

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
**«Сибирский государственный университет науки и технологий  
имени академика М.Ф. Решетнева»**

Институт информатики и телекоммуникаций (ИИТК)

институт/факультет/подразделение

Кафедра безопасности информационных технологий (БИТ)

кафедра

**КУРСОВАЯ РАБОТА**

Безопасность систем баз данных

Дисциплина

Проектирование, разработка и обеспечение безопасности  
реляционной базы данных

Тема работы

Преподаватель

Обучающийся

КБ23-01

номер группы

подпись, дата

подпись, дата

В.Г. Жуков

инициалы, фамилия

И.М. Смоликов

инициалы, фамилия

Красноярск 2025

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Сибирский государственный университет науки и технологий  
имени академика М.Ф. Решетнева»

Институт информатики и телекоммуникаций (ИИТК)

институт/факультет/подразделение

Кафедра безопасности информационных технологий (БИТ)

кафедра

## ЗАДАНИЕ

На курсовую работу по дисциплине «Безопасность систем баз данных»

Обучающемуся Смоликову Ивану Михайловичу

фамилия, имя, отчество

Тема работы «Проектирование, разработка и обеспечение безопасности реляционной  
базы данных»

Срок сдачи студентом работы «\_\_» \_\_\_\_\_ 20\_\_ г.

Перечень вопросов, подлежащих разработке при написании теоретической части:

Описание внутримашинной информационной базы (логическая структура, физическая  
структура, организация ведения информационной базы). Описание немашинной  
информационной базы (логическая структура, физическая структура, организация  
ведения информационной базы).

Перечень вопросов, подлежащих разработке при написании практической части:

Разработка мер по физической защите базы данных. Разработка представлений.

Разработка триггеров. Разработка функций/хранимых процедур. Создание пользователей  
и ролей. Назначение прав субъектам доступа. Аутентификация клиентов при соединении  
с базой данных. Защита канала связи между СУБД и клиентом. Организация  
журналирования. Организация шифрования данных. Резервное копирование и  
восстановление данных.

Дата выдачи задания «\_\_» \_\_\_\_\_ 20\_\_ г.

Руководитель

подпись

инициалы и фамилия

Задание принял к исполнению

подпись

инициалы и фамилия обучающегося

«\_\_» \_\_\_\_\_ 20\_\_ г.

## СОДЕРЖАНИЕ

ВВЕДЕНИЕ .....	4
1 ОПИСАНИЕ ВНУТРИМАШИНОЙ ИНФОРМАЦИОННОЙ БАЗЫ.....	5
1.1 Логическая структура .....	5
1.2 Физическая структура.....	9
1.3 Организация ведения информационной базы .....	23
2 ОПИСАНИЕ ВНЕМАШИНОЙ ИНФОРМАЦИОННОЙ БАЗЫ.....	24
2.1 Логическая структура .....	24
2.2 Физическая структура.....	28
2.3 Организация ведения информационной базы .....	36
3 ОБЕСПЕЧЕНИЕ БЕЗОПАСНОСТИ БАЗЫ ДАННЫХ .....	37
3.1 Разработка мер по физической защите базы данных .....	37
3.2 Разработка представлений .....	38
3.3 Разработка триггеров .....	42
3.4 Разработка функций.....	49
3.5 Создание пользователей и ролей.....	62
3.6 Назначение прав субъектам доступа.....	63
3.7 Аутентификация клиентов при соединении с базой данных .....	71
3.8 Защита канала связи между СУБД и клиентом .....	72
3.9 Организация журналирования .....	73
3.10 Организация шифрования данных .....	75
3.11 Резервное копирование и восстановление данных.....	75
ЗАКЛЮЧЕНИЕ .....	78
БИБЛИОГРАФИЧЕСКИЙ СПИСОК .....	79
ПРИЛОЖЕНИЯ .....	80

## ВВЕДЕНИЕ

В современном цифровом обществе базы данных играют ключевую роль в обеспечении структурированного хранения, обработки и управления информацией. Развитие технологий хранения данных неразрывно связано с эволюцией автоматизированных информационных систем, которые обеспечивают эффективную работу в различных отраслях — от государственного управления до децентрализованных финансов. Одной из наиболее устойчивых и широко используемых парадигм организации данных является реляционная модель, предложенная Эдгаром Коддом. Она до сих пор лежит в основе большинства современных систем управления базами данных (СУБД), предоставляя логически целостный и гибкий подход к построению и сопровождению информационных систем.

В последние годы особое внимание исследователей и разработчиков сосредоточено на инфраструктурах, связанных с криптовалютами и блокчейн-технологиями. Учет криптовалютных транзакций, мониторинг состояния кошельков и интеграция с различными блокчейнами требуют высокой степени формализации и надежности хранения данных. Реляционные базы данных, несмотря на конкуренцию со стороны распределённых и *NoSQL*-решений, продолжают оставаться важным компонентом подобных систем за счёт своей зрелости, поддержки транзакционности и целостности данных.

С учетом вышесказанного, можно утверждать, что безопасность, согласованность и структурированность хранения информации в рамках криптовалютной экосистемы напрямую зависит от грамотно спроектированной и защищённой базы данных. Причем защита охватывает не только сами данные, но и архитектурные компоненты — от программной среды до используемого оборудования и административных процедур.

Цель данной курсовой работы — разработка и реализация базы данных для автоматизации учета криптовалютных кошельков и осуществляемых транзакций, с акцентом на обеспечение целостности данных и информационной безопасности.

Задачи, выполняемые в данной курсовой работе:

- разработать и описать внутримашинную информационную базу;
- разработать и описать немашинную информационную базу;
- реализовать меры по защите БД.

# 1 ОПИСАНИЕ ВНУТРИМАШИНОЙ ИНФОРМАЦИОННОЙ БАЗЫ

## 1.1 Логическая структура

Объектом моделирования является автоматизация процессов управления данными в системе учета криптовалютных кошельков, токенов и транзакций.

В результате исследования предметной области были выделены следующие типы сущностей:

- Статусы — тип сущности, определяющий общее состояние объектов системы (например, активный, неактивный, заблокирован);
- Статусы транзакций — тип сущности, описывающий этап выполнения транзакций (например, ожидается, подтверждена, неудачна);
- Блокчейны — тип сущности, содержащий информацию о различных блокчейн-сетях, таких как *Ethereum*, *Bitcoin* и др.;
- Пользователи — тип сущности, описывающий зарегистрированных пользователей системы, включая логин, email и статус;
- Кошельки — тип сущности, определяющий криптокошельки, принадлежащие пользователям;
- Токены — тип сущности, описывающий криптовалютные активы, включая их символ, название, блокчейн и адрес контракта;
- Балансы — тип сущности, отражающий количество конкретного токена на конкретном кошельке;
- Обменники — тип сущности, содержащий информацию о криптовалютных биржах и их статусах;
- Транзакция с обменниками — тип сущности, фиксирующий взаимодействие пользователя с биржей (например, ввод или вывод средств), с указанием адреса получателя, суммы, комиссии и статуса;
- Транзакция между внутренними кошельками — слабый тип сущности, отображающий перевод токенов между кошельками внутри системы.

Взаимосвязь перечисленных типов сущностей в рамках высокоуровневой концептуальной модели данных «Сущность–Связь» (ER-модель) представлена в Приложении А. Рассмотрим указанные сущности более подробно.

### 1.1.1 Тип сущности «Статусы»

Тип сущности «Статусы» в рамках информационной базы описывается следующими атрибутами (свойствами сущности), определяющими экземпляры типа сущности «Статусы», которые могут быть идентифицированы уникальным образом:

- *id* — уникальный идентификатор записи статуса;

- *status\_name* — название статуса.

### 1.1.2 Тип сущности «Статусы транзакций»

Тип сущности «Статусы транзакций» в рамках информационной базы описывается следующими атрибутами (свойствами сущности), определяющими экземпляры типа сущности «Статусы транзакций», которые могут быть идентифицированы уникальным образом:

- *id* — уникальный идентификатор записи статуса транзакции;
- *transaction\_status\_name* — название статуса транзакции.

### 1.1.3 Тип сущности «Блокчейны»

Тип сущности «Блокчейны» в рамках информационной базы описывается следующими атрибутами (свойствами сущности), определяющими экземпляры типа сущности «Блокчейны», которые могут быть идентифицированы уникальным образом:

- *id* — уникальный идентификатор блокчейна;
- *symbol* — краткое обозначение блокчейна;
- *blockchain\_name* — полное название блокчейна.

### 1.1.4 Тип сущности «Пользователи»

Тип сущности «Пользователи» в рамках информационной базы описывается следующими атрибутами (свойствами сущности), определяющими экземпляры типа сущности «Пользователи», которые могут быть идентифицированы уникальным образом:

- *username* — уникальное имя пользователя;
- *email* — уникальный адрес электронной почты;
- *status\_id* — ссылка на статус пользователя;
- *updated\_at* — дата и время последнего обновления записи;
- *created\_at* — дата и время создания пользователя.

### 1.1.5 Тип сущности «Кошельки»

Тип сущности «Кошельки» в рамках информационной базы описывается следующими атрибутами (свойствами сущности), определяющими экземпляры типа сущности «Кошельки», которые могут быть идентифицированы уникальным образом:

- *address* — уникальный адрес кошелька;
- *user\_username* — имя пользователя, которому принадлежит кошелек;
- *status\_id* — статус кошелька;

- *updated\_at* — дата и время последнего обновления;
- *created\_at* — дата и время создания записи.

#### **1.1.6 Тип сущности «Токены»**

Тип сущности «Токены» в рамках информационной базы описывается следующими атрибутами (свойствами сущности), определяющими экземпляры типа сущности «Токены», которые могут быть идентифицированы уникальным образом:

- *id* — уникальный идентификатор токена;
- *blockchain\_id* — идентификатор блокчейна, к которому принадлежит токен;
- *symbol* — краткое название токена;
- *token\_name* — полное название токена;
- *contract\_address* — адрес смарт-контракта токена.

#### **1.1.7 Тип сущности «Балансы»**

Тип сущности «Балансы» в рамках информационной базы описывается следующими атрибутами (свойствами сущности), определяющими экземпляры типа сущности «Балансы», которые могут быть идентифицированы уникальным образом:

- *token\_id* — идентификатор токена;
- *wallet\_address* — адрес кошелька;
- *amount* — количество токенов;
- *updated\_at* — дата последнего обновления баланса;
- *created\_at* — дата создания записи.

#### **1.1.8 Тип сущности «Обменники»**

Тип сущности «Обменники» в рамках информационной базы описывается следующими атрибутами (свойствами сущности), определяющими экземпляры типа сущности «Обменники», которые могут быть идентифицированы уникальным образом:

- *id* — уникальный идентификатор обменника;
- *exchange\_name* — название обменника;
- *url* — веб-сайт обменника;
- *status\_id* — статус обменника;
- *updated\_at* — дата обновления информации;
- *created\_at* — дата добавления биржи в систему.

### 1.1.9 Тип сущности «Транзакции с обменниками»

Тип сущности «Транзакции с обменниками» в рамках информационной базы описывается следующими атрибутами (свойствами сущности), определяющими экземпляры типа сущности «Транзакции с обменниками», которые могут быть идентифицированы уникальным образом:

- *hash* — уникальный хэш транзакции;
- *wallet\_address* — адрес отправителя (кошелька);
- *exchange\_id* — идентификатор обменника;
- *recipient\_address* — адрес получателя (внешний или внутренний);
- *transaction\_status\_id* — статус транзакции;
- *token\_id* — идентификатор токена;
- *amount* — сумма перевода;
- *fee* — комиссия;
- *updated\_at* — дата и время последнего изменения;
- *created\_at* — дата создания записи.

### 1.1.10 Тип сущности «Транзакции между внутренними кошельками»

Тип сущности «Транзакции между внутренними кошельками» в рамках информационной базы описывается следующими атрибутами (свойствами сущности), определяющими экземпляры типа сущности «Транзакции между внутренними кошельками», которые могут быть идентифицированы уникальным образом:

- *hash* — уникальный хэш транзакции;
- *sender\_address* — адрес отправителя;
- *recipient\_address* — адрес получателя;
- *transaction\_status\_id* — статус транзакции;
- *token\_id* — идентификатор токена;
- *amount* — сумма перевода;
- *fee* — комиссия за транзакцию;
- *updated\_at* — дата последнего обновления;
- *created\_at* — дата создания транзакции.



### 1.1.12 Логические связи между типами сущностей

Описание логических связей между типами сущностей и их характеристика приведены ниже (таблица 1).

Таблица 1 – Связи между типами сущностей

№	Главная таблица	Подчиненная таблица	Тип связи
1	Статусы	Пользователи	1:M
2	Статусы	Кошельки	1:M
3	Статусы	Обменники	1:M
4	Статусы транзакций	Транзакции с обменниками	1:M
5	Статусы транзакций	Транзакции между внутренними кошельками	1:M
6	Блокчейны	Токены	1:M
7	Пользователи	Кошельки	1:M
8	Кошельки	Балансы	1:M
9	Токены	Балансы	1:M
10	Обменники	Транзакции с обменниками	1:M
11	Кошельки	Транзакции с обменниками	1:M
12	Кошельки	Транзакции между внутренними кошельками	1:M
13	Кошельки	Транзакции между внутренними кошельками	1:M
14	Токены	Транзакции с обменниками	1:M
15	Токены	Транзакции между внутренними кошельками	1:M

### 1.1.13 Логические связи между типами сущностей

Логическая структура внутримашинной информационной базы приведена в Приложении Б в виде высокоуровневой концептуальной модели данных «Сущность-Связь».

## 1.2 Физическая структура

Высокоуровневая концептуальная модель данных «Сущность-Связь», приведена в Приложении Б, с помощью методологии построения реляционных структур *IDEF1X* (Приложение В)

В состав информационной базы входят следующие таблицы базы данных (таблица 2).

