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 банковского приложения с применением технологии шифрования и маскирования в БД»

Выполнил студент Станчик Максим Андреевич

(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**

Современные банковские приложения играют ключевую роль в финансовой сфере, обеспечивая удобный и безопасный доступ к банковским услугам для клиентов. С увеличением объемов обрабатываемых данных и ростом числа киберугроз безопасность информации становится одной из главных задач для финансовых организаций. В условиях постоянного внимания к вопросам защиты личных данных и конфиденциальности клиентов, внедрение технологий шифрования и маскирования в базы данных становится не только актуальным, но и необходимым.

Целью данного курсового проекта является разработка базы данных для банковского приложения, которая будет интегрировать современные методы шифрования и маскирования данных. Это позволит обеспечить высокий уровень безопасности, защитив чувствительную информацию, такую как персональные данные клиентов, счета и кредиты.

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

В рамках проекта будут рассмотрены основные методы шифрования, используемые в современных системах управления базами данных (СУБД), а также подходы к маскированию данных, позволяющие минимизировать риски утечек информации.

Самой широко распространённой моделью базы данных является реляционная, основанная на теории множеств. В качестве СУБД для её разработки была выбрана Oracle 23c Express Edition Release 21.0.0.0.0 (version 21.3.0.0.0) ввиду своей высокой производительности и надёжности.

Для демонстрации работы базы данных необходимо разработать приложение, взаимодействующее с ней. Приложение реализовано с помощью языка программирования Java с использованием Spring и Hibernate.

Чтобы добиться реализации необходимого функционала, нужно выполнить следующие задачи:

* Провести анализ предметной области и определить основные сущности и атрибуты базы данных;
* Разработать концептуальную модель базы данных в нотации ER-диаграмм;
* Преобразовать концептуальную модель в логическую модель в нотации реляционной алгебры;
* Реализовать логическую модель в физической модели с использованием Oracle;
* Наполнить базу данных тестовыми данными;
* Разработать хранимые процедуры, функции и триггеры для реализации бизнес-логики базы данных;
* Разработать запросы для получения необходимой информации из базы данных.

1. **Постановка задачи**

**1.1 Аналитический обзор аналогов**

В настоящее время банки и финансовые учреждения активно внедряют современные информационные системы для управления своими операциями и повышением уровня обслуживания клиентов. Развитие технологий, особенно в области шифрования и защиты данных, становится особенно важным в свете растущих угроз кибербезопасности. Банковские приложения, предлагающие пользователям широкий спектр услуг — от управления счетами до проведения трансакций — должны обеспечивать высокий уровень защиты личной и финансовой информации.

Технологии шифрования и маскирования данных играют ключевую роль в этом процессе, обеспечивая безопасность информации как при передаче, так и при хранении. Реализация таких технологий в базах данных банковских приложений позволяет минимизировать риски утечек данных и защитить пользователей от мошенничества.

Для выполнения максимально эффективного и возможного маскирования и шифрования данных, важно провести анализ существующих аналогов.

Wells Fargo — один из крупнейших банков в США, предлагающим широкий спектр финансовых услуг как для частных, так и для корпоративных клиентов. Банк предоставляет доступ к различным продуктам, включая банковские счета, кредиты, ипотечные услуги, кредитные карты и инвестиционные решения. Веб-сайт Wells Fargo предоставляет удобный интерфейс для управления всеми этими услугами. Цифровые платформы, такие как Wells Fargo Online и мобильное приложение, обеспечивают удобный доступ к счетам, возможность оплаты и перевода денег, а также инструменты для финансового планирования. Банк предлагает высококачественную поддержку клиентов через различные каналы, включая телефон и онлайн-чат. Однако, несмотря на свои сильные стороны, Wells Fargo сталкивается с критикой за случаи мошенничества и вопросы безопасности, которые могут вызывать у клиентов опасения. Интерфейс веб-приложения представлен на рисунке 1.1.

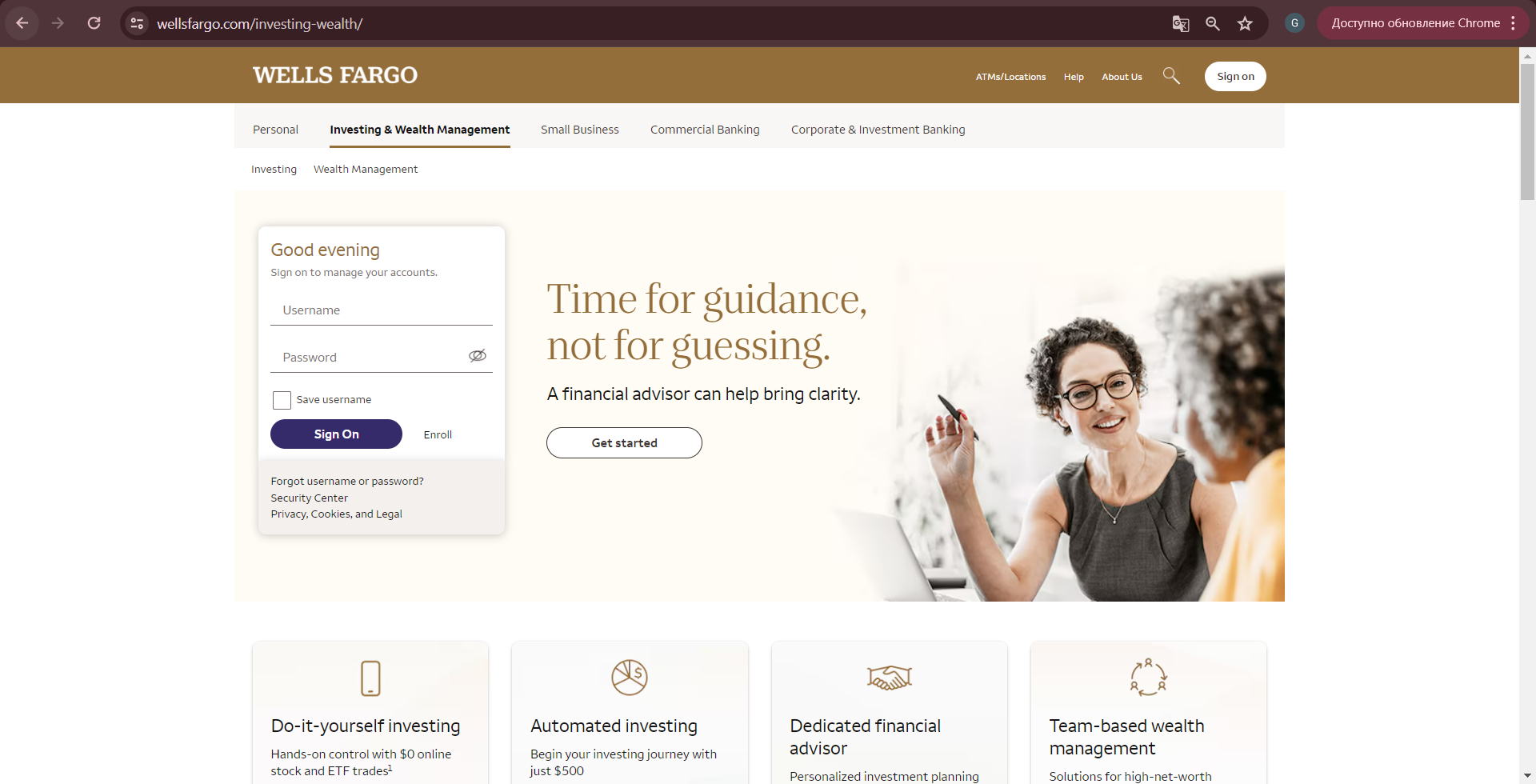


Рисунок 1.1 – интерфейс Wells Fargo

PNC Online Banking — это цифровая платформа от PNC Bank, одного из крупнейших банков США, обеспечивающая пользователям доступ к широкому спектру финансовых услуг. Платформа позволяет управлять текущими и сберегательными счетами, кредитными картами, ипотечными и личными кредитами. Пользователи могут легко осуществлять переводы, оплачивать счета и отслеживать свои расходы через удобный интерфейс. PNC также предлагает инструменты для финансового планирования, включая бюджетирование и управление сбережениями. Клиенты могут рассчитывать на качественную поддержку через телефонные линии и онлайн-чат. Однако некоторые пользователи отмечают недостаточную функциональность мобильного приложения и проблемы с безопасностью, что может вызывать опасения по поводу конфиденциальности данных. Интерфейс веб-приложения представлен на рисунке 1.2.

