики и другие объекты базы данных необходимые для её правильной работы.

С помощью таких процедур из пакета admin\_package, как AddCustomer, AddCategory, AddMenuItem, AddPosition, AddEmployee, AddTable он заполняет соответствующие таблицы информацией. С помощью процедур UpdateCustomer, UpdateCategory, UpdateMenuItem, UpdatePosition, UpdateEmployee, UpdateTable он может изменять записи в этих таблицах. А с помощью процедур DeleteCustomer, DeleteCategory, DeleteMenuItem, DeletePosition, DeleteEmployee, DeleteTable – удалять неактуальные записи с таблиц.

Также администратор отвечает за назначение официантов к столам с помощью процедур AssignWaiterToTable для привязки, и ClearWaiterFromTable для отвязки от стола. Процедуры ExportAdminsJSON и ImportAdminsJSON для экспорта и импорта таблицы админов в JSON формат тоже доступны только администратору в пакете admin\_package.

### **7.3 Сторона клиента**

В пакете клиентских процедур входят процедуры работы с заказами и бронированием столов. Все процедуры для удобной работы со стороны клиента реализованы в пакете user\_package.

Для работы с заказами реализованы процедуры AddOrder, CancelOrder, AddItemToOrder и DeleteItemFromOrder. Первые две процедуры работают непосредственно с заказом, с его регистрацией и отменой. Процедуры AddItemToOrder и DeleteItemfromOrder работают с позициями заказа, позволяют добавить блюдо к заказу или убрать его. При добавлении блюд к

заказу значение общей стоимости заказа увеличивается, также увеличивая значение стоимости заказанных блюд и значение количества заказанных блюд в записи клиента.

Для работы с бронированием стола реализованы процедуры CreateReservation и CancelReservation, которые регистрируют и отменяют бронирование соответственно. Клиент выбирает дату бронирования и стол, который он хочет забронировать, что меняет значение status у записи стола на Taken. Если статус стола уже является Taken, бронирование невозможно. Отмена брони удаляет запись о бронировании из таблицы и возвращает значение status у стола на Available.

#### **7.4 Вывод по разделу**

Разделение по ролям в базе данных ресторана Гюсто помогает четко разделить задачи между пользователями, будь это клиент или админ. Разделение по ролям помогает ограничить клиента от действий и процедур, которые могут помешать корректной работе базы данных.

## Заключение

В заключении курсового проекта, посвященного созданию базы данных французского ресторана с использованием СУБД Oracle и технологий резервного копирования и восстановления, можно отметить, что проект успешно достиг своих целей. Разработка базы данных включала проектирование структуры данных, создание таблиц и установление связей между ними, что обеспечило эффективное хранение и управление информацией о меню, клиентах и заказах. Использование возможностей Oracle, таких как табличные пространства, позволило оптимизировать хранение данных и обеспечить их безопасность.

Технологии резервного копирования и восстановления, внедренные в проект, обеспечивают надежность и защиту данных от потерь, что является критически важным для работы любого бизнеса. Проведение тестирования процессов резервного копирования подтвердило их эффективность и быстроту восстановления данных в случае необходимости.

Были проведены тесты над таблицами с большим количеством строк, которые доказали, что база данных GUSTO\_PDB имеет хорошую производительность и быструю скорость выполнения процедур и запросов.

В базе данных, с помощью соответствующих процедур, были реализованы экспорт и импорт в JSON формате, что поможет обезопасить данные.