Таблица 2 – Таблицы базы данных

№	Тип сущности	Наименование таблицы в базе данных
1	Статусы	<i>Statuses</i>
2	Статусы транзакций	<i>Transaction_statuses</i>
3	Блокчейны	<i>Blockchains</i>
4	Пользователи	<i>Users</i>
5	Кошельки	<i>Wallets</i>
6	Токены	<i>Tokens</i>
7	Балансы	<i>Balances</i>
8	Обменники	<i>Exchanges</i>
9	Транзакции с обменниками	<i>Exchange_transactions</i>
10	Транзакции между внутренними кошельками	<i>Internal_transactions</i>

Была рассмотрена каждая таблица базы данных и приведено описание наименования атрибутов, наименования атрибутов в базе данных, типов атрибутов и длину.

### 1.2.1 Таблица «Статусы»

С целью устранения логических аномалий, дублирования данных и повышения степени структурированности базы данных была проведена нормализация отношения, описывающего сущность «Статусы».

Приведение к первой нормальной форме (1НФ):

Отношение «Статусы» удовлетворяет требованиям первой нормальной формы, так как:

- каждый атрибут содержит исключительно атомарные (неделимые) значения;
- в структуре отсутствуют повторяющиеся группы и массивы;
- значения в пределах одного столбца однородны по типу данных;
- каждый атрибут имеет уникальное имя;
- в отношении отсутствуют полностью идентичные строки (кортежи);
- порядок следования строк и столбцов не влияет на логическую интерпретацию данных.

Таким образом, отношение соответствует критериям 1НФ и допускает корректную реализацию в реляционной модели данных.

Приведение ко второй нормальной форме (2НФ):

Поскольку в рассматриваемом отношении используется простой (несоставной) первичный ключ, условие отсутствия частичных функциональных зависимостей выполняется автоматически. Все не ключевые атрибуты находятся в полной функциональной зависимости от первичного ключа.

Следовательно, отношение «Статусы» удовлетворяет требованиям второй нормальной формы.

Приведение к третьей нормальной форме (3НФ):

Отношение «Статусы» также соответствует условиям третьей нормальной формы, поскольку:

- оно уже приведено ко второй нормальной форме;
- в нем отсутствуют транзитивные зависимости между не ключевыми атрибутами, то есть каждый не ключевой атрибут функционально зависит исключительно от первичного ключа и не зависит от других не ключевых атрибутов.

Таким образом, отношение «Статусы» приведено к третьей нормальной форме (3НФ). Это гарантирует логическую непротиворечивость структуры таблицы, минимизирует дублирование информации и обеспечивает устойчивость к основным видам аномалий при модификации данных.

Таблица 3 – Описание таблицы «Статусы»

№	Наименование атрибута	Наименование в базе данных	Тип атрибута	Длина, байт	NULL/NOT NULL
1	Код	<i>Id (PK)</i>	<i>Smallserial</i>	2	<i>NOT NULL</i>
2	Название	<i>Status_name</i>	<i>Varchar (30)</i>	30	<i>NOT NULL</i>

### 1.2.2 Таблица «Статусы транзакций»

С целью устранения логических аномалий, дублирования данных и повышения степени структурированности базы данных была проведена нормализация отношения, описывающего сущность «Статусы транзакций».

Приведение к первой нормальной форме (1НФ):

Отношение «Статусы транзакций» удовлетворяет требованиям первой нормальной формы, так как:

- каждый атрибут содержит исключительно атомарные (неделимые) значения;
- в структуре отсутствуют повторяющиеся группы и массивы;
- значения в пределах одного столбца однородны по типу данных;
- каждый атрибут имеет уникальное имя;
- в отношении отсутствуют полностью идентичные строки (кортежи);
- порядок следования строк и столбцов не влияет на логическую интерпретацию данных.

Таким образом, отношение соответствует критериям 1НФ и допускает корректную реализацию в реляционной модели данных.

Приведение ко второй нормальной форме (2НФ):

Поскольку в рассматриваемом отношении используется простой (несоставной) первичный ключ, условие отсутствия частичных функциональных зависимостей выполняется автоматически. Все не ключевые атрибуты находятся в полной функциональной зависимости от первичного ключа.

Следовательно, отношение «Статусы транзакций» удовлетворяет требованиям второй нормальной формы.

Приведение к третьей нормальной форме (3НФ):

Отношение «Статусы транзакций» также соответствует условиям третьей нормальной формы, поскольку:

- оно уже приведено ко второй нормальной форме;
- в нем отсутствуют транзитивные зависимости между не ключевыми атрибутами, то есть каждый не ключевой атрибут функционально зависит исключительно от первичного ключа и не зависит от других не ключевых атрибутов.

Таким образом, отношение «Статусы транзакций» приведено к третьей нормальной форме (3НФ). Это гарантирует логическую непротиворечивость структуры таблицы, минимизирует дублирование информации и обеспечивает устойчивость к основным видам аномалий при модификации данных.

Таблица 4 – Описание таблицы «Статусы транзакций»

№	Наименование атрибута	Наименование в базе данных	Тип атрибута	Длина, байт	NULL/NOT NULL
1	Код	<i>Id (PK)</i>	<i>Smallserial</i>	2	<i>NOT NULL</i>
2	Название	<i>Status_name</i>	<i>Varchar (30)</i>	30	<i>NOT NULL</i>

### 1.2.3 Таблица «Блокчейны»

С целью устранения логических аномалий, дублирования данных и повышения степени структурированности базы данных была проведена нормализация отношения, описывающего сущность «Блокчейны».

Приведение к первой нормальной форме (1НФ):

Отношение «Блокчейны» удовлетворяет требованиям первой нормальной формы, так как:

- каждый атрибут содержит исключительно атомарные (неделимые) значения;
- в структуре отсутствуют повторяющиеся группы и массивы;
- значения в пределах одного столбца однородны по типу данных;
- каждый атрибут имеет уникальное имя;
- в отношении отсутствуют полностью идентичные строки (кортежи);
- порядок следования строк и столбцов не влияет на логическую интерпретацию данных.

Таким образом, отношение соответствует критериям 1НФ и допускает корректную реализацию в реляционной модели данных.

Приведение ко второй нормальной форме (2НФ):

Поскольку в рассматриваемом отношении используется простой (несоставной) первичный ключ, условие отсутствия частичных функциональных зависимостей выполняется автоматически. Все не ключевые атрибуты находятся в полной функциональной зависимости от первичного ключа.

Следовательно, отношение «Блокчейны» удовлетворяет требованиям второй нормальной формы.

Приведение к третьей нормальной форме (3НФ):

Отношение «Блокчейны» также соответствует условиям третьей нормальной формы, поскольку:

- оно уже приведено ко второй нормальной форме;
- в нем отсутствуют транзитивные зависимости между не ключевыми атрибутами, то есть каждый не ключевой атрибут функционально зависит исключительно от первичного ключа и не зависит от других не ключевых атрибутов.

Таким образом, отношение «Блокчейны» приведено к третьей нормальной форме (3НФ). Это гарантирует логическую непротиворечивость структуры таблицы, минимизирует дублирование информации и обеспечивает устойчивость к основным видам аномалий при модификации данных.

Таблица 5 – Описание таблицы «Блокчейны»

№	Наименование атрибута	Наименование в базе данных	Тип атрибута	Длина, байт	NULL/NOT NULL
1	Код	<i>Id (PK)</i>	<i>Smallserial</i>	2	<i>NOT NULL</i>
2	Символ	<i>Symbol</i>	<i>Varchar (10)</i>	10	<i>NOT NULL</i>
3	Название	<i>Blockchain_name</i>	<i>Varchar (30)</i>	30	<i>NOT NULL</i>

#### 1.2.4 Таблица «Пользователи»

С целью устранения логических аномалий, дублирования данных и повышения степени структурированности базы данных была проведена нормализация отношения, описывающего сущность «Пользователи».

Приведение к первой нормальной форме (1НФ):

Отношение «Пользователи» удовлетворяет требованиям первой нормальной формы, так как:

- каждый атрибут содержит исключительно атомарные (неделимые) значения;
- в структуре отсутствуют повторяющиеся группы и массивы;
- значения в пределах одного столбца однородны по типу данных;
- каждый атрибут имеет уникальное имя;
- в отношении отсутствуют полностью идентичные строки (кортежи);
- порядок следования строк и столбцов не влияет на логическую интерпретацию данных.

Таким образом, отношение соответствует критериям 1НФ и допускает корректную реализацию в реляционной модели данных.

Приведение ко второй нормальной форме (2НФ):

Поскольку в рассматриваемом отношении используется простой (несоставной) первичный ключ, условие отсутствия частичных функциональных зависимостей выполняется автоматически. Все не ключевые

атрибуты находятся в полной функциональной зависимости от первичного ключа.

Следовательно, отношение «Пользователи» удовлетворяет требованиям второй нормальной формы.

Приведение к третьей нормальной форме (3НФ):

Отношение «Пользователи» также соответствует условиям третьей нормальной формы, поскольку:

- оно уже приведено ко второй нормальной форме;
- в нем отсутствуют транзитивные зависимости между не ключевыми атрибутами, то есть каждый не ключевой атрибут функционально зависит исключительно от первичного ключа и не зависит от других не ключевых атрибутов.

Таким образом, отношение «Пользователи» приведено к третьей нормальной форме (3НФ). Это гарантирует логическую непротиворечивость структуры таблицы, минимизирует дублирование информации и обеспечивает устойчивость к основным видам аномалий при модификации данных.

Таблица 6 – Описание таблицы «Пользователи»

№	Наименование атрибута	Наименование в базе данных	Тип атрибута	Длина, байт	NULL/NOT NULL
1	Имя	<i>Username (PK)</i>	<i>Varchar (30)</i>	30	<i>NOT NULL</i>
2	Электронная почта	<i>Email</i>	<i>Varchar (50)</i>	50	<i>NOT NULL</i>
3	Статус_код	<i>Status_id</i>	<i>Smallint</i>	2	<i>NOT NULL</i>
4	Дата_обновления	<i>Updated_at</i>	<i>Timestamp</i>	8	<i>NULL</i>
5	Дата_создания	<i>Created_at</i>	<i>Timestamp</i>	8	<i>NOT NULL</i>

### 1.2.5 Таблица «Кошельки»

С целью устранения логических аномалий, дублирования данных и повышения степени структурированности базы данных была проведена нормализация отношения, описывающего сущность «Кошельки».

Приведение к первой нормальной форме (1НФ):

Отношение «Кошельки» удовлетворяет требованиям первой нормальной формы, так как:

- каждый атрибут содержит исключительно атомарные (неделимые) значения;
- в структуре отсутствуют повторяющиеся группы и массивы;
- значения в пределах одного столбца однородны по типу данных;
- каждый атрибут имеет уникальное имя;
- в отношении отсутствуют полностью идентичные строки (кортежи);
- порядок следования строк и столбцов не влияет на логическую интерпретацию данных.

Таким образом, отношение соответствует критериям 1НФ и допускает корректную реализацию в реляционной модели данных.

Приведение ко второй нормальной форме (2НФ):

Поскольку в рассматриваемом отношении используется простой (несоставной) первичный ключ, условие отсутствия частичных функциональных зависимостей выполняется автоматически. Все не ключевые атрибуты находятся в полной функциональной зависимости от первичного ключа.

Следовательно, отношение «Кошельки» удовлетворяет требованиям второй нормальной формы.

Приведение к третьей нормальной форме (3НФ):

Отношение «Кошельки» также соответствует условиям третьей нормальной формы, поскольку:

- оно уже приведено ко второй нормальной форме;
- в нем отсутствуют транзитивные зависимости между не ключевыми атрибутами, то есть каждый не ключевой атрибут функционально зависит исключительно от первичного ключа и не зависит от других не ключевых атрибутов.

Таким образом, отношение «Кошельки» приведено к третьей нормальной форме (3НФ). Это гарантирует логическую непротиворечивость структуры таблицы, минимизирует дублирование информации и обеспечивает устойчивость к основным видам аномалий при модификации данных.

Таблица 6 – Описание таблицы «Кошельки»

№	Наименование атрибута	Наименование в базе данных	Тип атрибута	Длина, байт	NULL/NOT NULL
1	Адрес	<i>Address (PK)</i>	<i>Varchar (70)</i>	70	<i>NOT NULL</i>
2	Пользователь_имя	<i>User_username</i>	<i>Varchar (30)</i>	30	<i>NOT NULL</i>
3	Статус_код	<i>Status_id</i>	<i>Smallint</i>	2	<i>NOT NULL</i>
4	Дата_обновления	<i>Updated_at</i>	<i>Timestamp</i>	8	<i>NULL</i>
5	Дата_создания	<i>Created_at</i>	<i>Timestamp</i>	8	<i>NOT NULL</i>

### 1.2.6 Таблица «Токены»

С целью устранения логических аномалий, дублирования данных и повышения степени структурированности базы данных была проведена нормализация отношения, описывающего сущность «Токены».

Приведение к первой нормальной форме (1НФ):

Отношение «Токены» удовлетворяет требованиям первой нормальной формы, так как:

- каждый атрибут содержит исключительно атомарные (неделимые) значения;
- в структуре отсутствуют повторяющиеся группы и массивы;
- значения в пределах одного столбца однородны по типу данных;
- каждый атрибут имеет уникальное имя;
- в отношении отсутствуют полностью идентичные строки (кортежи);

– порядок следования строк и столбцов не влияет на логическую интерпретацию данных.

Таким образом, отношение соответствует критериям 1НФ и допускает корректную реализацию в реляционной модели данных.

Приведение ко второй нормальной форме (2НФ):

Поскольку в рассматриваемом отношении используется простой (несоставной) первичный ключ, условие отсутствия частичных функциональных зависимостей выполняется автоматически. Все не ключевые атрибуты находятся в полной функциональной зависимости от первичного ключа.

Следовательно, отношение «Токены» удовлетворяет требованиям второй нормальной формы.