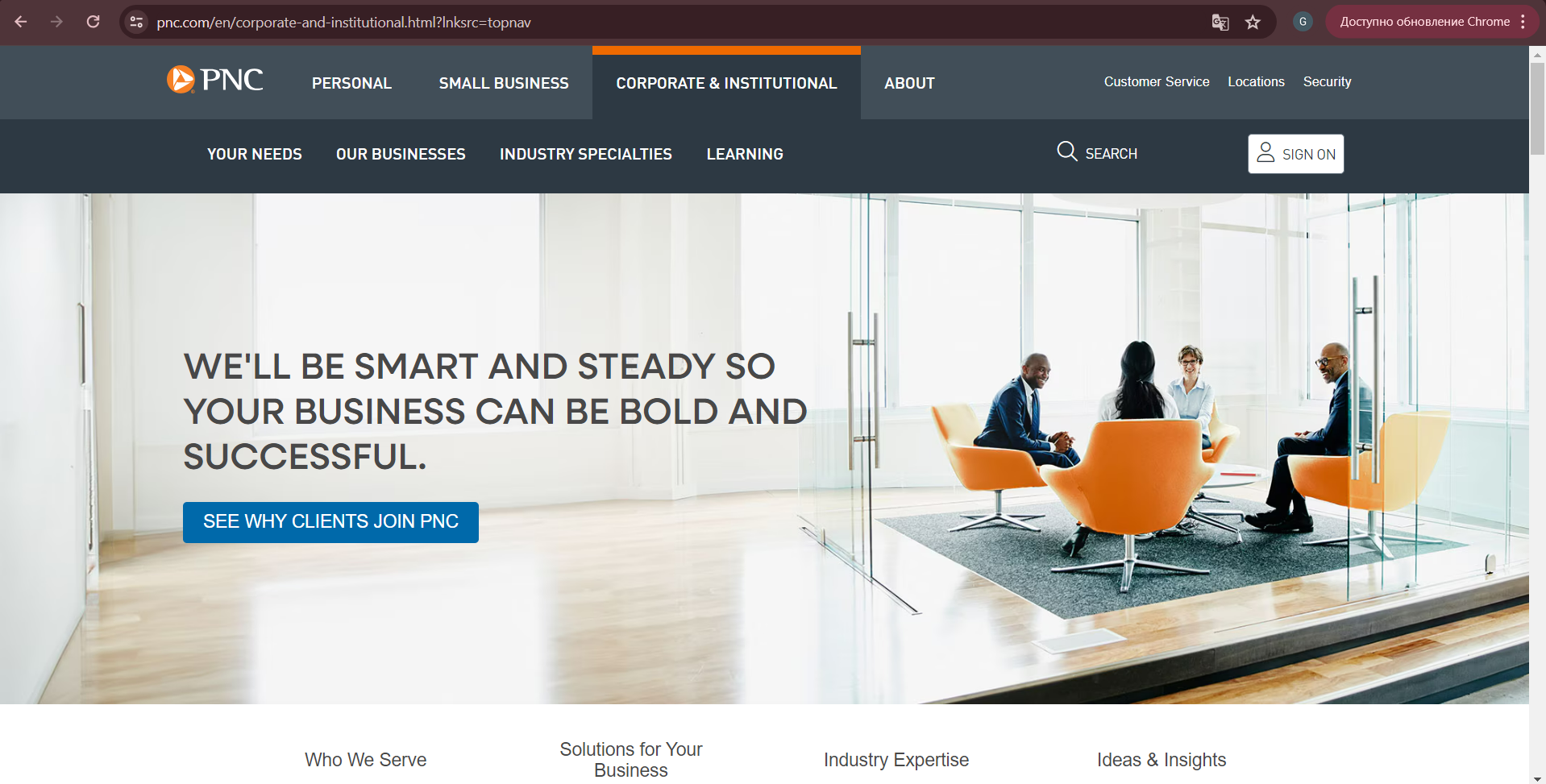


Рисунок 1.2 – интерфейс PNC Online Banking

Таким образом, можно сделать вывод, что существующие аналоги базы данных для банковского приложения имеют свои преимущества и недостатки, и не полностью удовлетворяют потребностям современных предприятий.

**1.2 Спецификация требований**

База данных должна быть реализована в СУБД Oracle, подключение к которой следует осуществлять при помощи одного из перечисленных специальных пользователей: администратор, авторизованный пользователь, гость. Доступ к данным осуществляется с помощью процедур, права на выполнение которых должны быть выданы пользователям.

Необходимо реализовать импорт и экспорт данных из JSON протестировать производительность базы данных при вставке в неё 100000 строк и проанализировать планы запросов, а также применить технологию шифрования и маскирования в базах данных.

Помимо этого, на основании рассмотренных аналогичных решений, можно сформулировать следующие требования к разрабатываемому программному продукту:

− администратор:

1. вставлять записи в таблицы, удалять записи из таблиц (любые);
2. создавать пользователей, удалять пользователей, обновлять

пользователей, просмотр информации о пользователях (id, ФИО и т.д.), просмотр данных таблиц;

1. возможность блокировки/разблокировки карты;
2. фильтрация и сортировка платежей по дате, сумме, типу;

− клиент:

1. управление пользовательскими счетами (добавление, удаление, чтение);
2. осуществление платежей;

3) управление оформленными кредитами и банковскими карточками.

1.2 Использование технологий при разработке проекта

Для выполнения данного курсового проекта были использованы следующие технологии:

Oracle 21c Express Edition Release 21.0.0.0.0 (version 21.3.0.0.0) —реляционная система управления базами данных (СУБД). Она достаточна надежна, так как предлагает надежные решения для резервного копирования и восстановления данных, такие как RMAN (Recovery Manager), использует систему управления доступом на основе ролей (RBAC), поддерживает свойства ACID (Atomicity, Consistency, Isolation, Durability), гарантируя целостность и надежность транзакций. Все выше перечисленные преимущества критически важны для финансовых приложений, что делает данную СУБД отличным выбором для разработки БД проекта.