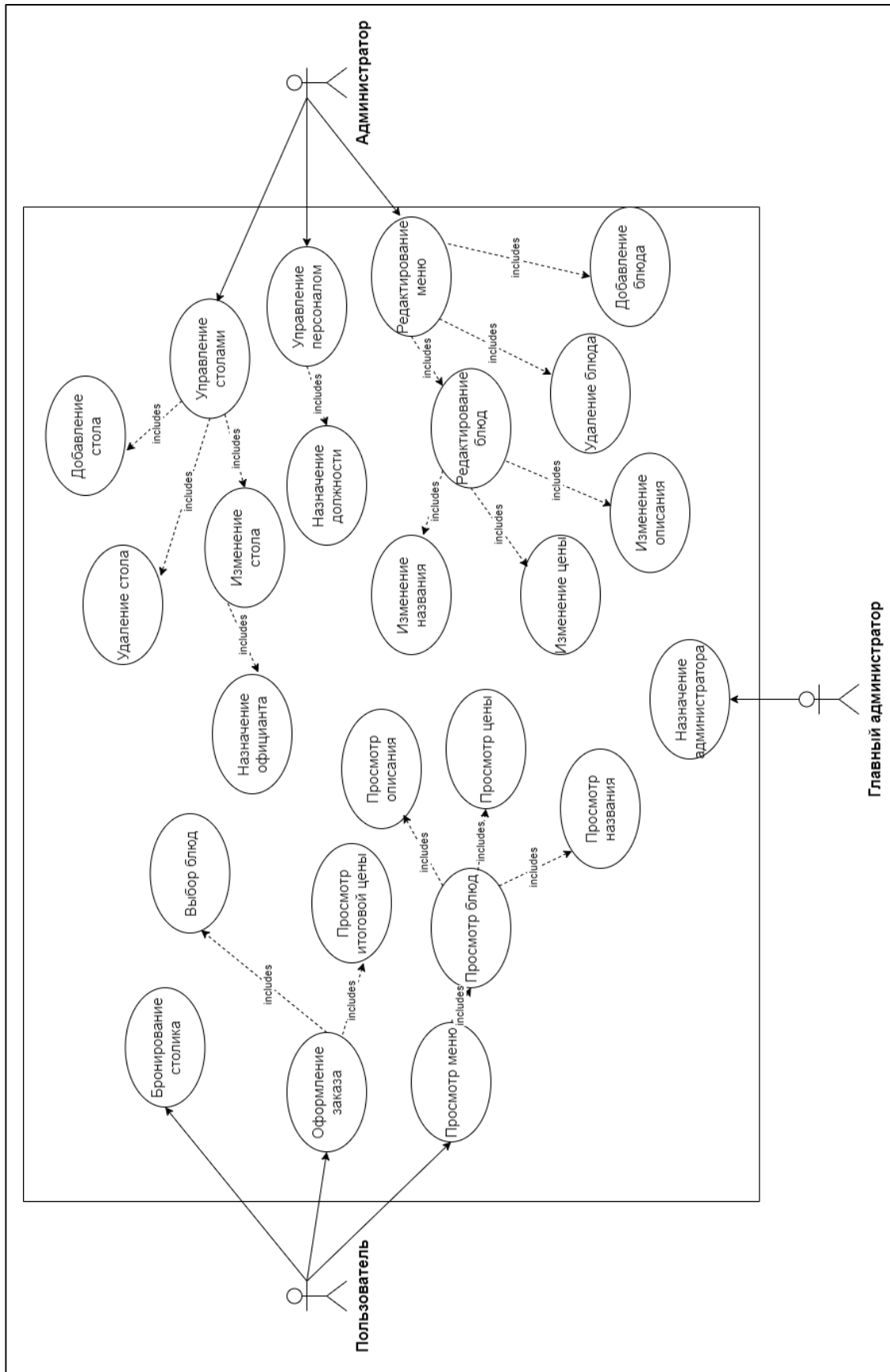
По итогам проекта можно сделать вывод, что созданная база данных полностью соответствует заявленным требованиям. Она обеспечивает надёжное хранение данных, удобный доступ к ним через процедуры и поддержку ключевых функций для работы с информацией. Полученные знания и навыки в области проектирования баз данных и работы с СУБД Oracle могут быть успешно применены в дальнейшем в профессиональной деятельности.

### **Список используемых источников**

1. Официальный сайт Oracle [Электронный ресурс] / Режим доступа – URL:  
<https://www.oracle.com/> – Дата доступа: 06.10.2024.
2. Официальная документация Oracle [Электронный ресурс] / Режим доступа – URL: <https://docs.oracle.com/en/> – Дата доступа: 10.10.2024.
3. Резервное копирование и восстановление базы данных Oracle [Электронный ресурс] / Режим доступа – URL: <https://medium.com/hetman-software/oracle-database-data-backup-and-restore-bf788a0c114c> - Дата доступа: 15.11.2024.
4. Работа с файлами в Oracle, [Электронный ресурс] / Режим доступа: [https://docs.oracle.com/cd/F49540\\_01/DOC/server.815/a68001/utl\\_file.htm](https://docs.oracle.com/cd/F49540_01/DOC/server.815/a68001/utl_file.htm). – Дата доступа: 22.11.2024.