Приведение к третьей нормальной форме (3НФ):

Отношение «Токены» также соответствует условиям третьей нормальной формы, поскольку:

- оно уже приведено ко второй нормальной форме;
- в нем отсутствуют транзитивные зависимости между не ключевыми атрибутами, то есть каждый не ключевой атрибут функционально зависит исключительно от первичного ключа и не зависит от других не ключевых атрибутов.

Таким образом, отношение «Токены» приведено к третьей нормальной форме (3НФ). Это гарантирует логическую непротиворечивость структуры таблицы, минимизирует дублирование информации и обеспечивает устойчивость к основным видам аномалий при модификации данных.

Таблица 7 – Описание таблицы «Токены»

№	Наименование атрибута	Наименование в базе данных	Тип атрибута	Длина, байт	NULL/NOT NULL
1	Код	<i>Id (PK)</i>	<i>Smallserial</i>	2	<i>NOT NULL</i>
2	Блокчейн_код	<i>Blockchain_id</i>	<i>Smallint</i>	2	<i>NOT NULL</i>
3	Символ	<i>Symbol</i>	<i>Varchar (10)</i>	10	<i>NOT NULL</i>
4	Название	<i>Token_name</i>	<i>Varchar (30)</i>	30	<i>NOT NULL</i>
5	Контракт адрес	<i>Contract_address</i>	<i>Varchar (70)</i>	70	<i>NOT NULL</i>

### 1.2.7 Таблица «Балансы»

С целью устранения логических аномалий, дублирования данных и повышения степени структурированности базы данных была проведена нормализация отношения, описывающего сущность «Балансы».

Приведение к первой нормальной форме (1НФ):

Отношение «Балансы» удовлетворяет требованиям первой нормальной формы, так как:

- каждый атрибут содержит исключительно атомарные (неделимые) значения;



- в структуре отсутствуют повторяющиеся группы и массивы;
- значения в пределах одного столбца однородны по типу данных;
- каждый атрибут имеет уникальное имя;
- в отношении отсутствуют полностью идентичные строки (кортежи);
- порядок следования строк и столбцов не влияет на логическую интерпретацию данных.

Таким образом, отношение соответствует критериям 1НФ и допускает корректную реализацию в реляционной модели данных.

Приведение ко второй нормальной форме (2НФ):

Поскольку в рассматриваемом отношении используется простой (несоставной) первичный ключ, условие отсутствия частичных функциональных зависимостей выполняется автоматически. Все не ключевые атрибуты находятся в полной функциональной зависимости от первичного ключа.

Следовательно, отношение «Балансы» удовлетворяет требованиям второй нормальной формы.

Приведение к третьей нормальной форме (3НФ):

Отношение «Балансы» также соответствует условиям третьей нормальной формы, поскольку:

- оно уже приведено ко второй нормальной форме;
- в нем отсутствуют транзитивные зависимости между не ключевыми атрибутами, то есть каждый не ключевой атрибут функционально зависит исключительно от первичного ключа и не зависит от других не ключевых атрибутов.

Таким образом, отношение «Балансы» приведено к третьей нормальной форме (3НФ). Это гарантирует логическую непротиворечивость структуры таблицы, минимизирует дублирование информации и обеспечивает устойчивость к основным видам аномалий при модификации данных.

Таблица 8 – Описание таблицы «Балансы»

№	Наименование атрибута	Наименование в базе данных	Тип атрибута	Длина, байт	NULL/NOT NULL
1	Токен_код	<i>Token_id (PK)</i>	<i>Smallint</i>	2	<i>NOT NULL</i>
2	Кошелек_адрес	<i>Wallet_address (PK)</i>	<i>Varchar (70)</i>	70	<i>NOT NULL</i>
3	Сумма	<i>Amount</i>	<i>Numeric (20,8)</i>	12	<i>NOT NULL</i>
4	Дата_обновления	<i>Updated_at</i>	<i>Timestamp</i>	8	<i>NULL</i>
5	Дата_создания	<i>Created_at</i>	<i>Timestamp</i>	8	<i>NOT NULL</i>

### 1.2.8 Таблица «Обменники»

С целью устранения логических аномалий, дублирования данных и повышения степени структурированности базы данных была проведена нормализация отношения, описывающего сущность «Обменники».

Приведение к первой нормальной форме (1НФ):

Отношение «Обменники» удовлетворяет требованиям первой нормальной формы, так как:

- каждый атрибут содержит исключительно атомарные (неделимые) значения;
- в структуре отсутствуют повторяющиеся группы и массивы;
- значения в пределах одного столбца однородны по типу данных;
- каждый атрибут имеет уникальное имя;
- в отношении отсутствуют полностью идентичные строки (кортежи);
- порядок следования строк и столбцов не влияет на логическую интерпретацию данных.

Таким образом, отношение соответствует критериям 1НФ и допускает корректную реализацию в реляционной модели данных.

Приведение ко второй нормальной форме (2НФ):

Поскольку в рассматриваемом отношении используется простой (несоставной) первичный ключ, условие отсутствия частичных функциональных зависимостей выполняется автоматически. Все не ключевые атрибуты находятся в полной функциональной зависимости от первичного ключа.

Следовательно, отношение «Обменники» удовлетворяет требованиям второй нормальной формы.

Приведение к третьей нормальной форме (3НФ):

Отношение «» также соответствует условиям третьей нормальной формы, поскольку:

- оно уже приведено ко второй нормальной форме;
- в нем отсутствуют транзитивные зависимости между не ключевыми атрибутами, то есть каждый не ключевой атрибут функционально зависит исключительно от первичного ключа и не зависит от других не ключевых атрибутов.

Таким образом, отношение «Обменники» приведено к третьей нормальной форме (3НФ). Это гарантирует логическую непротиворечивость структуры таблицы, минимизирует дублирование информации и обеспечивает устойчивость к основным видам аномалий при модификации данных.

Таблица 9 – Описание таблицы «Обменники»

№	Наименование атрибута	Наименование в базе данных	Тип атрибута	Длина, байт	NULL/NOT NULL
1	Код	<i>Id (PK)</i>	<i>Smallserial</i>	2	<i>NOT NULL</i>
2	Название	<i>Exchange_name</i>	<i>Varchar (50)</i>	50	<i>NOT NULL</i>
3	Ссылка	<i>Url</i>	<i>Varchar (100)</i>	100	<i>NOT NULL</i>
4	Статус_код	<i>Status_id</i>	<i>Smallint</i>	2	<i>NOT NULL</i>
5	Дата_обновления	<i>Updated_at</i>	<i>Timestamp</i>	8	<i>NULL</i>
6	Дата_создания	<i>Created_at</i>	<i>Timestamp</i>	8	<i>NOT NULL</i>

### 1.2.9 Таблица «Транзакции с обменниками»

С целью устранения логических аномалий, дублирования данных и повышения степени структурированности базы данных была проведена нормализация отношения, описывающего сущность «Транзакции с обменниками».

Приведение к первой нормальной форме (1НФ):

Отношение «Транзакции с обменниками» удовлетворяет требованиям первой нормальной формы, так как:

- каждый атрибут содержит исключительно атомарные (неделимые) значения;
- в структуре отсутствуют повторяющиеся группы и массивы;
- значения в пределах одного столбца однородны по типу данных;
- каждый атрибут имеет уникальное имя;
- в отношении отсутствуют полностью идентичные строки (кортежи);
- порядок следования строк и столбцов не влияет на логическую интерпретацию данных.

Таким образом, отношение соответствует критериям 1НФ и допускает корректную реализацию в реляционной модели данных.

Приведение ко второй нормальной форме (2НФ):

Поскольку в рассматриваемом отношении используется простой (несоставной) первичный ключ, условие отсутствия частичных функциональных зависимостей выполняется автоматически. Все не ключевые атрибуты находятся в полной функциональной зависимости от первичного ключа.

Следовательно, отношение «Транзакции с обменниками» удовлетворяет требованиям второй нормальной формы.

Приведение к третьей нормальной форме (3НФ):

Отношение «Транзакции с обменниками» также соответствует условиям третьей нормальной формы, поскольку:

- оно уже приведено ко второй нормальной форме;
- в нем отсутствуют транзитивные зависимости между не ключевыми атрибутами, то есть каждый не ключевой атрибут функционально зависит

исключительно от первичного ключа и не зависит от других не ключевых атрибутов.

Таким образом, отношение «Транзакции с обменниками» приведено к третьей нормальной форме (3НФ). Это гарантирует логическую непротиворечивость структуры таблицы, минимизирует дублирование информации и обеспечивает устойчивость к основным видам аномалий при модификации данных.

Таблица 10 – Описание таблицы «Транзакции с обменниками»

№	Наименование атрибута	Наименование в базе данных	Тип атрибута	Длина, байт	NULL/NOT NULL
1	Хэш	<i>Hash (PK)</i>	<i>Varchar (70)</i>	70	<i>NOT NULL</i>
2	Кошелек_адрес	<i>Wallet_address</i>	<i>Varchar (70)</i>	70	<i>NOT NULL</i>
3	Обменник_код	<i>Exchange_id</i>	<i>Smallint</i>	2	<i>NOT NULL</i>
4	Адрес_получателя	<i>Recipient_address</i>	<i>Varchar (70)</i>	70	<i>NOT NULL</i>
5	Статус_транзакции_код	<i>Transaction_status_id</i>	<i>Smallint</i>	2	<i>NOT NULL</i>
6	Токен_код	<i>Token_id</i>	<i>Smallint</i>	2	<i>NOT NULL</i>
7	Сумма	<i>Amount</i>	<i>Numeric (20,8)</i>	12	<i>NOT NULL</i>
8	Комиссия	<i>Fee</i>	<i>Numeric (20,8)</i>	12	<i>NOT NULL</i>
9	Дата_создания	<i>Created_at</i>	<i>Timestamp</i>	8	<i>NOT NULL</i>

#### 1.2.10 Таблица «Транзакции между внутренними кошельками»

С целью устранения логических аномалий, дублирования данных и повышения степени структурированности базы данных была проведена нормализация отношения, описывающего сущность «Транзакции между внутренними кошельками».

Приведение к первой нормальной форме (1НФ):

Отношение «Транзакции между внутренними кошельками» удовлетворяет требованиям первой нормальной формы, так как:

- каждый атрибут содержит исключительно атомарные (неделимые) значения;
- в структуре отсутствуют повторяющиеся группы и массивы;
- значения в пределах одного столбца однородны по типу данных;
- каждый атрибут имеет уникальное имя;
- в отношении отсутствуют полностью идентичные строки (кортежи);
- порядок следования строк и столбцов не влияет на логическую интерпретацию данных.

Таким образом, отношение соответствует критериям 1НФ и допускает корректную реализацию в реляционной модели данных.

Приведение ко второй нормальной форме (2НФ):

Поскольку в рассматриваемом отношении используется простой (несоставной) первичный ключ, условие отсутствия частичных функциональных зависимостей выполняется автоматически. Все не ключевые атрибуты находятся в полной функциональной зависимости от первичного ключа.

Следовательно, отношение «Транзакции между внутренними кошельками» удовлетворяет требованиям второй нормальной формы.

Приведение к третьей нормальной форме (3НФ):

Отношение «Транзакции между внутренними кошельками» также соответствует условиям третьей нормальной формы, поскольку:

- оно уже приведено ко второй нормальной форме;
- в нем отсутствуют транзитивные зависимости между не ключевыми атрибутами, то есть каждый не ключевой атрибут функционально зависит исключительно от первичного ключа и не зависит от других не ключевых атрибутов.

Таким образом, отношение «Транзакции между внутренними кошельками» приведено к третьей нормальной форме (3НФ). Это гарантирует логическую непротиворечивость структуры таблицы, минимизирует дублирование информации и обеспечивает устойчивость к основным видам аномалий при модификации данных.

Таблица 11 – Описание таблицы «Транзакции между внутренними кошельками»

№	Наименование атрибута	Наименование в базе данных	Тип атрибута	Длина, байт	NULL/NOT NULL
1	Хэш	<i>Hash (PK)</i>	<i>Varchar (70)</i>	70	<i>NOT NULL</i>
2	Адрес_отправителя	<i>Sender_address</i>	<i>Varchar (70)</i>	70	<i>NOT NULL</i>
3	Адрес_получателя	<i>Recipient_address</i>	<i>Varchar (70)</i>	70	<i>NOT NULL</i>
4	Статус_транзакции_код	<i>Transaction_status_id</i>	<i>Smallint</i>	2	<i>NOT NULL</i>
5	Токен_код	<i>Token_id</i>	<i>Smallint</i>	2	<i>NOT NULL</i>
6	Сумма	<i>Amount</i>	<i>Numeric (20,8)</i>	12	<i>NOT NULL</i>
7	Комиссия	<i>Fee</i>	<i>Numeric (20,8)</i>	12	<i>NOT NULL</i>
8	Дата_создания	<i>Created_at</i>	<i>Timestamp</i>	8	<i>NOT NULL</i>

### 1.2.11 Физические связи между таблицами

Описание физических связей между таблицами и их характеристика представлена в таблице 12.