Docker version 25.0.3, build 4debf41 — это платформа для разработки, доставки и запуска приложений в контейнерах. Docker предоставляет некоторые возможности для разработчиков, что сделало его неплохим инструментом для разработки приложения:

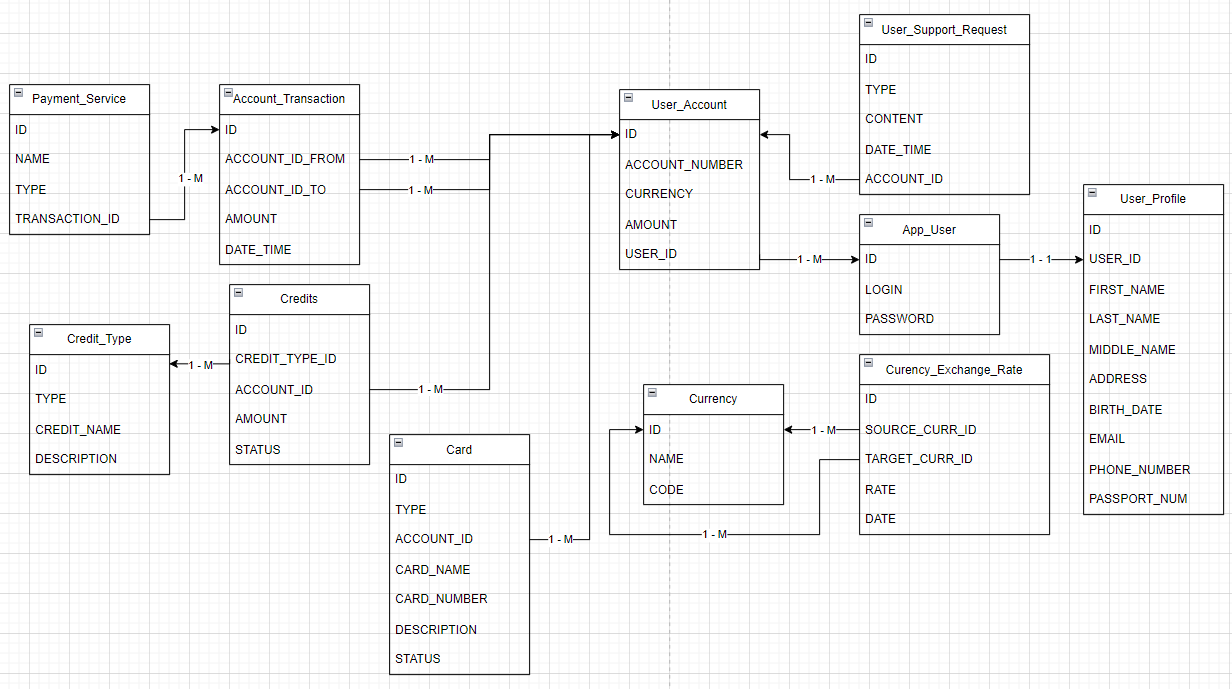
* Контейнеры позволяют разработчикам упаковывать приложения и все их зависимости в единый образ, что обеспечивает согласованность среды выполнения на разных этапах разработки и развертывания;
* Созданные образы можно легко развертывать на различных серверах или облачных платформах. Это значительно упрощает процесс переноса приложения из разработки в продакшен;
* Docker позволяет создавать стабильные и воспроизводимые среды для тестирования, что упрощает процесс CI/CD (непрерывная интеграция и доставка).
* SQL Developer — это интегрированная среда разработки (IDE) для работы с базами данных Oracle. SQL Developer предоставляет мощные инструменты для управления базами данных, написания и выполнения SQL-запросов, а также для визуального проектирования баз данных.

Исходя из описанных выше технологий, можно сделать вывод о том, что для реализации данного проекта были использованы современные инструменты и технологии, что позволило повысить эффективность работы. Oracle предоставляет высокую производительность и надежность в работе с базами данных, SQL Developer обеспечивает удобное и интуитивно понятное управление базами данных, а Docker обеспечивает согласованность среды выполнения на разных этапах разработки и развертывания. В целом, использование таких технологий позволило реализовать проект на высоком уровне и обеспечить его успешное выполнение.

2. Проектирование базы данных

2.1 Схема базы данных

Логическая схема БД приведена на рисунке 2.1.



Листинг 2.1 – логическая схема базы данных

Она включает в себя 11 таблиц: Payment\_Service, Credit\_Type, Credits, Transaction, Card, Account, Currency, App\_User, Currency\_Exchange\_Rate, User\_Support\_Request, User\_Profile.

Таблица App\_User хранит информацию о пользователях базы данных. Описание её столбцов представлено в таблице 2.1.

Таблица 2.1. Описание столбцов таблицы App\_User

|  |  |  |  |
| --- | --- | --- | --- |
| Название столбца | Тип данных | Описание | Ограничения |
| ID | number(10) | Идентификатор пользователя, первичный ключ | not null |
| LOGIN | nvarchar2(255) | Логин пользователя | not null |
| PASSWORD | raw(2048) | Пароль пользователя | not null |

Таблица User\_Profile хранит более подробную информацию о пользователе. У одного пользователя может быть только один профиль. Описание её столбцов представлено в таблице 2.2.

Таблица 2.2. Описание столбцов таблицы User\_Profile

|  |  |  |  |
| --- | --- | --- | --- |
| Название столбца | Тип данных | Описание | Ограничения |
| ID | number(10) | Идентификатор дополнительной информации о пользователе, первичный ключ | not null |
| USER\_ID | number(10) | Идентификатор пользователя, внешний ключ, ссылается на столбец ID в таблице App\_User | not null |
| FIRST\_NAME | nvarchar2(50) | Имя пользователя | not null |
| LAST\_NAME | nvarchar2(50) | Фамилия пользователя | not null |
| MIDDLE\_NAME | nvarchar2(50) | Второе имя пользователя | (Отсутствуют) |
| ADDRESS | nvarchar2(255) | Адрес пользователя | (Отсутствуют) |
| BIRTH\_DATE | date | День рождения пользователя | (Отсутствуют) |
| EMAIL | nvarchar2(100) | Электронная почта пользователя | (Отсутствуют) |
| PHONE\_NUMBER | nvarchar2(50) | Номер телефона пользователя | (Отсутствуют) |
| PASSPORT\_NUM | nvarchar2(20) | Номер паспорта | not null |

Таблица User\_Support\_Request хранит информацию о запросе поддержки пользователя. Описание её столбцов представлено в таблице 2.3.

Таблица 2.3. Описание столбцов таблицы User\_Support\_Request

|  |  |  |  |
| --- | --- | --- | --- |
| Название столбца | Тип данных | Описание | Ограничения |
| ID | number(10) | Идентификатор запроса поддержки | not null |
| TYPE | nvarchar2(50) | Тип запроса поддержки | (Отсутствуют) |
| CONTENT | clob | Содержимое запроса поддержки | (Отсутствуют) |
| DATE\_TIME | timestamp | Дата и время создания запроса | (Отсутствуют) |
| ACCOUNT\_ID | number(10) | Идентификатор счета пользователя, внешний ключ, ссылается на столбец ID в таблице Account | not null |

Таблица User\_Account хранит информацию о счете пользователя, которых может быть несколько. Описание столбцов представлено в таблице 2.4.

Таблица 2.4. Описание столбцов таблицы User\_Account

|  |  |  |  |
| --- | --- | --- | --- |
| Название столбца | Тип данных | Описание | Ограничения |
| ID | number(10) | Идентификатор счета, первичный ключ | not null |
| ACCOUNT\_NUMBER | nvarchar2(16) | Номер счета | not null |
| CURRENCY | varchar2(4) | Код валюты (например, "USD", "EUR"). | not null |
| AMOUNT | number(15,2) | Баланс счета с точностью до двух десятичных знаков | default 0.0 |
| USER\_ID | number(10) | Идентификатор пользователя, внешний ключ, ссылается на столбец ID в таблице App\_User | not null |

Таблица Currency хранит информацию о валюте. Описание столбцов таблицы представлено в таблице 2.5.

Таблица 2.5. Описание столбцов таблицы Currency

|  |  |  |  |
| --- | --- | --- | --- |
| Название столбца | Тип данных | Описание | Ограничения |
| ID | number(10) | Идентификатор валюты, первичный ключ | not null |
| NAME | varchar2(100) | Название валюты | not null |
| CODE | varchar2(3) | Код валюты | not null |

Таблица Currency\_Exchange\_Rate хранит информацию о курсе обмена валюты. Описание столбцов таблицы представлено в таблице 2.6.