# Приложение А

## Диаграмма вариантов использования



## Приложение Б

### Листинг спецификаций пакетов процедур

```
create or replace PACKAGE sys_admin_package AS
    PROCEDURE AddAdmin (
        p_first_name IN VARCHAR2,
        p_last_name IN VARCHAR2
    );
    PROCEDURE DeleteAdmin (
        p_admin_id IN NUMBER
    );
    PROCEDURE UpdateAdmin (
        p_admin_id IN NUMBER,
        p_first_name IN VARCHAR2,
        p_last_name IN VARCHAR2
    );
    PROCEDURE ExportAdminsJSON;
    PROCEDURE ImportAdminsJSON;
end;

create or replace PACKAGE admin_package AS
    PROCEDURE AddCustomer (
        p_first_name IN VARCHAR2,
        p_last_name IN VARCHAR2,
        p_phone IN VARCHAR2
    );
    PROCEDURE DeleteCustomer (
        p_customer_id IN NUMBER
    );
    PROCEDURE UpdateCustomer (
        p_customer_id IN NUMBER,
        p_first_name IN VARCHAR2,
        p_last_name IN VARCHAR2,
        p_phone IN VARCHAR2
    );
    PROCEDURE AddMenuItem (
        p_name IN VARCHAR2,
        p_description IN VARCHAR2,
        p_price IN NUMBER,
        p_category IN NUMBER
    );
    PROCEDURE UpdateMenuItem (
        p_item_id IN NUMBER,
```

```

        p_name IN VARCHAR2,
        p_description IN VARCHAR2,
        p_price IN NUMBER,
        p_category IN VARCHAR2
    );
    PROCEDURE DeleteMenuItem (
        p_item_id IN NUMBER
    );
    PROCEDURE AddCategory(
        p_category_name VARCHAR2,
        p_category_description VARCHAR2
    );
    PROCEDURE UpdateCategory(
        p_id NUMBER,
        p_category_name VARCHAR2,
        p_category_description VARCHAR2
    );
    PROCEDURE DeleteCategory(
        p_id NUMBER
    );

    PROCEDURE AddPosition(
        p_position_name VARCHAR2
    );
    PROCEDURE UpdatePosition(
        p_position_id NUMBER,
        p_position_name VARCHAR2
    );
    PROCEDURE DeletePosition(
        p_position_id NUMBER
    );
    PROCEDURE AddEmployee(
        p_first_name VARCHAR2,
        p_last_name VARCHAR2,
        p_position_id NUMBER,
        p_salary NUMBER
    );
    PROCEDURE UpdateEmployee(
        p_employee_id NUMBER,
        p_first_name VARCHAR2,
        p_last_name VARCHAR2,
        p_position_id NUMBER,
        p_salary NUMBER
    );
    PROCEDURE AddTable (

```

```

        p_capacity IN NUMBER
    );
    PROCEDURE UpdateTable (
        p_table_id IN NUMBER,
        p_capacity IN NUMBER
    );
    PROCEDURE DeleteTable (
        p_table_id IN NUMBER
    );
    PROCEDURE AssignWaiterToTable (
        p_table_id IN NUMBER,
        p_employee_id IN NUMBER
    );
    PROCEDURE DeleteEmployee(
        p_employee_id NUMBER
    );
    PROCEDURE ClearWaiterFromTable (
        p_table_id IN NUMBER
    );

    PROCEDURE VIPCustomersPrice;
    PROCEDURE VIPCustomersOrderItems;
    PROCEDURE DailyBuyStatistics;
    PROCEDURE MonthlyBuyStatistics;
    PROCEDURE YearlyBuyStatistics;
    PROCEDURE TotalBuyStatistics;

end admin_package;

create or replace package user_package as
    PROCEDURE AddOrder(
        p_customer_id NUMBER
    );
    procedure CancelOrder(
        p_customer_id NUMBER,
        p_order_id NUMBER
    );
    PROCEDURE AddItemToOrder (
        p_customer_id NUMBER,
        p_order_id NUMBER,
        p_order_item NUMBER,
        p_quantity NUMBER
    );
    PROCEDURE DeleteItemFromOrder(
        p_customer_id NUMBER,

```



```
        p_order_item_id NUMBER
    );
    PROCEDURE CreateReservation (
        p_customer_id IN NUMBER,
        p_table_id IN NUMBER,
        p_reservation_time IN TIMESTAMP
    );
    PROCEDURE DeleteReservation (
        p_customer_id IN NUMBER,
        p_table_id IN NUMBER
    );
end;
```