Таблица 12 – Связи между типами сущностей

№	Главная таблица	Подчиненная таблица	Тип связи	Действие ссылочной целостности	
				Обновление	Удаление
1	Статусы	Пользователи	1:M	<i>RESTRICT</i>	<i>RESTRICT</i>
2	Статусы	Кошельки	1:M	<i>RESTRICT</i>	<i>RESTRICT</i>
3	Статусы	Обменники	1:M	<i>RESTRICT</i>	<i>RESTRICT</i>
4	Статусы транзакций	Транзакции с обменниками	1:M	<i>RESTRICT</i>	<i>RESTRICT</i>
5	Статусы транзакций	Транзакции между внутренними кошельками	1:M	<i>RESTRICT</i>	<i>RESTRICT</i>
6	Блокчейны	Токены	1:M	<i>RESTRICT</i>	<i>RESTRICT</i>
7	Пользователи	Кошельки	1:M	<i>RESTRICT</i>	<i>CASCADE</i>
8	Кошельки	Балансы	1:M	<i>RESTRICT</i>	<i>CASCADE</i>
9	Токены	Балансы	1:M	<i>RESTRICT</i>	<i>RESTRICT</i>
10	Обменники	Транзакции с обменниками	1:M	<i>RESTRICT</i>	<i>RESTRICT</i>
11	Кошельки	Транзакции с обменниками	1:M	<i>RESTRICT</i>	<i>RESTRICT</i>
12	Кошельки	Транзакции между внутренними кошельками	1:M	<i>RESTRICT</i>	<i>RESTRICT</i>
13	Кошельки	Транзакции между внутренними кошельками	1:M	<i>RESTRICT</i>	<i>RESTRICT</i>
14	Токены	Транзакции с обменниками	1:M	<i>RESTRICT</i>	<i>RESTRICT</i>
15	Токены	Транзакции между внутренними кошельками	1:M	<i>RESTRICT</i>	<i>RESTRICT</i>

### **1.2.12 Физическая структура внутримашинной информационной базы**

Физическая структура внутримашинной информационной базы приведена в Приложении Г в виде реляционной модели данных.

### **1.3 Организация ведения информационной базы**

Существует два способа поставки информационной базы: либо в виде уже готовой базы данных, либо в форме *SQL*-скрипта. Для создания базы данных из *SQL*-скрипта необходимо выполнить следующие шаги:

1. Создать и зарегистрировать новую базу данных.
2. Зайти в редактор скриптов и установить соединение с созданной базой данных.
3. Выполнить полученный *SQL*-скрипт.

При необходимости внесения изменений или добавления информации в базу данных используется интерфейс клиентской подсистемы. Все операции по обслуживанию СУБД, включая диагностику работоспособности и резервное копирование данных, должны проводиться в соответствии с установленными правилами эксплуатации и администрирования СУБД.

## 2 ОПИСАНИЕ ВНЕМАШИНОЙ ИНФОРМАЦИОННОЙ БАЗЫ

### 2.1 Логическая структура

В отличие от традиционных организаций, деятельность которых подчинена формализованным требованиям (таким как федеральные законы, ГОСТы, и иные нормативные акты), данная система функционирует в рамках собственной внутренней логики и саморегулируемых процессов. Это осознанный выбор, обусловленный следующими причинами:

#### 1. Нестандартный характер деятельности

Платформа не ведет классическую хозяйственную деятельность и не попадает под типовую классификацию "предприятие" или "юридическое лицо с типовым ОКВЭД". Ее функции связаны с маршрутизацией цифровых активов, управлением кошельками, агрегированием пользовательских операций и взаимодействием с внешними контрагентами через автоматизированные интерфейсы. Подобная деятельность часто выходит за рамки тех процессов, на которые ориентированы традиционные законы.

#### 2. Гибкость и независимость архитектуры

Использование собственных правил позволяет оперативно адаптировать систему под изменяющиеся условия: технические, инфраструктурные или пользовательские. Унифицированные государственные стандарты не всегда подходят к высокодинамичной среде цифровых активов.

Например, в отличие от традиционного бухгалтерского учета, система отчетности здесь предназначена не для налогообложения, а для:

- мониторинга технической стабильности;
- обеспечения контроля внутренней маршрутизации средств;
- принятия решений по операционным процессам в реальном времени.

#### 3. Прозрачность без внешней отчетности

Хотя деятельность платформы не регламентируется извне в привычном смысле, она предполагает внутреннюю прозрачность. Для этого выстраивается система отчетов, охватывающая все ключевые элементы инфраструктуры: от статусов кошельков и пользователей, обменников до аналитики по транзакциям.

Важно подчеркнуть:

Прозрачность обеспечивается не потому, что так требует закон, а потому, что это необходимо для самоуправления, координации ролей и доверия между участниками системы.

По результатам работы системы предусмотрено формирование отчетов:

##### 1. Отчет «Статусы пользователей»

Позволяет оперативно оценить текущее состояние пользовательской базы: кто активен, кто заблокирован, кто временно неактивен. Это критически важно для принятия решений об ограничении доступа, повторной активации или анализе оттока. Кому адресован: менеджер аккаунтов.

Отчёт содержит таблицу со следующей информацией:



- *username* — имя пользователя;
- *email* — адрес электронной почты;
- *status\_name* — название статуса;
- *updated\_at* — дата обновления.

## 2. Отчет «Статусы кошельков»

Позволяет следить за тем, какие кошельки в системе активны, а какие заблокированы или заморожены. Это важно для контроля над перемещением средств и безопасности системы. Кому адресован: менеджер аккаунтов.

Отчёт содержит таблицу со следующей информацией:

- *address* — адрес кошелька;
- *username* — имя пользователя;
- *email* — адрес электронной почты;
- *status\_name* — название статуса;
- *updated\_at* — дата обновления.

## 3. Отчет «Статусы обменников»

Позволяет видеть текущее состояние всех подключенных обменников — какие из них работают, какие недоступны. Это необходимо для принятия решений о маршрутизации средств и отключении/подключении провайдеров. Кому адресован: менеджер обменников.

Отчёт содержит таблицу со следующей информацией:

- *exchange\_name* — название обменника;
- *url* — адрес веб-сайта;
- *status\_name* — название статуса;
- *updated\_at* — дата обновления.

## 4. Отчет «Балансы токенов по кошелькам»

Позволяет получить полную картину по остаткам всех токенов на всех кошельках пользователей. Кому адресован: всем.

Отчёт содержит таблицу со следующей информацией:

- *wallet\_address* — адрес кошелька;
- *username* — владелец кошелька;
- *email* — адрес электронной почты;
- *token\_symbol* — символ токена;
- *token\_name* — полное название токена;
- *amount* — текущий остаток токена на кошельке;
- *updated\_at* — дата обновления.

## 5. Отчет «Транзакции с обменниками по статусам»

Отчет позволяет отслеживать текущие статусы всех транзакций, связанных с внешними обменниками. Это помогает менеджеру обменников

быстро выявлять зависшие, успешно выполненные или ошибочные переводы, контролировать работу обменников и своевременно реагировать на проблемы. Кому адресован: менеджер обменников.

Отчёт содержит таблицу со следующей информацией:

- *quantity\_wallet* — количество кошельков;
- *token\_symbol* — символ токена;
- *amount* — сумма переводов;
- *average\_commission* — средняя комиссия;
- *quantity\_ok* — количество удачных;
- *quantity\_average* — среднее состояние;
- *quantity\_error* — количество неудачных;
- *ok\_and\_error\_ratio* — соотношение ok/error.

#### 6. Отчет «Транзакции с обменниками по комиссиям»

Отчет позволяет анализировать суммы комиссий, уплаченных при проведении транзакций через биржи и обменники. Это помогает менеджеру обменников контролировать доходы от комиссий, выявлять аномалии, оптимизировать тарифы и выявлять подозрительные операции. Кому адресован: менеджер обменников.

Отчёт содержит таблицу со следующей информацией:

- *hash* — уникальный хэш транзакции;
- *wallet\_address* — адрес кошелька отправителя;
- *exchange\_name* — название биржи или обменника;
- *recipient\_address* — адрес получателя;
- *token\_symbol* — символ токена;
- *amount* — сумма перевода;
- *fee* — комиссия, уплаченная за транзакцию;
- *average\_commission* — средняя комиссия;
- *avg\_commission\_difference* — разница между средней комиссией и текущей.

#### 7. Отчет «Транзакции с обменниками по токенам»

Отчет помогает анализировать объемы и активность транзакций по каждому токenu, проходящему через биржи и обменники. Это позволяет менеджеру обменников оценивать популярность токенов, распределение оборотов и планировать работу с ликвидностью. Кому адресован: менеджер обменников.

Отчёт содержит таблицу со следующей информацией:

- *token\_symbol* — символ токена;
- *token\_name* — имя токена;
- *blockchain* — блокчейн;
- *quantity\_ok* — количество удачных;
- *quantity\_average* — среднее состояние;

- *quantity\_error* — количество неудачных;
- *ok\_and\_error\_ratio* — соотношение ok/error;
- *amount* — общая сумма операций;
- *average\_commission* — средняя комиссия.

#### 8. Отчет «Транзакции с обменниками по обменникам»

Отчет позволяет детально отслеживать все транзакции, проходящие через каждый конкретный обменник или биржу. Это важно для оценки активности, контроля качества работы конкретных обменников и анализа операционных показателей по каждому провайдеру. Кому адресован: менеджер обменников.

Отчёт содержит таблицу со следующей информацией:

- *exchange\_name* — название обменника;
- *url* — ссылка обменника;
- *quantity\_ok* — количество удачных;
- *quantity\_error* — количество неудачных;
- *ok\_and\_error\_ratio* — соотношение ok/error;
- *token\_symbol* — символ токена;
- *amount* — общая сумма перевода;
- *average\_commission* — средняя комиссия.

#### 9. Отчет «Внутренние транзакции по статусам»

Отчет отображает текущие статусы всех внутренних переводов между кошельками внутри платформы. Это помогает менеджеру внутренних транзакций контролировать успешность, обработку и сбои при переводах, а также выявлять проблемные операции для быстрого реагирования. Кому адресован: менеджер внутренних транзакций.

Отчёт содержит таблицу со следующей информацией:

- *quantity\_wallet* — количество кошельков;
- *token\_symbol* — символ токена;
- *amount* — сумма переводов;
- *average\_commission* — средняя комиссия;
- *quantity\_ok* — количество удачных;
- *quantity\_average* — среднее состояние;
- *quantity\_error* — количество неудачных;
- *ok\_and\_error\_ratio* — соотношение ok/error.

#### 10. Отчет «Внутренние транзакции по комиссии»

Отчет позволяет анализировать суммы комиссий, уплаченных при внутренних переводах между кошельками внутри платформы. Это помогает менеджеру внутренних транзакций контролировать эффективность и доходность внутренних операций, выявлять аномалии в комиссиях и

оптимизировать внутренние тарифы. Кому адресован: менеджер внутренних транзакций.

Отчёт содержит таблицу со следующей информацией:

- *hash* — уникальный хэш транзакции;
- *sender\_address* — адрес кошелька отправителя;
- *recipient\_address* — адрес получателя;
- *token\_symbol* — символ токена;
- *amount* — сумма перевода;
- *fee* — комиссия, уплаченная за транзакцию;
- *average\_commission* — средняя комиссия;
- *avg\_commission\_difference* — разница между средней комиссией и текущей.

#### 11. Отчет «Внутренние транзакции по токенам»

Отчет позволяет анализировать объемы и активность внутренних переводов по каждому токenu внутри платформы. Это помогает менеджеру внутренних транзакций отслеживать популярность токенов, распределение оборотов и оптимизировать управление ликвидностью. Кому адресован: менеджер внутренних транзакций.

Отчёт содержит таблицу со следующей информацией:

- *token\_symbol* — символ токена;
- *token\_name* — имя токена;
- *blockchain* — блокчейн;
- *quantity\_ok* — количество удачных;
- *quantity\_average* — среднее состояние;
- *quantity\_error* — количество неудачных;
- *ok\_and\_error\_ratio* — соотношение ok/error;
- *amount* — общая сумма операций;
- *average\_commission* — средняя комиссия.

## 2.2 Физическая структура

Формирование отчетов происходит через интерфейс подсистемы. Для отчетов имеется возможность выгрузки данных в следующие форматы:

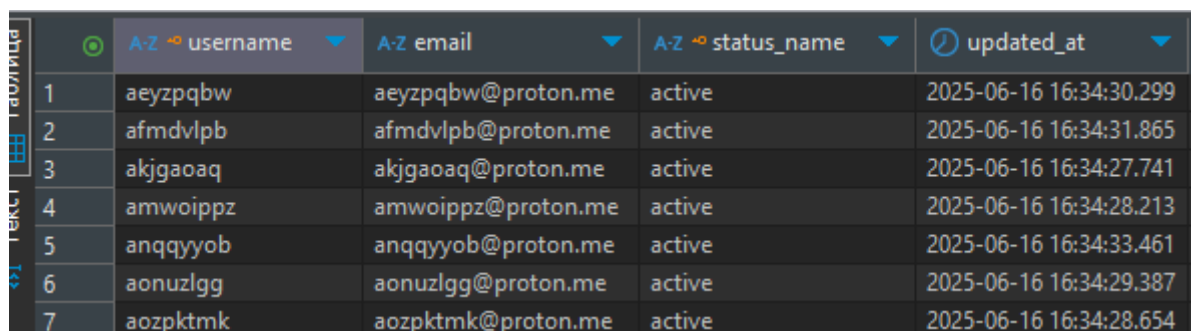
- 1) *XML*;
- 2) *JSON*;
- 3) *CSV*;
- 4) *HTML*.

### 2.2.1 Отчет «Статусы пользователей»

Запрос:

```
SELECT
    u.username,
    u.email,
    s.status_name,
    u.updated_at
FROM users u
JOIN statuses s ON u.status_id = s.id
ORDER BY u.username;
```

Результат запроса (рисунок 1).



The screenshot shows a database query result table with 5 columns: 'username', 'email', 'status\_name', and 'updated\_at'. The table contains 7 rows of data, all with 'active' status. The interface includes a sidebar with icons for 'таблица' (table), 'текст' (text), and 'схема' (schema), and a top bar with sorting options (A-Z, Z-A, and a clock icon).