Таблица 2.6. Описание столбцов таблицы Currency\_Exchange\_Rate

|  |  |  |  |
| --- | --- | --- | --- |
| Название столбца | Тип данных | Описание | Ограничения |
| ID | number(10) | Идентификатор курса обмена, первичный ключ | not null |
| SOURCE\_CURR\_ID | number(10) | Идентификатор исходной валюты, внешний ключ (ссылается на столбец ID в таблице Currency) | not null |
| TARGET\_CURR\_ID | number(10) | Идентификатор целевой валюты, внешний ключ (ссылается на столбец ID в таблице Currency) | not null |
| RATE | number(15, 2) | Обменный курс между исходной и целевой валютами | not null |
| RATE\_DATE | date | Дата, на которую установлен курс обмена | not null |

Таблица Transaction хранит информацию о транзакции. Описание столбцов таблицы представлено в таблице 2.7.

Таблица 2.7. Описание столбцов таблицы Account\_Transaction

|  |  |  |  |
| --- | --- | --- | --- |
| Название столбца | Тип данных | Описание | Ограничения |
| ID | number(10) | Идентификатор транзакции, первичный ключ | not null |
| ACCOUNT\_ID\_FROM | number(10) | Идентификатор аккаунта, откуда происходит транзакция, внешний ключ, который ссылается на столбец ID в таблице Acount | not null |
| ACCOUNT\_ID\_TO | number(10) | Идентификатор аккаунта, куда направляется транзакция, внешний ключ, который ссылается на столбец ID в таблице Acount | not null |
| AMOUNT | number(15, 2) | Сумма транзакции с точностью до двух десятичных знаков | not null |
| DATE\_TIME | timestamp | Дата и время проведения транзакции | not null |

Таблица Credits хранит информацию о кредите. Описание столбцов таблицы представлено в таблице 2.8.

Таблица 2.8. Описание столбцов таблицы Credits

|  |  |  |  |
| --- | --- | --- | --- |
| Название столбца | Тип данных | Описание | Ограничения |
| ID | number(10) | Идентификатор кредита, первичный ключ | not null |
| CREDIT\_TYPE\_ID | number(10) | Идентификатор типа кредита, внешний ключ (ссылается на столбец ID в таблице Credit\_Type) | not null |
| ACCOUNT\_ID | number(10) | Идентификатор аккаунта, к которому относится кредит, внешний ключ (ссылается на столбец ID в таблице Account) | not null |
| AMOUNT | number(15, 2) | Сумма кредита с точностью до двух десятичных знаков | not null |
| STATUS | char(1) | Статус кредита (0, "-", "+") | (отсутствуют) |

Таблица Card хранит информацию о банковской карте. Описание столбцов таблицы представлено в таблице 2.9.

Таблица 2.9. Описание столбцов таблицы Card

|  |  |  |  |
| --- | --- | --- | --- |
| Название столбца | Тип данных | Описание | Ограничения |
| ID | number(10) | Идентификатор банковской карты, первичный ключ | not null |
| TYPE | nvarchar2(50) | Тип карты | not null |
| ACCOUNT\_ID | number(10) | Идентификатор аккаунта, к которому относится карта, внешний ключ (ссылается на столбец ID в таблице Account) | not null |
| CARD\_NAME | nvarchar2(100) | Название карты | (Отсутствуют) |
| CARD\_NUMBER | number(16) | Номер карты | not null |
| DESCRIPTION | nvarchar2(255) | Описание карты | (Отсутствуют) |
| STATUS | nvarchar2(20) | Статус карты | (Отсутствуют) |

Таблица Credit\_Type хранит информацию о типе кредита. Описание столбцов таблицы представлено в таблице 2.10.

Таблица 2.10. Описание столбцов таблицы Credit\_Type

|  |  |  |  |
| --- | --- | --- | --- |
| Название столбца | Тип данных | Описание | Ограничения |
| ID | number(10) | Идентификатор типа кредита, первичный ключ | not null |
| TYPE | nvarchar2(50) | Тип кредита | not null |
| CREDIT\_NAME | nvarchar2(100) | Название кредита | not null |
| DESCRIPTION | nvarchar2(255) | Описание типа кредита | not null |

Таблица Payment\_Service хранит информацию о платёжном сервисе. Описание столбцов таблицы представлено в таблице 2.11.

Таблица 2.11. Описание столбцов таблицы Payment\_Service

|  |  |  |  |
| --- | --- | --- | --- |
| Название столбца | Тип данных | Описание | Ограничения |
| ID | number(10) | Идентификатор платёжного сервиса, первичный ключ | not null |
| NAME | nvarchar2(100) | Название платёжного сервиса | not null |
| TYPE | nvarchar2(50) | Тип платёжного сервиса | not null |
| TRANSACTION\_ID | number(10) | Идентификатор транзакции, связанной с платёжным сервисом, внешний ключ, который ссылается на идентификатор транзакции в таблице Transaction | not null |

Скрипт создания всех таблиц базы данных приведён в приложении А.

**3. Разработка объектов базы данных**

**3.1 Создание пользователей, ролей**

Для работы с базой данных необходимо выделить несколько пользователей с определёнными привилегиями. Начало разработки базы данных начинается с созданий её объектов. Для этого необходимо создать пользователя и администратора и присвоить им профили безопасности.

Администратору будут выданы привилегии создания сессии, создание и изменение основных объектов базы данных: таблицы, последовательностей и т.д. Пользователю будут выданы права на вызов получение данных из определённых таблиц. С выданными правами пользователь сможет использовать основную функциональность базы данных. Скрипт создания представлен на листинге 3.1.

|  |
| --- |
| *-- Создание профиля для пользователя (сотрудник банка)*  create profile user\_profile limit  failed\_login\_attempts 5  sessions\_per\_user 10  cpu\_per\_session unlimited  cpu\_per\_call 3000  connect\_time 45  idle\_time 15  logical\_reads\_per\_session default  logical\_reads\_per\_call default  private\_sga 15k  composite\_limit 5000000;  *-- Создание профиля для администратора (разработчик бд)*  create profile admin\_profile limit  sessions\_per\_user unlimited  cpu\_per\_session unlimited  cpu\_per\_call unlimited  connect\_time unlimited  idle\_time unlimited  logical\_reads\_per\_session default  logical\_reads\_per\_call default  private\_sga unlimited  composite\_limit unlimited;  *-- Клиент - пользователь приложения*  *-- Создание администратора (разработчик бд)*  create user admin\_user identified by admin\_password  default tablespace users\_tbs  temporary tablespace temp\_tbs  profile admin\_profile;  *-- Создание пользователя (сотрудник банка)*  create user user\_user IDENTIFIED by user\_password  default TABLESPACE USERS\_TBS  temporary TABLESPACE TEMP\_TBS  profile user\_profile; |

Листинг 3.1 – Создание пользователя для менеджера

**3.2 Создание пользовательских процедур**

Для доступа к данным были реализованы хранимые процедуры, которые можно разделить на 3 категории в зависимости от вариантов использования: общие для всех, только для пользователя и только для администратора.

Список хранимых процедур, являющихся общими для всех, приведён в таблице 3.1.

Таблица 3.1. Список процедур для всех

|  |  |
| --- | --- |
| Название процедуры | Описание |
| GetListCurrenciesAndExchangeRates | Получить список всех валют и их курс обмена |
| GetClientProfileInfo | Получить информацию о профиле пользователя |
| GetAllClientSupportRequests | Получить все запросы в службу поддержки, проведенные клиентом за определенный период времени |
| GetAllCreditTypes | Получить информацию о типах кредитов |
| GetListOfClientAccountsByClientId | Просмотреть все счета клиента по его идентификатору |
| DeleteClientProfile | Удалить профиль клиента |

Список процедур только для администратора приведён в таблице 3.2.