## Приложение В

### Процедуры, выполняемые планировщиками

```
create or replace PROCEDURE Clear_Day(
    p_item_id number
) as
    item_name VARCHAR2(50);
    buy_number number;
begin
    select name into item_name from MenuItems where
item_id = p_item_id;
    select daily_buy into buy_number from MenuItems
where item_id = p_item_id;

    dbms_output.put_line('За день блюдо ' || item_name
|| ' было продано ' || buy_number || ' раз.');
```

```
    update MenuItems
set daily_buy = 0 where item_id = p_item_id;
COMMIT;
EXCEPTION
WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Ошибка: ' || SQLERRM);
    ROLLBACK;
    RAISE;
end;
```

```
create or replace PROCEDURE Clear_Month(
    p_item_id number
) as
    item_name VARCHAR2(50);
    buy_number number;
begin
    select name into item_name from MenuItems where
item_id = p_item_id;
    select monthly_buy into buy_number from MenuItems
where item_id = p_item_id;

    dbms_output.put_line('За месяц блюдо ' || item_name
|| ' было продано ' || buy_number || ' раз.');
```

```
    update MenuItems
set monthly_buy = 0 where item_id = p_item_id;
COMMIT;
EXCEPTION
WHEN OTHERS THEN
```

```

        DBMS_OUTPUT.PUT_LINE('Ошибка: ' || SQLERRM);
        ROLLBACK;
        RAISE;
end;

create or replace PROCEDURE Clear_Year(
    p_item_id number
) as
    item_name VARCHAR2(50);
    buy_number number;
begin
    select name into item_name from MenuItems where
item_id = p_item_id;
    select yearly_buy into buy_number from MenuItems
where item_id = p_item_id;

    dbms_output.put_line('За год блюдо ' || item_name
|| ' было продано ' || buy_number || ' раз.');
```

```

    update MenuItems
    set yearly_buy = 0 where item_id = p_item_id;
    COMMIT;
    EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Ошибка: ' || SQLERRM);
        ROLLBACK;
        RAISE;
end;

create or replace PROCEDURE Clear_Total(
    p_item_id number
) as
    item_name VARCHAR2(50);
    buy_number number;
begin
    select name into item_name from MenuItems where
item_id = p_item_id;
    select total_buy into buy_number from MenuItems
where item_id = p_item_id;

    dbms_output.put_line('За все время блюдо ' ||
item_name || ' было продано ' || buy_number || '
раз.');
```

```

    update MenuItems
    set total_buy = 0 where item_id = p_item_id;
    COMMIT;

```

```

        EXCEPTION
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('Ошибка: ' || SQLERRM);
            ROLLBACK;
            RAISE;
end;

create or replace PROCEDURE Clear_Orders as
begin
    DELETE FROM OrderItems;
    DELETE FROM Orders;
    Commit;
end;

create or replace PROCEDURE Clear_Old_Reservation as
begin
    DELETE FROM Reservations where reservation_date <
SYSTIMESTAMP - INTERVAL '24' HOUR;
    Commit;
end;

```

## Приложение Г

### Процедура импорта из JSON файла

```
PROCEDURE ImportAdminsJSON AS
  l_file UTL_FILE.FILE_TYPE;
  l_json_data CLOB;
  l_admin_id NUMBER;
  l_fname VARCHAR2(50);
  l_lname VARCHAR2(50);
  v_ErrorMessage VARCHAR2(4000);
BEGIN
  l_file := UTL_FILE.FOPEN('JSON_DIR', 'customers.json', 'r', 32767);

  LOOP
    BEGIN
      UTL_FILE.GET_LINE(l_file, l_json_data);
      SELECT JSON_VALUE(l_json_data, '$.admin_id'),
JSON_VALUE(l_json_data, '$.first_name'), JSON_VALUE(l_json_data,
'$.last_name')
      INTO l_admin_id, l_fname, l_lname
      FROM DUAL;

      MERGE INTO Admins r
      USING (SELECT l_admin_id AS admin_id, l_fname AS first_name,
l_lname AS last_name FROM DUAL) src
      ON (r.admin_id = src.admin_id)
      WHEN MATCHED THEN
        UPDATE SET r.first_name = src.first_name, r.last_name = src.last_name
      WHEN NOT MATCHED THEN
        INSERT (r.admin_id, r.first_name, r.last_name) VALUES
(src.admin_id, src.first_name, src.last_name);
    EXCEPTION
      WHEN NO_DATA_FOUND THEN
        EXIT;
      WHEN OTHERS THEN
        v_ErrorMessage := SQLERRM;
        DBMS_OUTPUT.put_line('Ошибка при импорте Admins: ' ||
v_ErrorMessage || ' Данные: ' || l_json_data);

        CONTINUE;
    END;
  END LOOP;

  UTL_FILE.FCLOSE(l_file);
```

```
    DBMS_OUTPUT.PUT_LINE('Импорт таблицы Admins завершен  
успешно.');
```

EXCEPTION

```
    WHEN OTHERS THEN  
        v_ErrorMessage := SQLERRM;  
        DBMS_OUTPUT.put_line('Ошибка при импорте Admins: ' ||  
v_ErrorMessage);  
        IF UTL_FILE.IS_OPEN(l_file) THEN  
            UTL_FILE.FCLOSE(l_file);  
        END IF;  
END;
```