	A-Z username	A-Z email	A-Z status_name	updated_at
1	aeypqbw	aeypqbw@proton.me	active	2025-06-16 16:34:30.299
2	afmdvlpb	afmdvlpb@proton.me	active	2025-06-16 16:34:31.865
3	akjgaoaq	akjgaoaq@proton.me	active	2025-06-16 16:34:27.741
4	amwoippz	amwoippz@proton.me	active	2025-06-16 16:34:28.213
5	anqyyob	anqyyob@proton.me	active	2025-06-16 16:34:33.461
6	aonuzlgg	aonuzlgg@proton.me	active	2025-06-16 16:34:29.387
7	aopkmtk	aopkmtk@proton.me	active	2025-06-16 16:34:28.654

Рисунок 1 – результат запроса

### 2.2.2 Отчет «Статусы кошельков»

Запрос:

```
SELECT
    w.address,
    u.username,
    u.email,
    s.status_name,
    w.updated_at
FROM wallets w
JOIN users u ON w.user_username = u.username
JOIN statuses s ON w.status_id = s.id
ORDER BY w.address;
```

Результат запроса (рисунок 2).

		A-Z address	A-Z username	A-Z email	A-Z status_name	updated_at
1		0017ed83583f5785	dpggcmqy	dpggcmqy@proton.me	active	2025-06-16 16:34:32.553
2		005185a05f560f33	tztpnipv	tztpnipv@proton.me	active	2025-06-16 16:34:29.012
3		00793027d492ba8c	yzxeveiw	yzxeveiw@proton.me	active	2025-06-16 16:34:31.637
4		008cd3e1316d1d79	hydgzvzhf	hydgzvzhf@proton.me	active	2025-06-16 16:34:31.845
5		009e3077cff1e25e	kawkmcqg	kawkmcqg@proton.me	active	2025-06-16 16:34:30.748
6		00aafc75c1d38edc	kojhoomw	kojhoomw@proton.me	active	2025-06-16 16:34:32.238
7		00b3a2b4eeb17be6	kckifbbr	kckifbbr@proton.me	active	2025-06-16 16:34:32.780

Рисунок 2 – результат запроса

### 2.2.3 Отчет «Статусы обменников»

Запрос:

```
SELECT
    e.exchange_name,
    e.url,
    s.status_name,
    e.updated_at
FROM exchanges e
JOIN statuses s ON e.status_id = s.id
ORDER BY e.exchange_name;
```

Результат запроса (рисунок 3).

		A-Z exchange_name	A-Z url	A-Z status_name	updated_at
1		Binance Dark	<a href="http://binanced33mke5mx.onion">http://binanced33mke5mx.onion</a>	active	2025-06-16 16:34:27.285
2		Bitfinex Shadow	<a href="http://bitfinex7sh4dowq.onion">http://bitfinex7sh4dowq.onion</a>	active	2025-06-16 16:34:27.285
3		Bitstamp Tor	<a href="http://bitstampTORx8bqd3.onion">http://bitstampTORx8bqd3.onion</a>	active	2025-06-16 16:34:27.285
4		Bybit Anonymous	<a href="http://bybit4privacy42dj.onion">http://bybit4privacy42dj.onion</a>	active	2025-06-16 16:34:27.285

Рисунок 3 – результат запроса

### 2.2.4 Отчет «Балансы токенов по кошелькам»

Запрос:

```
SELECT
    b.wallet_address,
    u.username,
    u.email,
    t.symbol AS token_symbol,
    t.token_name,
    b.amount,
    b.updated_at
FROM balances b
JOIN wallets w ON b.wallet_address = w.address
JOIN users u ON w.user_username = u.username
```

*JOIN tokens t ON b.token\_id = t.id  
ORDER BY b.wallet\_address, t.symbol;*

Результат запроса (рисунок 4).

	wallet_address	username	email	token_symbol	token_name	amount	updated_at
1	0017ed83583f5785	dgpgcmq	dgpgcmq@proton.me	DAI	Dai Stablecoin	2 518,35	2025-06-16 16:34:32.554
2	0017ed83583f5785	dgpgcmq	dgpgcmq@proton.me	LINK	Chainlink	4 467,1	2025-06-16 16:34:32.554
3	0017ed83583f5785	dgpgcmq	dgpgcmq@proton.me	USDC	USD Coin	7 536,95	2025-06-16 16:34:32.554
4	0017ed83583f5785	dgpgcmq	dgpgcmq@proton.me	USDT	Tether USD	1 779,93	2025-06-16 16:34:32.554
5	005185a05f560f33	tztpnipv	tztpnipv@proton.me	DAI	Dai Stablecoin	4 593,92	2025-06-16 16:34:29.014
6	005185a05f560f33	tztpnipv	tztpnipv@proton.me	LINK	Chainlink	6 533,87	2025-06-16 16:34:29.014
7	005185a05f560f33	tztpnipv	tztpnipv@proton.me	USDC	USD Coin	1 632,63	2025-06-16 16:34:29.014

Рисунок 4 – результат запроса

## 2.2.5 Отчет «Транзакции с обменниками по статусам»

Запрос:

*SELECT*

```

COUNT(DISTINCT et.wallet_address) AS quantity_wallet,
t.symbol AS token_symbol,
SUM(et.amount) AS amount,
AVG(et.fee) AS average_commission,
COUNT(CASE WHEN ts.transaction_status_name = 'confirmed' THEN 1
END) AS quantity_ok,
COUNT(CASE WHEN ts.transaction_status_name = 'pending' THEN 1
END) AS quantity_average,
COUNT(CASE WHEN ts.transaction_status_name = 'failed' THEN 1
END) AS quantity_error,
CASE
WHEN COUNT(CASE WHEN ts.transaction_status_name = 'failed'
THEN 1 END) = 0 THEN NULL
ELSE ROUND(
CAST(COUNT(CASE WHEN ts.transaction_status_name =
'confirmed' THEN 1 END) AS numeric) /
COUNT(CASE WHEN ts.transaction_status_name = 'failed' THEN 1
END), 2)
END AS ok_and_error_ratio
FROM exchange_transactions et
JOIN transaction_statuses ts ON et.transaction_status_id = ts.id
JOIN tokens t ON et.token_id = t.id
GROUP BY t.symbol
ORDER BY t.symbol;
```

Результат запроса (рисунок 5).

	123 quantity_wallet	A-Z token_symbol	123 amount	123 average_commission	123 quantity_ok	123 quantity_average	123 quantity_error	123 ok_and_error_ratio
1	1 557	DAI	6 490 250,35	3,4286643165	2 553	0	0	[NULL]
2	1 518	LINK	6 027 546,5	3,4544758395	2 442	0	0	[NULL]
3	1 492	USDC	6 171 191,63	3,4831651376	2 398	0	0	[NULL]
4	1 578	USDT	6 547 517,93	3,4671729958	2 607	0	0	[NULL]

Рисунок 5 – результат запроса

## 2.2.6 Отчет «Транзакции с обменниками по комиссиям»

Запрос:

```
WITH avg_fee AS (
    SELECT AVG(fee) AS average_commission FROM exchange_transactions
)
SELECT
    et.hash,
    et.wallet_address,
    e.exchange_name,
    et.recipient_address,
    t.symbol AS token_symbol,
    et.amount,
    et.fee,
    af.average_commission,
    et.fee - af.average_commission AS avg_commission_difference
FROM exchange_transactions et
JOIN exchanges e ON et.exchange_id = e.id
JOIN tokens t ON et.token_id = t.id
CROSS JOIN avg_fee af
ORDER BY et.created_at DESC;
```

Результат запроса (рисунок 6).

	A-Z hash	A-Z wallet_address	A-Z exchange_name	A-Z recipient_address	A-Z token_symbol	123 amount	123 fee	123 average_commission
1	12b6fe3bb8d3cabaf20361bdc2e9714e42cc349ab0c823a092472ce3f6488c	e1ac4bcc5bd4cb87	KuCoin Deep	b86d59e91d131d5	USDC	1 086,74	4,02	3,458076
2	a207f4dc6883e5db6e6c5dbb651cabe129ef68e816f7f7db9620c785b16fab905	e1ac4bcc5bd4cb87	Bitfinex Shadow	5b29ceb59c3eb889	USDT	521,93	0,57	3,458076
3	94b996ef026470a5e2041b3678cd9dc9efb3873fe7169f2870c0d194046e4c8	e1ac4bcc5bd4cb87	KuCoin Deep	2879bc4424495286	DAI	3 101,98	3,98	3,458076
4	635011ecc0d5167c7cae90e03c0b7b331d946116dff23409c0c9d7ae4b59a28	e1ac4bcc5bd4cb87	Coimbase Hidden	41b48a10f2435dcf	USDT	2 665,87	1,02	3,458076
5	e2b702897c3acff657b790b1b12f1bc2bac280caef8cd2a5c3de6f15aa9f094	e1ac4bcc5bd4cb87	Bitfinex Shadow	6956a0732080a99	DAI	2 806,7	0,12	3,458076
6	ae17b61f8d915ab14b2b38528018a7f29f95add58b0b0b81b380caa99c8ac3f	405833ceadc0c3b0	Coimbase Hidden	cc64d554031f09e9	USDT	3 737,31	1,44	3,458076

Рисунок 6 – результат запроса

## 2.2.7 Отчет «Транзакции с обменниками по токенам»

Запрос:

```
SELECT
    t.symbol AS token_symbol,
    t.token_name,
    b.blockchain_name AS blockchain,
    COUNT(CASE WHEN ts.transaction_status_name = 'confirmed' THEN 1
END) AS quantity_ok,
    COUNT(CASE WHEN ts.transaction_status_name = 'pending' THEN 1
END) AS quantity_average,
```



```

COUNT(CASE WHEN ts.transaction_status_name = 'failed' THEN 1
END) AS quantity_error,
CASE
    WHEN COUNT(CASE WHEN ts.transaction_status_name = 'failed'
THEN 1 END) = 0 THEN NULL
    ELSE ROUND(
        CAST(COUNT(CASE WHEN ts.transaction_status_name =
'confirmed' THEN 1 END) AS numeric) /
        COUNT(CASE WHEN ts.transaction_status_name = 'failed' THEN 1
END), 2)
    END AS ok_and_error_ratio,
SUM(et.amount) AS amount,
AVG(et.fee) AS average_commission
FROM exchange_transactions et
JOIN tokens t ON et.token_id = t.id
JOIN blockchains b ON t.blockchain_id = b.id
JOIN transaction_statuses ts ON et.transaction_status_id = ts.id
GROUP BY t.symbol, t.token_name, b.blockchain_name
ORDER BY t.symbol;

```

Результат запроса (рисунок 7).

	A-Z token_symbol	A-Z token_name	A-Z blockchain	123 quantity_ok	123 quantity_average	123 quantity_error	123 ok_and_error_ratio	123 amount	123 average_commission
1	DAI	Dai Stablecoin	Ethereum	2 553	0	0	[NULL]	6 490 250,35	3,4286643165
2	LINK	Chainlink	Ethereum	2 442	0	0	[NULL]	6 027 546,5	3,4544758395
3	USDC	USD Coin	Ethereum	2 398	0	0	[NULL]	6 171 191,63	3,4831651376
4	USDT	Tether USD	Ethereum	2 607	0	0	[NULL]	6 547 517,93	3,4671729958

Рисунок 7 – результат запроса

## 2.2.8 Отчет «Транзакции с обменниками по обменникам»

Запрос:

```

SELECT
    e.exchange_name,
    e.url,
    COUNT(CASE WHEN ts.transaction_status_name = 'confirmed' THEN 1
END) AS quantity_ok,
    COUNT(CASE WHEN ts.transaction_status_name = 'failed' THEN 1
END) AS quantity_error,
CASE
    WHEN COUNT(CASE WHEN ts.transaction_status_name = 'failed'
THEN 1 END) = 0 THEN NULL
    ELSE ROUND(
        CAST(COUNT(CASE WHEN ts.transaction_status_name =
'confirmed' THEN 1 END) AS numeric) /

```

```

COUNT(CASE WHEN ts.transaction_status_name = 'failed' THEN 1
END), 2)
END AS ok_and_error_ratio,
t.symbol AS token_symbol,
SUM(et.amount) AS amount,
AVG(et.fee) AS average_commission
FROM exchange_transactions et
JOIN exchanges e ON et.exchange_id = e.id
JOIN tokens t ON et.token_id = t.id
JOIN transaction_statuses ts ON et.transaction_status_id = ts.id
GROUP BY e.exchange_name, e.url, t.symbol
ORDER BY e.exchange_name, t.symbol;

```

Результат запроса (рисунок 8).

	A-z exchange_name	A-z url	123 quantity_ok	123 quantity_error	123 ok_and_error_ratio	A-z token_symbol	123 amount	123 average_commission
1	Binance Dark	<a href="http://binanced33mke5mx.onion">http://binanced33mke5mx.onion</a>	504	0	[NULL]	DAI	1 306 754,24	3,3173015873
2	Binance Dark	<a href="http://binanced33mke5mx.onion">http://binanced33mke5mx.onion</a>	478	0	[NULL]	LINK	1 196 618,69	3,5872803347
3	Binance Dark	<a href="http://binanced33mke5mx.onion">http://binanced33mke5mx.onion</a>	470	0	[NULL]	USDC	1 248 377,99	3,5091702128
4	Binance Dark	<a href="http://binanced33mke5mx.onion">http://binanced33mke5mx.onion</a>	532	0	[NULL]	USDT	1 349 151,38	3,3859962406
5	Bitfinex Shadow	<a href="http://bitfinex7sh4dowq.onion">http://bitfinex7sh4dowq.onion</a>	496	0	[NULL]	DAI	1 210 374,64	3,5416733871

Рисунок 8 – результат запроса

## 2.2.9 Отчет «Внутренние транзакции по статусам»

Запрос:

```

SELECT
COUNT(DISTINCT it.sender_address) + COUNT(DISTINCT
it.recipient_address) AS quantity_wallet,
t.symbol AS token_symbol,
SUM(it.amount) AS amount,
AVG(it.fee) AS average_commission,
COUNT(CASE WHEN ts.transaction_status_name = 'confirmed' THEN 1
END) AS quantity_ok,
COUNT(CASE WHEN ts.transaction_status_name = 'pending' THEN 1
END) AS quantity_average,
COUNT(CASE WHEN ts.transaction_status_name = 'failed' THEN 1
END) AS quantity_error,
CASE
WHEN COUNT(CASE WHEN ts.transaction_status_name = 'failed'
THEN 1 END) = 0 THEN NULL
ELSE ROUND(
CAST(COUNT(CASE WHEN ts.transaction_status_name =
'confirmed' THEN 1 END) AS numeric) /
COUNT(CASE WHEN ts.transaction_status_name = 'failed' THEN 1
END), 2)
END AS ok_and_error_ratio
FROM internal_transactions it

```

```

JOIN tokens t ON it.token_id = t.id
JOIN transaction_statuses ts ON it.transaction_status_id = ts.id
GROUP BY t.symbol
ORDER BY t.symbol;

```

Результат запроса (рисунок 9).