Таблица 3.2. Список процедур только для администраторов (сотрудников банка)

|  |  |
| --- | --- |
| Название процедуры | Описание |
| GetTransactionById | Получить информацию о транзакции по её идентификатору |
| GetCardById | Получить информацию о банковской карте по её идентификатору |
| AddLoanType | Добавить тип кредита |
| UpdateLoanType | Изменить тип кредита |
| DeleteLoanType | Удалить тип кредита |
| UpdateCurrencyInfo (было бы круто еще курс к доллару устанавливать) | Изменить информацию о валюте |
| GetClientInfoById | Получить информацию о клиенте по идентификатору клиента |
| GetClientLoansById | Получить информацию о кредитах клиента по идентификатору клиента |
| GetAllClientCardsById | Получить информацию о всех банковских картах клиента |
| AddLoanTransaction | Добавить новую транзакцию клиенту |
| UpdateClientCardStatus | Изменить статус банковской карты клиента (заблокирована / доступна) |
| GetAllClientInfos | Получить список всех клиентов и информацию о них |
| GetAllClientCards | Получить список всех банковских карт клиента по идентификатору клиента |
| ProcessLoanApplication | Одобрить или отклонить заявку на получение кредита |
| GetLoanApplications | Получить список всех заявок на кредиты |
| AddAccountToClient | Добавить счет клиенту |
| CloseClientAccount | Закрыть счет клиенту |
| AddBankCard | Добавить банковскую карту |
| DeleteBankCard | Удалить банковскую карту |
| GetAllCurrencies | Вывод данных из таблицы Currency |
| GetAllAppUsers | Вывод данных из таблицы App\_User |
| GetAllUserAccounts | Вывод данных из таблицы User\_Account |
| GetAllUserProfiles | Вывод данных из таблицы User\_Profile |
| GetAllUserSupportRequests | Вывод данных из таблицы User\_Support\_Request |
| GetAllCurrencyExchangeRates | Вывод данных из таблицы Currency\_Exchange\_Rate |
| GetAllAccountTransactions | Вывод данных из таблицы Account\_Transaction |
| GetAllCards | Вывод данных из таблицы Card |
| GetAllPaymentServices | Вывод данных из таблицы Payment\_Service |
| GetAllCredits | Вывод данных из таблицы Credits |

Таблица 3.3. Список процедур только для клиентов

|  |  |
| --- | --- |
| Название процедуры | Описание |
| GetAccountBalance | Получить текущий баланс счета |
| GetTransactionHistory | Получить историю всех транзакций клиента за указанный период времени |
| CreateSupportRequest | Создать запрос в службу поддержки с указанием темы и содержимого |
| SubmitLoanApplication | Подать заявку на кредит с указанием суммы и типа кредита |
| TransferFunds | Перевести средства с одного счета клиента на другой |
| ExchangeFunds | Обменять средства с одного счета клиента на другой с использованием указанного курса обмена валют |
| ChangeAccountCurrency | Изменить валюту счета |
| RegisterClient | Регистрация клиента |
| LoginClient | Авторизация клиента |

Реализация всех приведённых выше процедур и присвоение привилегий на их вызов представлено в приложении Б.

Таблица 3.4. Вспомогательные процедуры

|  |  |
| --- | --- |
| Название процедуры | Описание процедуры |
| encrypt(p\_plain\_text VARCHAR2) | шифрует обычный текст. |
| decrypt(p\_encrypted\_text RAW) | расшифровывает зашифрованный текст |
| generate\_salt(p\_email VARCHAR2) | генерирует соль на основе электронной почты сотрудника |

**3.3 Разработка функций**

В базе данных была создана единственная функция, которая возвращает номер счета. Её реализация представлена в приложении В.

**3.4 Разработка индексов базы данных**

Индексы — это структуры данных, построенные на основе одного или нескольких столбцов таблицы. Индекс предоставляет быстрый доступ к данным таблицы и позволяет ускорить выполнение операций SELECT, UPDATE и DELETE, так как он содержит отсортированные значения из одного или нескольких столбцов и указатели на строки таблицы.

Применение индексов представляет собой компромисс между ускорением получения результатов запросов и замедлением обновлений и вставок данных. Первая часть этого компромисса – ускорение запросов – довольно очевидна: если поиск выполняется по отсортированному индексу вместо полного сканирования всей таблиц, то запрос проходит намного быстрее. Но всякий раз, когда вы обновляете, вставляете или удаляете строку таблицы с индексами, индексы также должны быть обновлены соответствующим образом. То есть такие операции на таблицах с индексами обходятся дороже.

На основании проделанных тестов производительности было принято решение о создании индексов для основных таблиц.

Список индексов приведён в таблице 3.6.

Таблица 3.6. Список индексов

|  |  |  |
| --- | --- | --- |
| Название индекса | Таблица, к которой применен индекс | Столбец |
| IDX\_User\_Account\_Account\_Number | User\_Account | ACCOUNT\_NUMBER |
| IDX\_User\_Account\_User\_ID | User\_Account | USER\_ID |
| IDX\_User\_Profile\_User\_ID | User\_Profile | USER\_ID |
| IDX\_User\_Profile\_Email | User\_Profile | EMAIL |
| IDX\_Payment\_Service\_Transaction\_ID | Payment\_Service | TRANSACTION\_ID |
| IDX\_User\_Support\_Request\_Account\_ID | User\_Support\_Request | ACCOUNT\_ID |
| IDX\_User\_Support\_Request\_Date\_Time | User\_Support\_Request | DATE\_TIME |
| IDX\_Card\_Account\_ID | Card | ACCOUNT\_ID |
| IDX\_Card\_Number | Card | CARD\_NUMBER |
| IDX\_Transaction\_Account\_From | Account\_Transaction | ACCOUNT\_ID\_FROM |
| IDX\_Transaction\_Account\_To | Account\_Transaction | ACCOUNT\_ID\_TO |
| IDX\_Transaction\_Date\_Time | Account\_Transaction | DATE\_TIME |

**3.5. Разработка последовательностей**

Последовательности в SQL используются для генерации уникальных числовых идентификаторов. Они часто используются для автоматического создания первичных ключей. Создание последовательностей представлено в листинге 3.2.

|  |
| --- |
| *--Последовательность для поддержки запросов*  *create sequence SEQ\_SUPPORT\_REQUEST\_ID start with 1 increment by 1 nocache;*  *--Последовательность для кредитов*  *CREATE sequence SEQ\_CREDITS\_ID start with 1 increment by 1 nocache;*  *--Последовательность для транзакций*  *CREATE sequence SEQ\_TRANSACTION\_ID start with 1 increment by 1 nocache;*  *--Последовательность для карт*  *CREATE sequence SEQ\_CARD\_ID start with 1 increment by 1 nocache;*  *--Последовательность для типов кредитов*  *create sequence SEQ\_CREDIT\_TYPE\_ID start with 1 increment by 1 nocache;*  *--Последовательность для валют*  *create sequence SEQ\_CURRENCY\_ID start with 1 increment by 1 nocache;*  *--Последовательность для генерации уникальных значений*  *create sequence SEQ\_ACCOUNT\_ID start with 1 increment by 1 nocache;* |

Листинг 3.2 – Создание директории для хранения результатов импорта

**3.8. Директории**

Для реализации экспорта и импорта в JSON создадим необходимую директорию. Реализация представлена дальше в листинге 3.3.

|  |
| --- |
| *CREATE OR REPLACE DIRECTORY JSON\_FILES AS '/home/oracle/files';* |

Листинг 3.2 – Создание директории для хранения результатов импорта

В ней будут создаваться файлы JSON экспорта и импорта. Объект Directory является логической ссылкой в базе данных на каталог файловой системы сервера, где установлена БД Oracle. Владельцем всех объектов Directory в базе данных является пользователь SYS, даже если объект Directory создан другим пользователем. Имена объектов Directory уникальны внутри всей БД. Все объекты Directory хранятся в табличном пространстве SYS.