	A-Z exchange_name	A-Z url	I23 quantity_ok	I23 quantity_error	I23 ok_and_error_ratio	A-Z token_symbol	I23 amount	I23 average_commission
1	Binance Dark	http://binanced33mke5mx.onion	504	0	[NULL]	DAI	1 306 754,24	3,3173015873
2	Binance Dark	http://binanced33mke5mx.onion	478	0	[NULL]	LINK	1 196 618,69	3,5872803347
3	Binance Dark	http://binanced33mke5mx.onion	470	0	[NULL]	USDC	1 248 377,59	3,5091702128
4	Binance Dark	http://binanced33mke5mx.onion	532	0	[NULL]	USDT	1 349 151,38	3,3859962406

Рисунок 9 – результат запроса

## 2.2.10 Отчет «Внутренние транзакции по комиссиям»

Запрос:

```

WITH avg_fee AS (
    SELECT AVG(fee) AS average_commission FROM internal_transactions
)
SELECT
    it.hash,
    it.sender_address,
    it.recipient_address,
    t.symbol AS token_symbol,
    it.amount,
    it.fee,
    af.average_commission,
    it.fee - af.average_commission AS avg_commission_difference
FROM internal_transactions it
JOIN tokens t ON it.token_id = t.id
CROSS JOIN avg_fee af
ORDER BY it.created_at DESC;

```

Результат запроса (рисунок 10).

	A-Z hash	A-Z sender_address	A-Z recipient_address	A-Z token_symbol	I23 amount	I23 fee	I23 average_commission	I23 avg_commission_difference
1	fea583d8963d359f9b0f99417d8a390056fe2b8aeb3891fe12d583a443be2179	e1ac4bcc5bd4cb07	56a01ca077f64eba	LINK	290.01	5.13	3,49139625	1,63860375
2	c3acfb6d17c0c33d9d9f0f116c8a95002ef7f0b178a2f6c88aecd3d6c5b7313	e1ac4bcc5bd4cb07	56a01ca077f64eba	DAI	300.86	2.09	3,49139625	-1,40139625
3	e53bf2983f94940ccb94e54f1b807959f1d43070fa5da5cbb3ec90ab	e1ac4bcc5bd4cb07	56a01ca077f64eba	USDC	1 913.59	0.96	3,49139625	-2,53139625
4	abff9e0717802a161fbd12724445cb28c95837ecd1d3588d9f98112161e576	e1ac4bcc5bd4cb07	56a01ca077f64eba	USDT	2 192.92	2.91	3,49139625	-0,58139625
5	fe7b1f516a5b13724b4e1d202aaf1b08777b9fa02b6e3af5cd8093996ad18af	e1ac4bcc5bd4cb07	32029a4922c0d7ac	LINK	3 650.48	0.1	3,49139625	-3,39139625

Рисунок 10 – результат запроса

## 2.2.11 Отчет «Внутренние транзакции по токенам»

Запрос:

```

SELECT
    t.symbol AS token_symbol,
    t.token_name,
    b.blockchain_name AS blockchain,

```

```

COUNT(CASE WHEN ts.transaction_status_name = 'confirmed' THEN 1
END) AS quantity_ok,
COUNT(CASE WHEN ts.transaction_status_name = 'pending' THEN 1
END) AS quantity_average,
COUNT(CASE WHEN ts.transaction_status_name = 'failed' THEN 1
END) AS quantity_error,
CASE
WHEN COUNT(CASE WHEN ts.transaction_status_name = 'failed'
THEN 1 END) = 0 THEN NULL
ELSE ROUND(
CAST(COUNT(CASE WHEN ts.transaction_status_name =
'confirmed' THEN 1 END) AS numeric) /
COUNT(CASE WHEN ts.transaction_status_name = 'failed' THEN 1
END), 2)
END AS ok_and_error_ratio,
SUM(it.amount) AS amount,
AVG(it.fee) AS average_commission
FROM internal_transactions it
JOIN tokens t ON it.token_id = t.id
JOIN blockchains b ON t.blockchain_id = b.id
JOIN transaction_statuses ts ON it.transaction_status_id = ts.id
GROUP BY t.symbol, t.token_name, b.blockchain_name
ORDER BY t.symbol;

```

Результат запроса (рисунок 11).

	token_symbol	token_name	blockchain	quantity_ok	quantity_average	quantity_error	ok_and_error_ratio	amount	average_commission
1	DAI	Dai Stablecoin	Ethereum	10 000	0	0	[NULL]	25 290 230,02	3,505804
2	LINK	Chainlink	Ethereum	10 000	0	0	[NULL]	24 947 099,81	3,500653
3	USDC	USD Coin	Ethereum	10 000	0	0	[NULL]	25 080 247,99	3,493017
4	USDT	Tether USD	Ethereum	10 000	0	0	[NULL]	25 150 667,73	3,466111

Рисунок 11 – результат запроса

## 2.3 Организация ведения информационной базы

После обработки данные из базы данных извлекаются и сохраняются в файлы в вышеперечисленных форматах, удобных для хранения и дальнейшего использования. Эти отчёты хранятся на персональных компьютерах и доступны для просмотра сотрудникам с соответствующими правами доступа.

### **3 ОБЕСПЕЧЕНИЕ БЕЗОПАСНОСТИ БАЗЫ ДАННЫХ**

База данных представляет собой одну из форм информационных систем, значит, для обеспечения информационной безопасности необходимо гарантировать три основных свойства данных, хранящихся в базе данных: конфиденциальность, доступность и целостность. Для достижения этой цели СУБД предлагает следующие инструменты и механизмы:

1. Представления;
2. Триггеры;
3. Функции;
4. Роли и привилегии;
5. Резервное копирование и восстановление.

В дальнейшем мы рассмотрим эти механизмы и проведем соответствующие настройки в системе управления базами данных.

#### **3.1 Разработка мер по физической защите базы данных**

В отличие от традиционных организаций, где регламенты по физической безопасности диктуются внешними нормативными актами, данная система функционирует в рамках собственной архитектуры и внутренней логики. Несмотря на то, что в инфраструктуре используется один сервер, он является критически важным узлом — обрабатывает базы данных, управляет кошельками и маршрутизирует цифровые активы. Поэтому к его физической защите предъявляются особые требования, вытекающие из внутренних критериев безопасности и устойчивости.

Предусмотрены следующие меры:

1. Размещение сервера в контролируемом помещении

Сервер установлен в изолированном пространстве с ограниченным физическим доступом. Допуск предоставляется только доверенным техническим специалистам на основании внутреннего протокола.

2. Система видеонаблюдения и сигнализации

Помещение оснащено видеонаблюдением с архивированием записей и системой оповещения в случае несанкционированного проникновения.

3. Противопожарная безопасность

Используются датчики температуры и автономные средства пожаротушения, адаптированные под малое серверное помещение. Мониторинг осуществляется дистанционно в режиме 24/7.

4. Контролируемый доступ на основе внутреннего регламента

Введён пропускной режим с регистрацией всех посещений. Ключи доступа (физические или электронные) выданы ограниченному кругу лиц, согласно утверждённой внутренней модели доверия.

## 3.2 Разработка представлений

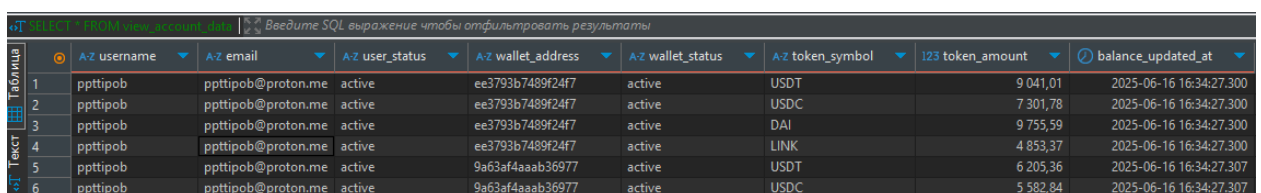
Представление в базе данных представляет собой результат выполнения запроса к базе данных, определенного с помощью оператора *SELECT*, в момент обращения к этому представлению. Иногда представления называют «виртуальными таблицами» из-за того, что они доступны для пользователя как таблицы, но сами не содержат данных; они извлекают данные из базовых таблиц в момент обращения к ним. Если данные в базовой таблице изменены, то при обращении пользователя к представлению, использующему эту таблицу, ему будут предоставлены актуальные данные.

Представление *view\_account\_data* показывает полную информацию о пользователях: их логины, email, статусы, связанные кошельки и балансы токенов. Оно удобно для сотрудников, которые занимаются управлением аккаунтами и поддержкой пользователей, чтобы быстро видеть состояние их кошельков и средств. Также это полезно для руководителей, которым нужен полный обзор по пользователям и их активам. Для сотрудников, работающих с транзакциями на биржах или внутренними переводами, это представление не актуально, так как там нет информации о движении средств.

Запрос:

```
CREATE VIEW view_account_data AS
SELECT u.username,
       u.email,
       s.status_name AS user_status,
       w.address AS wallet_address,
       sw.status_name AS wallet_status,
       t.symbol AS token_symbol,
       b.amount AS token_amount,
       b.updated_at AS balance_updated_at
FROM users u
JOIN statuses s ON u.status_id = s.id
LEFT JOIN wallets w ON u.username = w.user_username
LEFT JOIN statuses sw ON w.status_id = sw.id
LEFT JOIN balances b ON w.address = b.wallet_address
LEFT JOIN tokens t ON b.token_id = t.id;
```

Фрагмент данных представления показан на рисунке 12.



	A: username	A: email	A: user_status	A: wallet_address	A: wallet_status	A: token_symbol	123 token_amount	balance_updated_at
1	ppttipob	ppttipob@proton.me	active	ee3793b7489f24f7	active	USDT	9 041,01	2025-06-16 16:34:27.300
2	ppttipob	ppttipob@proton.me	active	ee3793b7489f24f7	active	USDC	7 301,78	2025-06-16 16:34:27.300
3	ppttipob	ppttipob@proton.me	active	ee3793b7489f24f7	active	DAI	9 755,59	2025-06-16 16:34:27.300
4	ppttipob	ppttipob@proton.me	active	ee3793b7489f24f7	active	LINK	4 853,37	2025-06-16 16:34:27.300
5	ppttipob	ppttipob@proton.me	active	9a63af4aaab36977	active	USDT	6 205,36	2025-06-16 16:34:27.307
6	ppttipob	ppttipob@proton.me	active	9a63af4aaab36977	active	USDC	5 582,84	2025-06-16 16:34:27.307

Рисунок 12 – фрагмент данных

Представление `view_exchange_transactions_full` содержит подробную информацию о транзакциях, связанных с биржами: кто отправитель, на какой бирже произошла операция, получатель, тип токена, сумма, комиссия, статус и время создания. Это удобно для сотрудников, которые контролируют и проверяют операции с биржами, отслеживают статусы переводов и решают вопросы по проблемным транзакциям. Также руководители, отвечающие за работу с внешними обменами, могут использовать это представление для мониторинга и анализа. Для тех, кто занимается внутренними переводами или управлением аккаунтами и балансами, эта информация не является основной.

Запрос:

```
CREATE VIEW view_exchange_transactions_full AS
SELECT et.hash,
       w.user_username AS sender_username,
       et.wallet_address,
       e.exchange_name,
       et.recipient_address,
       t.symbol AS token_symbol,
       et.amount,
       et.fee,
       ts.transaction_status_name AS status,
       et.created_at
FROM exchange_transactions et
JOIN wallets w ON et.wallet_address = w.address
JOIN exchanges e ON et.exchange_id = e.id
JOIN tokens t ON et.token_id = t.id
JOIN transaction_statuses ts ON et.transaction_status_id = ts.id;
```

Фрагмент данных представления показан на рисунке 13.

	A: hash	A: sender_username	A: wallet_address	A: exchange_name	A: recipient_address	A: token_symbol	123 amount	123 fee	A: status	created
1	2c525286528f0d019d76acc3175487a8indexdec22a4469c53cebfad06b743	pptipob	ea3793b7489f247	Coinbase Hidden	b778-da19ed6770ce	DAI	4 085,27	4,38	confirmed	2025-06-16
2	0e813a5bbd17da139f0912f9a2d3d8ecb4b56b5fa010d1d94c325bfc333213	pptipob	ea3793b7489f247	Coinbase Hidden	f1f7e04e0e7024d0	USDT	3 049,12	1,31	confirmed	2025-06-16
3	11d1b7eac113dc385a5a0a2d21f0ad188cf5a7d0069cf884bc6a92779c4ff	pptipob	ea3793b7489f247	KuCoin Deep	d17a44c99a830ea	USDC	412,73	5,03	confirmed	2025-06-16
4	5bff39363591c55a5304af833fda6ef1d2077f5181f22a338d6efcb235dad85	pptipob	ea3793b7489f247	Binance Dark	09afa905a88f3da	LINK	3 597,84	0,67	confirmed	2025-06-16
5	91b9cfb3a3cc6a2df5fbb8afa2b7aad6b7655415aed0909473abc499dc208df	pptipob	ea3793b7489f247	Coinbase Hidden	1674943a94a09b07	USDT	1 747,41	4,28	confirmed	2025-06-16

Рисунок 13 – фрагмент данных

Представление `view_internal_transactions_full` показывает подробные данные о внутренних переводах между пользователями платформы: кто отправитель и получатель (как по адресу кошелька, так и по имени пользователя), какой токен участвовал, сумма, комиссия, статус транзакции и время создания. Это представление полезно для сотрудников, которые занимаются контролем и анализом внутренних переводов, разрешают спорные ситуации и следят за корректностью операций внутри платформы. Для тех, кто работает с внешними биржевыми транзакциями или управляет аккаунтами и балансами, эта информация не является приоритетной.