**4. Описание процедур импорта и экспорта**

**4.1 Экспорт в JSON**

Процедура ExportUserAccountsToJSON предназначена для экспорта данных из таблицы User\_Account в формате JSON. Процедура открывает файл для записи и затем проходит по всем записям в таблице User\_Account. Каждая запись преобразуется в JSON-объект с помощью функции JSON\_OBJECT, где каждому полю записи соответствует ключ JSON-объекта. Полученный JSON-объект записывается в файл. После обработки всех записей файл закрывается. Если в процессе работы процедуры происходит ошибка, файл также закрывается, и ошибка передается вызывающей стороне. Это обеспечивает корректное завершение работы с файлом даже в случае возникновения ошибок. Скрипт создания процедуры ExportUserAccountsToJSON представлен на листинге 4.1.

|  |
| --- |
| CREATE OR REPLACE PROCEDURE ExportUserAccountsToJSON(P\_DIRECTORY IN VARCHAR2) AS  file\_handle UTL\_FILE.file\_type;  json\_record CLOB := '[';  json\_obj JSON\_OBJECT\_T;  first\_record BOOLEAN := TRUE;  BEGIN  file\_handle := UTL\_FILE.FOPEN(P\_DIRECTORY, 'account\_transactions.json', 'w');  FOR rec IN (SELECT \* FROM admin\_user.Account\_Transaction) LOOP  json\_obj := JSON\_OBJECT\_T();  json\_obj.put('ID', rec.ID);  json\_obj.put('ACCOUNT\_ID\_FROM', rec.ACCOUNT\_ID\_FROM);  json\_obj.put('ACCOUNT\_ID\_TO', rec.ACCOUNT\_ID\_TO);  json\_obj.put('AMOUNT', rec.AMOUNT);    json\_obj.put('DATE\_TIME', TO\_CHAR(rec.DATE\_TIME, 'YYYY-MM-DD"T"HH24:MI:SS'));  IF NOT first\_record THEN  json\_record := json\_record || ',';  ELSE  first\_record := FALSE;  END IF;  json\_record := json\_record || json\_obj.to\_clob();  END LOOP;  json\_record := json\_record || ']';  UTL\_FILE.PUT\_LINE(file\_handle, json\_record);  UTL\_FILE.FCLOSE(file\_handle);  EXCEPTION  WHEN OTHERS THEN  IF UTL\_FILE.IS\_OPEN(file\_handle) THEN  UTL\_FILE.FCLOSE(file\_handle);  END IF;  RAISE;  END; |

Листинг 4.1 – Процедура экспорта

Этот файл можно использовать для обмена данными между различными системами или для резервного копирования данных.

**4.2 Импорт в JSON**

Процедура import\_json предназначена для импорта данных в формате JSON в таблицу employees. Скрипт создания процедуры import\_json представлен на листинге 4.2.

директорию. Реализация представлена дальше в листинге 3.3.

|  |
| --- |
| CREATE OR REPLACE PROCEDURE ImportUserAccountsFromJSON IS  v\_file UTL\_FILE.file\_type;  v\_data CLOB := '';  v\_line VARCHAR2(32767);  v\_json\_entry VARCHAR2(32767);  v\_id NUMBER;  v\_acc\_from NUMBER;  v\_acc\_to NUMBER;  v\_amount NUMBER;  v\_date\_time VARCHAR2(30);  v\_exists NUMBER := 0;  BEGIN  v\_file := UTL\_FILE.FOPEN('JSON\_FILES', 'account\_transactions.json', 'r');  DBMS\_OUTPUT.PUT\_LINE('Начало чтения файла...');  LOOP  BEGIN  UTL\_FILE.GET\_LINE(v\_file, v\_line);  v\_data := v\_data || v\_line;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  EXIT;  END;  END LOOP;  UTL\_FILE.FCLOSE(v\_file);  DBMS\_OUTPUT.PUT\_LINE('Содержимое файла: ' || v\_data);  IF v\_data LIKE '[%' AND v\_data LIKE '%]' THEN  v\_data := SUBSTR(v\_data, 2, LENGTH(v\_data) - 2);  END IF;  WHILE INSTR(v\_data, '{') > 0 LOOP  v\_json\_entry := SUBSTR(v\_data, INSTR(v\_data, '{'), INSTR(v\_data, '}') - INSTR(v\_data, '{') + 1);  v\_data := SUBSTR(v\_data, INSTR(v\_data, '}') + 1);  v\_id := TO\_NUMBER(REGEXP\_SUBSTR(v\_json\_entry, '"ID"\s\*:\s\*(\d+)', 1, 1, NULL, 1));  v\_acc\_from := TO\_NUMBER(REGEXP\_SUBSTR(v\_json\_entry, '"ACCOUNT\_ID\_FROM"\s\*:\s\*(\d+)', 1, 1, NULL, 1));  v\_acc\_to := TO\_NUMBER(REGEXP\_SUBSTR(v\_json\_entry, '"ACCOUNT\_ID\_TO"\s\*:\s\*(\d+)', 1, 1, NULL, 1));  v\_amount := TO\_NUMBER(REGEXP\_SUBSTR(v\_json\_entry, '"AMOUNT"\s\*:\s\*([0-9]+(?:\.[0-9]+)?)', 1, 1, NULL, 1));  v\_date\_time := REGEXP\_SUBSTR(v\_json\_entry, '"DATE\_TIME"\s\*:\s\*"([^"]+)"', 1, 1, NULL, 1);  IF v\_id IS NOT NULL AND v\_acc\_from IS NOT NULL AND v\_acc\_to IS NOT NULL AND v\_amount IS NOT NULL THEN  SELECT COUNT(\*) INTO v\_exists  FROM admin\_user.Account\_Transaction  WHERE ID = v\_id;  IF v\_exists = 0 THEN  INSERT INTO admin\_user.Account\_Transaction (ID, ACCOUNT\_ID\_FROM, ACCOUNT\_ID\_TO, AMOUNT, DATE\_TIME)  VALUES (v\_id, v\_acc\_from, v\_acc\_to, v\_amount, TO\_TIMESTAMP(v\_date\_time, 'YYYY-MM-DD"T"HH24:MI:SS'));  DBMS\_OUTPUT.PUT\_LINE('Запись вставлена: ID = ' || v\_id);  ELSE  DBMS\_OUTPUT.PUT\_LINE('Запись уже существует: ID = ' || v\_id || ', пропуск.');  END IF;  ELSE  DBMS\_OUTPUT.PUT\_LINE('Пропуск записи из-за NULL значений: ' || v\_json\_entry);  END IF;  END LOOP;  COMMIT;  DBMS\_OUTPUT.PUT\_LINE('Импорт данных завершен успешно.');  EXCEPTION  WHEN OTHERS THEN  ROLLBACK;  DBMS\_OUTPUT.PUT\_LINE('Ошибка: ' || SQLERRM);  RAISE;  END; |

Листинг 4.2 – Процедура импорта

Процедура открывает файл для чтения и считывает все строки в переменную v\_data. Затем эти данные преобразуются в массив JSON-объектов. Процедура проходит по каждому JSON-объекту в массиве, извлекает данные и вставляет их в таблицу User\_Account. Если в процессе работы процедуры происходит ошибка, все изменения откатываются, и ошибка передается вызывающей стороне. Это обеспечивает корректное завершение работы с файлом и таблицей даже в случае возникновения ошибок.

**5. Тестирование производительности**

Для проверки производительности базы данных необходимо заполнить ее большим количеством различных данных и узнать время выполнения одного запроса. Для тестирования производительности мы добавляем в таблицу User\_Account 100000 строк. Скрипт вставки представлен на листинге 5.1.

|  |
| --- |
| DECLARE  v\_currency\_id number;  v\_user\_id number;  BEGIN  FOR i IN 1..100000 LOOP  SELECT ID INTO v\_user\_id  FROM (  SELECT ID FROM admin\_user.App\_User ORDER BY DBMS\_RANDOM.VALUE  )  WHERE ROWNUM = 1;  SELECT ID INTO v\_currency\_id  FROM (  SELECT ID FROM admin\_user.Currency ORDER BY DBMS\_RANDOM.VALUE  )  WHERE ROWNUM = 1;  INSERT INTO admin\_user.User\_Account (ACCOUNT\_NUMBER, CURRENCY, AMOUNT, USER\_ID)  VALUES (  'ACC' || TO\_CHAR(i),  v\_currency\_id,  ROUND(DBMS\_RANDOM.VALUE(100, 10000), 2),  v\_user\_id  );  IF MOD(i, 1000) = 0 THEN  COMMIT;  END IF;  END LOOP;  COMMIT;  END; |

Листинг 5.1 – Вставка 100000 строк в таблицу

Для получения плана выполнения SQL-запроса в Oracle можно воспользоваться инструментом EXPLAIN PLAN используя функцию “Explain Plan”. [Результат будет отображаться в специальной вкладке “Plan” в формате, смешивающем древовидную структуру и таблицу.](https://www.jetbrains.com/help/datagrip/query-execution-plan.html) Результат представлен на рисунке 5.1.

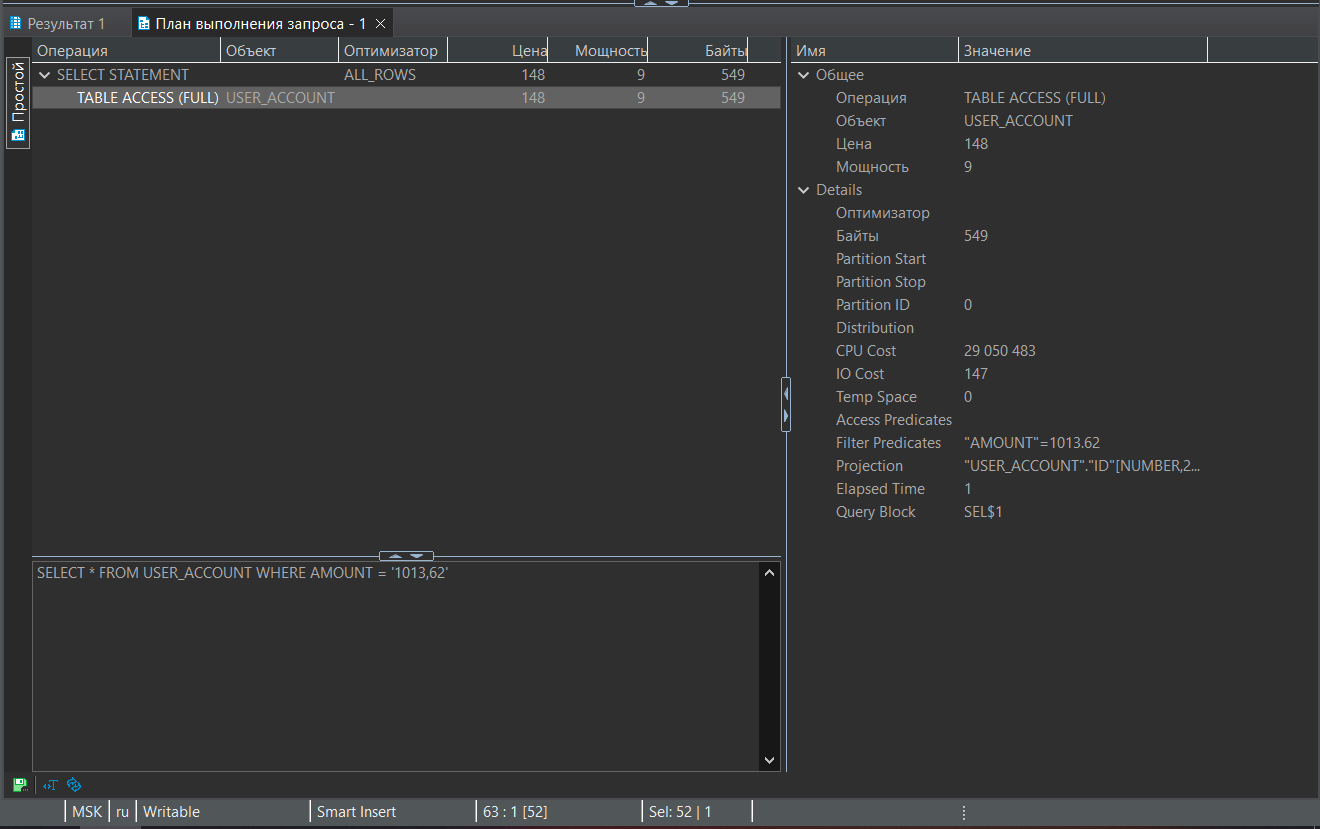


Рисунок 5.1 – Получения плана выполнения запроса

Каждая строка плана выполнения представляет операцию, которую база данных выполнит, а также ее стоимость, учитывая различные факторы, такие как использование индексов, объем данных и другие. Каждая операция оценивается по ожидаемому количеству возвращаемых строк. В данном случае план выполнения состоит из двух строк:

SELECT STATEMENT: это верхний уровень плана выполнения, который описывает операцию SELECT в целом.

TABLE ACCESS FULL: это операция, которая читает все строки таблицы (без использования индексов), и возвращает их в качестве результата запроса. Стоимость этого плана выполнения равна 148 (без указания процента использования ресурсов ЦПУ) и означает, что база данных ожидает, что выполнение этого запроса займет примерно 148 единицы измерения работы (например, время CPU или количество операций ввода-вывода).

Проведенный анализ производительности базы данных с таблицей User\_Account, содержащей большое количество строк, позволяет сделать следующие выводы. Первоначально, до создания индексов, запрос к таблице осуществлялся полным сканированием, что может привести к замедлению при работе с большим объемом данных.

Создание индексов на соответствующих столбцах таблицы существенно улучшило время выполнения запроса. Индексы позволяют эффективнее организовывать доступ к данным, снижая стоимость выполнения операций. Таким образом, база данных оказывается более подготовленной к обработке больших объемов данных, что является важным аспектом в условиях активной работы системы с множеством запросов и операций.