Запрос:

```
CREATE VIEW view_internal_transactions_full AS
```

```

SELECT it.hash,
       sw.user_username AS sender_username,
       it.sender_address,
       rw.user_username AS recipient_username,
       it.recipient_address,
       t.symbol AS token_symbol,
       it.amount,
       it.fee,
       ts.transaction_status_name AS status,
       it.created_at
FROM internal_transactions it
JOIN wallets sw ON it.sender_address = sw.address
JOIN wallets rw ON it.recipient_address = rw.address
JOIN tokens t ON it.token_id = t.id
JOIN transaction_statuses ts ON it.transaction_status_id = ts.id;

```

Фрагмент данных представления показан на рисунке 14.

hash	sender_username	sender_address	recipient_username	recipient_address	token_symbol	amount	fee	status	created_at
9fcb378801d27d2395dc2232e71da0ce0f302010bce4b2a942e7e55a93b87b6c	pptipob	ee3793b7489f247	myqfadey	00ca700029ed13aa	USDT	1 570.61	0.26	confirmed	2025-06
27e5da0abfa0a1148eb5b615a2975f3e07c1cc9a400169a3f4766e47ca7477	pptipob	ee3793b7489f247	myqfadey	00ca700029ed13aa	USDC	991.23	5.04	confirmed	2025-06
a69beeb20aee808bd5726e909d4e7468ae5e93d2d2cb2a0ec28c5b4ff9208	pptipob	ee3793b7489f247	myqfadey	00ca700029ed13aa	DAI	1 977.82	2.31	confirmed	2025-06
26b4297af1502aa5c5da1161371b933b618d997508fc94629bee4bf23a737be	pptipob	ee3793b7489f247	myqfadey	00ca700029ed13aa	LINK	662.7	3.81	confirmed	2025-06
cb2169f4c7260f9d03f60630bb6503eda8419aca8e29114dc564662bf30614	pptipob	ee3793b7489f247	sgdstds	d04cbdc506e95ba	USDT	3 349.5	0.1	confirmed	2025-06
259a12d74d6687659cba9c23961414ae77c27e03935f6195ffa83a5ce0ada	pptipob	ee3793b7489f247	sgdstds	d04cbdc506e95ba	USDC	3 337.92	4.55	confirmed	2025-06
828061a27ba5565f60a519548c3f3449601fe3a7728c8697a9c2731b390598	pptipob	ee3793b7489f247	sgdstds	d04cbdc506e95ba	DAI	4 412.85	1.39	confirmed	2025-06
9fc775eb1d211df0b5509a642845fda4f46a05983802143361eb06c2a032c1d	pptipob	ee3793b7489f247	sgdstds	d04cbdc506e95ba	LINK	2 777.68	5.68	confirmed	2025-06

Рисунок 14 – фрагмент данных

Представление `view_token_info` содержит основную информацию о токенах: их символы, полные названия, адреса контрактов и данные о соответствующих блокчейнах. Это удобно для сотрудников, которые работают с каталогом токенов, настраивают или проверяют поддержку различных криптовалют и блокчейнов. Также это полезно для тех, кто занимается интеграцией или мониторингом активов на платформе. Для сотрудников, фокусирующихся на транзакциях или управлении аккаунтами, это представление носит вспомогательный характер.

Запрос:

```

CREATE VIEW view_token_info AS
SELECT t.symbol AS token_symbol,
       t.token_name,
       t.contract_address,
       b.symbol AS blockchain_symbol,
       b.blockchain_name
FROM tokens t
JOIN blockchains b ON t.blockchain_id = b.id;

```

Фрагмент данных представления показан на рисунке 15.



	A-Z token_symbol	A-Z token_name	A-Z contract_address	A-Z blockchain_symbol	A-Z blockchain_name
1	USDT	Tether USD	0xdAC17F958D2ee523a2206206994597C13D831ec7	ETH	Ethereum
2	USDC	USD Coin	0xA0b86991c6218b36c1d19D4a2e9Eb0cE3606EB48	ETH	Ethereum
3	DAI	Dai Stablecoin	0x6B175474E89094C4ADa98b954EedeAC495271d0F	ETH	Ethereum
4	LINK	Chainlink	0x514910771AF9Ca656af840dff83E8264EcF986CA	ETH	Ethereum

Рисунок 15 – фрагмент данных

Представление *view\_transactions\_all* объединяет информацию о всех транзакциях на платформе — как связанных с биржами, так и внутренних переводах между пользователями. Оно показывает тип транзакции, отправителя, получателя, используемый токен, сумму, комиссию, статус и время создания. Такое представление удобно для сотрудников, которым нужен полный обзор всех движений средств, например, для аналитиков, службы безопасности или руководителей, контролирующих финансовые операции. Для специалистов, работающих только с отдельными частями системы (только внутренние переводы или только аккаунты и балансы), это представление может быть избыточным.

Запрос:

```
CREATE VIEW view_transactions_all AS
SELECT
    'exchange' AS transaction_type,
    et.hash,
    w.user_username AS sender_username,
    et.recipient_address,
    e.exchange_name AS recipient_entity,
    t.symbol AS token_symbol,
    et.amount,
    et.fee,
    ts.transaction_status_name AS status,
    et.created_at
FROM exchange_transactions et
JOIN wallets w ON et.wallet_address = w.address
JOIN exchanges e ON et.exchange_id = e.id
JOIN tokens t ON et.token_id = t.id
JOIN transaction_statuses ts ON et.transaction_status_id = ts.id

UNION ALL

SELECT
    'internal' AS transaction_type,
    it.hash,
    sw.user_username AS sender_username,
    it.recipient_address,
    rw.user_username AS recipient_entity,
    t.symbol AS token_symbol,
    it.amount,
```

```

it.fee,
ts.transaction_status_name AS status,
it.created_at
FROM internal_transactions it
JOIN wallets sw ON it.sender_address = sw.address
JOIN wallets rw ON it.recipient_address = rw.address
JOIN tokens t ON it.token_id = t.id
JOIN transaction_statuses ts ON it.transaction_status_id = ts.id;

```

Фрагмент данных представления показан на рисунке 16.

	transaction_type	AZ hash	AZ sender_username	AZ recipient_address	AZ recipient_entity	AZ token_symbol	123 amount	123 fee	AZ status	created
7	exchange	60697b4fd3df8a1ba96b6316bccdcfd88a16fb73dfb216cc16f6bf42507eaa7b	pprtipob	f1a09f175b11f111	Binance Dark	USDT	1 059,64	4,89	confirmed	2023-06-16
8	exchange	5a66a2170a4320af019af096f18768dba01c417ffdcab47e2594c52ca5675	pprtipob	01b9da8219ec978	Bitfinex Shadow	USDC	963,89	6,79	confirmed	2023-06-16
9	exchange	12b94f702e79a15b831ea561e94cdf2aae49d533ae381b2cb56aa77b08c531	pprtipob	fcce36d02ce9f02d	Kraken	USDC	4 961,74	0,01	confirmed	2023-06-16
10	exchange	6ef5393bcd41ab85249f16da377298d52a061efc8f689c92b70863a6ecd20f65	pprtipob	18e2d9a2b3f59a5	Bitfinex Shadow	LINK	4 342,41	1,3	confirmed	2023-06-16
11	exchange	a801b2c49db8b9ae289df5f1a38fc1c1f39e35b9f138e290c10bb7aba2925368	pprtipob	#f59835d21fc0e7b	Bitfinex Shadow	USDC	3 737,44	0,02	confirmed	2023-06-16
12	exchange	6c50de0b3cd4605d331d3141b0c8d7d840497e7b4f9e08c9fdc2593fce2e3f0d	pprtipob	ae794fc62eededcd	Binance Dark	USDT	3 784,91	2,07	confirmed	2023-06-16
13	exchange	0b04756e0428e566d8c1b41619572b312869d5fe39c46847d82efec3912d9f78	pprtipob	f5a4d37129a5ea2c	Coinbase Hidden	USDT	4 303,78	4,87	confirmed	2023-06-16
14	exchange	cc1752e4fca6017f64089f3820c0b8f1fedebc46fbcccd8a6a1bd9bee0bd	pprtipob	12cd50f434c6a9b7	Coinbase Hidden	USDC	2 622,07	6,46	confirmed	2023-06-16

Рисунок 16 – фрагмент данных

### 3.3 Разработка триггеров

Триггер представляет собой инструкцию для базы данных, которая автоматически выполняет определенное действие при возникновении определенного типа операции. Триггеры могут поддерживать условия ссылочной целостности, гарантировать правильность вводимых данных перед тем, как они будут добавлены в таблицу.

В рамках курсовой работы были реализованы триггеры, обеспечивающие автоматизацию логики, проверку целостности данных и выполнение бизнес-правил. Ниже приведены триггеры, реализованные для базы данных.

#### 3.3.1 Триггер «trg\_allow\_internal\_update\_only\_if\_pending»

Ограничивает обновление внутренних транзакций только в случае, если их статус — «pending». Это предотвращает изменение завершённых или отменённых транзакций.

Код:

```

CREATE TRIGGER trg_allow_internal_update_only_if_pending
BEFORE UPDATE ON internal_transactions
FOR EACH ROW
EXECUTE FUNCTION allow_internal_update_only_if_pending();

```

### 3.3.2 Триггер «*trg\_allow\_update\_only\_if\_pending*»

Запрещает обновление транзакций, связанных с биржами, если их статус не «*pending*». Это обеспечивает неизменяемость завершённых или неуспешных транзакций.

Код:

```
CREATE TRIGGER trg_allow_update_only_if_pending
BEFORE UPDATE ON exchange_transactions
FOR EACH ROW
EXECUTE FUNCTION allow_update_only_if_pending();
```

### 3.3.3 Триггер «*trg\_allow\_update\_only\_if\_suspended*»

Разрешает изменение данных о бирже только в том случае, если её текущий статус — «*suspended*», либо если обновляется только поле «*status\_id*». Это предотвращает правки активных или удалённых бирж.

Код:

```
CREATE TRIGGER trg_allow_update_only_if_suspended
BEFORE UPDATE ON exchanges
FOR EACH ROW
EXECUTE FUNCTION allow_exchange_update_only_if_suspended();
```

### 3.3.4 Триггер «*trg\_check\_proton\_email*»

Гарантирует, что при создании или обновлении пользователя используется только электронная почта с доменом «*@proton.me*». Это может быть нужно для соблюдения политики конфиденциальности или ограничения регистрации.

Код:

```
CREATE TRIGGER trg_check_proton_email
BEFORE INSERT OR UPDATE ON users
FOR EACH ROW
EXECUTE FUNCTION check_proton_email();
```

### 3.3.5 Триггер «*trg\_deduct\_balance\_if\_confirmed*»

Автоматически вычитает сумму и комиссию из баланса кошелька при смене статуса биржевой транзакции на «*confirmed*». Обеспечивает корректное отражение завершённой транзакции в балансе пользователя.

Код:

```
CREATE TRIGGER trg_deduct_balance_if_confirmed  
AFTER UPDATE ON exchange_transactions  
FOR EACH ROW  
EXECUTE FUNCTION deduct_balance_if_confirmed();
```

### 3.3.6 Триггер «trg\_email\_change\_only\_if\_suspended»

Запрещает изменение «email» пользователя, если его статус не равен «suspended». Это ограничение повышает безопасность и управляемость аккаунтов.

Код:

```
CREATE TRIGGER trg_email_change_only_if_suspended  
BEFORE UPDATE OF email ON users  
FOR EACH ROW  
EXECUTE FUNCTION allow_email_change_only_if_suspended();
```

### 3.3.7 Триггер «trg\_prevent\_delete\_confirmed\_exchange\_tx»

Блокирует удаление биржевой транзакции, если её статус — «confirmed». Это обеспечивает неизменяемость подтверждённых операций в системе и предотвращает возможную потерю данных.

Код:

```
CREATE TRIGGER trg_prevent_delete_confirmed_exchange_tx  
BEFORE DELETE ON exchange_transactions  
FOR EACH ROW  
EXECUTE FUNCTION prevent_delete_confirmed_exchange_tx();
```

### 3.3.8 Триггер «trg\_prevent\_delete\_confirmed\_internal\_tx»

Запрещает удаление внутренней транзакции между пользователями, если её статус — «confirmed». Это сохраняет неизменяемость подтверждённых переводов внутри платформы.

Код:

```
CREATE TRIGGER trg_prevent_delete_confirmed_internal_tx  
BEFORE DELETE ON internal_transactions  
FOR EACH ROW  
EXECUTE FUNCTION prevent_delete_confirmed_internal_tx();
```

### 3.3.9 Триггер «*trg\_prevent\_exchange\_status\_change\_if\_banned*»

Блокирует изменение статуса биржи, если текущий статус — «*banned*». Это предотвращает восстановление или изменение заблокированных бирж.

Код:

```
CREATE TRIGGER trg_prevent_exchange_status_change_if_banned
BEFORE UPDATE OF status_id ON exchanges
FOR EACH ROW
EXECUTE FUNCTION prevent_exchange_status_change_if_banned();
```

### 3.3.10 Триггер «*trg\_prevent\_status\_change\_if\_banned*»

Запрещает изменение статуса пользователя, если его текущий статус — «*banned*». Это не даёт восстановить или изменить статус заблокированного пользователя.