**6. Описание технологии и ее применения в базе данных**

**6.1 Шифрование данных в Oracle**

Для шифрования данных в Oracle существует пакет DBMS\_CRYPTO позволяющий использовать алгоритмы симметричного и асимметричного шифрования, хэширования и цифровой подписи. DBMS\_CRYPTO поддерживает алгоритмы шифрования, такие как AES (Advanced Encryption Standard), DES (Data Encryption Standard), 3DES (Triple Data Encryption Standard), RSA (Rivest-Shamir-Adleman) и другие. В данном курсовом проекте мы будем использовать алгоритм AES (Advanced Encryption Standard) для шифрования и дешифрования данных в базе данных Oracle. 256 AES является одним из наиболее распространенных и надежных алгоритмов симметричного шифрования, который широко применяется в различных сферах, включая информационную безопасность, защиту данных и конфиденциальность. AES основан на блочном шифровании, где данные разбиваются на фиксированные блоки и каждый блок шифруется независимо. Алгоритм AES поддерживает различные длины ключей, включая 128, 192 и 256 бит, что позволяет выбирать уровень безопасности и производительности в соответствии с требованиями проекта. Функция шифрования данных, используется, если в таблице нужно зашифровать конфиденциальные данные. Функция представлена на листинге 6.1

|  |
| --- |
| CREATE OR REPLACE FUNCTION encrypt(p\_plain\_text VARCHAR2, p\_salt VARCHAR2) RETURN RAW IS  encryption\_key RAW(256) := HEXTORAW('0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF');  encrypted\_raw RAW(2048); BEGIN  encrypted\_raw := DBMS\_CRYPTO.ENCRYPT(  src => UTL\_I18N.STRING\_TO\_RAW(p\_plain\_text || p\_salt, 'AL32UTF8'),  typ => DBMS\_CRYPTO.ENCRYPT\_AES256 + DBMS\_CRYPTO.CHAIN\_CBC + DBMS\_CRYPTO.PAD\_PKCS5,  key => encryption\_key  );  RETURN encrypted\_raw; END encrypt; / |

Листинг 6.1 – Функция для шифрования данных

Реализация функции дешифрования представлены в листинге 6.2.

|  |
| --- |
| CREATE OR REPLACE FUNCTION decrypt(p\_encrypted\_text RAW, p\_salt VARCHAR2) RETURN VARCHAR2 IS  encryption\_key RAW(256) := HEXTORAW('0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF');  decrypted\_raw RAW(2048); BEGIN  decrypted\_raw := DBMS\_CRYPTO.DECRYPT(  src => p\_encrypted\_text,  typ => DBMS\_CRYPTO.ENCRYPT\_AES256 + DBMS\_CRYPTO.CHAIN\_CBC + DBMS\_CRYPTO.PAD\_PKCS5,  key => encryption\_key  );  RETURN separate\_string(UTL\_I18N.RAW\_TO\_CHAR(decrypted\_raw, 'AL32UTF8'), p\_salt); END decrypt; |

Листинг 6.2 – Функция для дешифрования данных

**Приложение А**

Листинг создания таблиц

create table Currency (ID number(10) generated by default on null as identity,

NAME varchar2(100) not null,

CODE varchar2(3) not null,

constraint PK\_Currency primary key (ID)

);

create table App\_User (ID number(10) generated by default on null as identity,

LOGIN nvarchar2(255) not null,

PASSWORD nvarchar2(255) not null,

constraint User\_PK primary key (ID)

);

create table User\_Account (ID number(10) generated by default on null as identity,

NAME nvarchar2(50) not null,

CURRENCY varchar2(4) not null,

AMOUNT number(15,2) default 0.0,

USER\_ID number(10) not null,

constraint PK\_Account primary key (ID),

constraint FK\_User foreign key (USER\_ID) references App\_User (ID)

);

create table User\_Profile (ID number(10) generated by default on null as identity,

USER\_ID number(10) not null,

FIRST\_NAME nvarchar2(50) not null,

LAST\_NAME nvarchar2(50) not null,

MIDDLE\_NAME nvarchar2(50),

ADDRESS nvarchar2(255),

BIRTH\_DATE date,

EMAIL nvarchar2(100),

PASSPORT\_NUM nvarchar2(50) not null,

PHONE\_NUMBER nvarchar2(50),

constraint PK\_User\_Profile primary key (ID),

constraint FK\_User\_Profile\_User foreign key (USER\_ID) references App\_User (ID)

);

create table User\_Support\_Request (ID number(10) generated by default on null as identity,

TYPE nvarchar2(50),

CONTENT clob,

DATE\_TIME timestamp,

ACCOUNT\_ID number(10) not null,

constraint PK\_User\_Support\_Request primary key (ID),

constraint FK\_User\_Support\_Request\_Account foreign key (ACCOUNT\_ID) references User\_Account(ID)

);

create table Currency\_Exchange\_Rate (ID number(10) generated by default on null as identity,

SOURCE\_CURR\_ID number(10) not null,

TARGET\_CURR\_ID number(10) not null,

RATE number(15, 2) not null,

RATE\_DATE date not null,

constraint PK\_Currency\_Exchange\_Rate primary key (ID),

constraint FK\_Currency\_Exchange\_Rate\_Source foreign key (SOURCE\_CURR\_ID) references Currency(ID),

constraint FK\_Currency\_Exchange\_Rate\_Target foreign key (TARGET\_CURR\_ID) references Currency(ID)

);

create table Account\_Transaction (ID number(10) generated by default on null as identity,

ACCOUNT\_ID\_FROM number(10) not null,

ACCOUNT\_ID\_TO number(10) not null,

AMOUNT number(15, 2) not null,

DATE\_TIME timestamp not null,

constraint PK\_Transaction primary key (ID),

constraint FK\_Transaction\_Account\_From foreign key (ACCOUNT\_ID\_FROM) references User\_Account(ID),

constraint FK\_Transaction\_Account\_To foreign key (ACCOUNT\_ID\_TO) references User\_Account(ID)

);

create table Card (ID number(10) generated by default on null as identity,

TYPE nvarchar2(50) not null,

ACCOUNT\_ID number(10) not null,

CARD\_NAME nvarchar2(100),

DESCRIPTION nvarchar2(255),

STATUS nvarchar2(20),

constraint PK\_Card primary key (ID),

constraint FK\_Card\_Account foreign key (ACCOUNT\_ID) references User\_Account(ID)

);

create table Credit\_Type (ID number(10) generated by default on null as identity,

TYPE nvarchar2(50) not null,

CREDIT\_NAME nvarchar2(100) not null,

DESCRIPTION nvarchar2(255) not null,

constraint PK\_Credit\_Type primary key (ID)

);

create table Payment\_Service (ID number(10) generated by default on null as identity,

NAME nvarchar2(100) not null,

TYPE nvarchar2(50) not null ,

TRANSACTION\_ID number(10) not null,

constraint PK\_Payment\_Service primary key (ID),

constraint FK\_Payment\_Service\_Transaction foreign key (TRANSACTION\_ID) references Account\_Transaction(ID)

);

create table Credits (ID number(10) generated by default on null as identity,

CREDIT\_TYPE\_ID number(10) not null,

ACCOUNT\_ID number(10) not null,

AMOUNT number(15, 2) not null,

constraint PK\_Credits primary key (ID),

constraint FK\_Credits\_CreditType foreign key (CREDIT\_TYPE\_ID) references Credit\_Type(ID),

constraint FK\_Credits\_Account foreign key (ACCOUNT\_ID) references User\_Account(ID)

);

**Список используемых источников**

1. Oracle Database Documentation [Электронный ресурс]. – Электронные данные. – Режим доступа: https://docs.oracle.com/en/database/oracle/oracle-database/index.html. Дата доступа: 22.09.2024.
2. Oracle Text Reference 21c [Электронный ресурс]. – Электронные данные. – Режим доступа: https://docs.oracle.com/en/database/oracle/oracle-database/21/ccref/text-reference.pdf. Дата доступа: 22.09.2024.
3. Java Servlet Specification [Электронный ресурс]. – Электронные данные. – Режим доступа: https://javaee.github.io/servlet-spec/downloads/servlet-4.0/servlet-4\_0\_FINAL.pdf . Дата доступа: 22.09.2024.
4. Java EE – Java Servlets and JSPs [Электронный ресурс]. – Электронные данные. – Режим доступа: https://medium.com/javarevisited/java-ee-java-servlets-and-jsps-6ed1d2fca237. Дата доступа: 22.09.2024.