Код:

```
CREATE TRIGGER trg_prevent_status_change_if_banned
BEFORE UPDATE OF status_id ON users
FOR EACH ROW
EXECUTE FUNCTION prevent_status_change_if_banned();
```

### 3.3.11 Триггер «*trg\_prevent\_wallet\_status\_change\_if\_banned*»

Запрещает изменение статуса кошелька, если его текущий статус — «*banned*». Это предотвращает восстановление или изменение статуса заблокированного кошелька.

Код:

```
CREATE TRIGGER trg_prevent_wallet_status_change_if_banned
BEFORE UPDATE OF status_id ON wallets
FOR EACH ROW
EXECUTE FUNCTION prevent_wallet_status_change_if_banned();
```

### 3.3.12 Триггер «*trg\_set\_balances\_updated\_at*»

Автоматически обновляет поле «*updated\_at*» у балансов токенов при их изменении, чтобы фиксировать время последнего обновления записи.

Код:

```
CREATE TRIGGER trg_set_balances_updated_at
BEFORE UPDATE ON balances
FOR EACH ROW
```

```
EXECUTE FUNCTION set_balances_updated_at();
```

### **3.3.13 Триггер «*trg\_set\_exchange\_tx\_updated\_at*»**

Автоматически обновляет поле «*updated\_at*» у записей биржевых транзакций при их изменении, чтобы фиксировать время последнего обновления.

Код:

```
CREATE TRIGGER trg_set_exchange_tx_updated_at  
BEFORE UPDATE ON exchange_transactions  
FOR EACH ROW  
EXECUTE FUNCTION set_exchange_tx_updated_at();
```

### **3.3.14 Триггер «*trg\_set\_exchanges\_updated\_at*»**

Автоматически обновляет поле «*updated\_at*» у записей бирж при их изменении, фиксируя время последнего обновления.

Код:

```
CREATE TRIGGER trg_set_exchanges_updated_at  
BEFORE UPDATE ON exchanges  
FOR EACH ROW  
EXECUTE FUNCTION set_exchanges_updated_at();
```

### **3.3.15 Триггер «*trg\_set\_internal\_tx\_updated\_at*»**

Автоматически обновляет поле «*updated\_at*» у внутренних транзакций при их изменении, чтобы фиксировать время последнего обновления записи.

Код:

```
CREATE TRIGGER trg_set_internal_tx_updated_at  
BEFORE UPDATE ON internal_transactions  
FOR EACH ROW  
EXECUTE FUNCTION set_internal_tx_updated_at();
```

### **3.3.16 Триггер «*trg\_set\_users\_updated\_at*»**

Автоматически обновляет поле «*updated\_at*» у записей пользователей при их изменении, фиксируя время последнего обновления.

Код:

```
CREATE TRIGGER trg_set_users_updated_at  
BEFORE UPDATE ON users
```

```
FOR EACH ROW  
EXECUTE FUNCTION set_users_updated_at();
```

### **3.3.17 Триггер «trg\_sync\_wallets\_status»**

Синхронизирует статус всех кошельков пользователя при изменении статуса самого пользователя. Если у пользователя меняется «*status\_id*», у всех его кошельков автоматически обновляется этот статус.

```
Код:  
CREATE TRIGGER trg_sync_wallets_status  
AFTER UPDATE OF status_id ON users  
FOR EACH ROW  
EXECUTE FUNCTION sync_wallets_status_with_user();
```

### **3.3.18 Триггер «trg\_update\_balances\_after\_confirmed\_internal\_tx»**

Обновляет балансы кошельков после подтверждения внутренней транзакции. При переходе статуса транзакции в «*confirmed*» списывает сумму с отправителя (включая комиссию) и зачисляет сумму получателю.

```
Код:  
CREATE TRIGGER trg_update_balances_after_confirmed_internal_tx  
AFTER UPDATE ON internal_transactions  
FOR EACH ROW  
EXECUTE FUNCTION update_balances_after_confirmed_internal_tx();
```

### **3.3.19 Триггер «trg\_update\_wallets\_updated\_at»**

Автоматически обновляет поле «*updated\_at*» у записей кошельков при их изменении, чтобы фиксировать время последнего обновления.

```
Код:  
CREATE TRIGGER trg_update_wallets_updated_at  
BEFORE UPDATE ON wallets  
FOR EACH ROW  
EXECUTE FUNCTION update_wallets_updated_at();
```

### **3.3.20 Триггер «trg\_validate\_exchange\_transaction»**

Валидация транзакции на бирже перед вставкой или обновлением. Проверяет, что кошелек и биржа активны, баланс достаточен для суммы и комиссии, а комиссия не превышает 5% от суммы.

Код:

```
CREATE TRIGGER trg_validate_exchange_transaction  
BEFORE INSERT OR UPDATE ON exchange_transactions  
FOR EACH ROW  
EXECUTE FUNCTION validate_exchange_transaction();
```

### **3.3.21 Триггер «trg\_validate\_internal\_transaction»**

Валидация внутренней транзакции перед вставкой или обновлением. Проверяет, что отправитель и получатель — разные активные кошельки, баланс отправителя достаточен для суммы и комиссии, а комиссия не превышает 5% от суммы.

Код:

```
CREATE TRIGGER trg_validate_internal_transaction  
BEFORE INSERT OR UPDATE ON internal_transactions  
FOR EACH ROW  
EXECUTE FUNCTION validate_internal_transaction();
```

### **3.3.22 Триггер «trg\_validate\_onion\_url»**

Валидация URL биржи перед вставкой или обновлением. Проверяет, что URL заканчивается на «.onion».

Код:

```
CREATE TRIGGER trg_validate_onion_url  
BEFORE INSERT OR UPDATE OF url ON exchanges  
FOR EACH ROW  
EXECUTE FUNCTION validate_onion_url();
```



### 3.4 Разработка функций

В данном разделе описываются функции, которые играют критически важную роль в обеспечении надёжности и безопасности системы, поскольку автоматически обрабатывают события, связанные с транзакциями, пользователями, кошельками и биржами.

Функциональность включает в себя следующее:

- Проверка правомерности создания и изменения транзакций — функции *validate\_exchange\_transaction* и *validate\_internal\_transaction* срабатывают перед вставкой или обновлением записей в таблицы транзакций, проверяя статусы, наличие средств, лимит комиссии и корректность сторон.

- Запрет на удаление подтверждённых транзакций — функции *prevent\_delete\_confirmed\_exchange\_tx* и *prevent\_delete\_confirmed\_internal\_tx* не позволяют удалить записи, если их статус равен «confirmed».

- Списание и начисление средств по транзакциям — функции *deduct\_balance\_if\_confirmed* и *update\_balances\_after\_confirmed\_internal\_tx* выполняют корректировку балансов при подтверждении транзакций, включая комиссии.

- Синхронизация статусов между связанными сущностями — например, *sync\_wallets\_status\_with\_user* обновляет статус всех кошельков при изменении статуса пользователя.

- Контроль правомерности изменений статуса — функции *prevent\_status\_change\_if\_banned*, *prevent\_wallet\_status\_change\_if\_banned*, *prevent\_exchange\_status\_change\_if\_banned* не позволяют изменять статусы объектов, если они находятся в заблокированном (*banned*) состоянии.

- Ограничения на изменение полей — функции *allow\_email\_change\_only\_if\_suspended*, *allow\_update\_only\_if\_pending*, *allow\_internal\_update\_only\_if\_pending* и другие позволяют изменять поля объекта только при определённых статусах.

- Обновление временных меток — функции *set\_users\_updated\_at*, *set\_wallets\_updated\_at*, *set\_exchange\_tx\_updated\_at* и аналогичные устанавливают поле *updated\_at* в текущую дату и время при каждом обновлении.

- Проверка корректности URL-адреса — функция *validate\_onion\_url* гарантирует, что адрес биржи является валидным «.onion-URL».

- Проверка корректности email-адреса — функция *check\_proton\_email* проверяет, что *email* пользователя зарегистрирован на домене *ProtonMail*.

Каждая из описанных функций связана с соответствующим триггером, который срабатывает при вставке, обновлении или удалении данных в конкретной таблице. Это обеспечивает строгую реализацию бизнес-правил на уровне базы данных и предотвращает ошибки, вызванные некорректными действиями пользователей или приложений.

### 3.4.1 Функция «allow\_internal\_update\_only\_if\_pending»

```
CREATE OR REPLACE FUNCTION
allow_internal_update_only_if_pending()
RETURNS TRIGGER AS $$
DECLARE
    current_status VARCHAR;
BEGIN
    SELECT transaction_status_name INTO current_status
    FROM transaction_statuses
    WHERE id = OLD.transaction_status_id;

    IF current_status <> 'pending' THEN
        RAISE EXCEPTION 'Internal transaction can only be modified when
status is "pending"';
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

### 3.4.2 Функция «allow\_update\_only\_if\_pending»

```
CREATE OR REPLACE FUNCTION allow_update_only_if_pending()
RETURNS TRIGGER AS $$
DECLARE
    status_name VARCHAR;
BEGIN
    -- Получаем имя текущего статуса транзакции
    SELECT transaction_status_name INTO status_name
    FROM transaction_statuses
    WHERE id = OLD.transaction_status_id;

    -- Разрешаем изменения только если статус "pending"
    IF status_name <> 'pending' THEN
        RAISE EXCEPTION 'Exchange transaction can only be modified when
status is "pending"';
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

### 3.4.3 Функция «allow\_exchange\_update\_only\_if\_suspended»

```
CREATE OR REPLACE FUNCTION
public.allow_exchange_update_only_if_suspended()
RETURNS trigger
LANGUAGE plpgsql
AS $function$
DECLARE
    current_status_name VARCHAR;
BEGIN
    -- Получаем имя текущего статуса
    SELECT status_name INTO current_status_name
    FROM statuses
    WHERE id = OLD.status_id;

    -- Если меняется что-то кроме status_id
    IF NEW.status_id = OLD.status_id THEN
        IF current_status_name <> 'suspended' THEN
            RAISE EXCEPTION 'Only status_id can be updated unless status is
"suspended"';
        END IF;
    END IF;

    RETURN NEW;
END;
$function$;
```

### 3.4.4 Функция «check\_proton\_email»

```
CREATE OR REPLACE FUNCTION check_proton_email()
RETURNS TRIGGER AS $$
BEGIN
    -- Проверка: email должен оканчиваться на '@proton.me'
    IF RIGHT(NEW.email, 10) <> '@proton.me' THEN
        RAISE EXCEPTION 'Email must be a @proton.me address';
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

### 3.4.5 Функция «deduct\_balance\_if\_confirmed»

```
CREATE OR REPLACE FUNCTION deduct_balance_if_confirmed()
```

```

RETURNS TRIGGER AS $$
DECLARE
    status_name VARCHAR;
BEGIN
    -- Получаем имя нового статуса
    SELECT transaction_status_name INTO status_name
    FROM transaction_statuses
    WHERE id = NEW.transaction_status_id;

    -- Только если статус "confirmed"
    IF status_name = 'confirmed' THEN
        -- Пытаемся обновить баланс
        UPDATE balances
        SET amount = amount - (NEW.amount + NEW.fee),
            updated_at = now()
        WHERE wallet_address = NEW.wallet_address
            AND token_id = NEW.token_id;

        -- Можно добавить проверку, чтобы убедиться, что баланс был
найден
        IF NOT FOUND THEN
            RAISE EXCEPTION 'Balance entry not found for wallet % and token
%', NEW.wallet_address, NEW.token_id;
        END IF;
    END IF;

    RETURN NULL; -- AFTER-триггер
END;
$$ LANGUAGE plpgsql;

```

### 3.4.6 Функция «allow\_email\_change\_only\_if\_suspended»

```

CREATE OR REPLACE FUNCTION
allow_email_change_only_if_suspended()
RETURNS TRIGGER AS $$
DECLARE
    current_status_name VARCHAR;
BEGIN
    -- Проверяем, изменяется ли email
    IF NEW.email IS DISTINCT FROM OLD.email THEN
        -- Получаем текущее имя статуса
        SELECT status_name INTO current_status_name
        FROM statuses
        WHERE id = OLD.status_id;
    
```

```

        -- Если статус не "suspended", запрещаем изменение email
        IF current_status_name <> 'suspended' THEN
            RAISE EXCEPTION 'Email can only be changed when status is
"suspended"';
        END IF;
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

```

#### 3.4.7 Функция «prevent\_delete\_confirmed\_exchange\_tx»

```

CREATE          OR          REPLACE          FUNCTION
prevent_delete_confirmed_exchange_tx()
RETURNS trigger
LANGUAGE plpgsql
AS $$
DECLARE
    status_name TEXT;
BEGIN
    -- Получаем текстовое имя статуса
    SELECT transaction_status_name
    INTO status_name
    FROM transaction_statuses
    WHERE id = OLD.transaction_status_id;

    IF status_name = 'confirmed' THEN
        RAISE EXCEPTION 'Cannot delete confirmed exchange transaction';
    END IF;

    RETURN OLD;
END;
$$;

```

#### 3.4.8 Функция «prevent\_delete\_confirmed\_internal\_tx»

```

CREATE          OR          REPLACE          FUNCTION
prevent_delete_confirmed_internal_tx()
RETURNS trigger
LANGUAGE plpgsql
AS $$
DECLARE

```

```

    status_name TEXT;
BEGIN
    SELECT transaction_status_name
    INTO status_name
    FROM transaction_statuses
    WHERE id = OLD.transaction_status_id;

    IF status_name = 'confirmed' THEN
        RAISE EXCEPTION 'Cannot delete confirmed internal transaction';
    END IF;

    RETURN OLD;
END;
$$;

```

### 3.4.9 Функция «prevent\_exchange\_status\_change\_if\_banned»

```

CREATE OR REPLACE FUNCTION
prevent_exchange_status_change_if_banned()
RETURNS TRIGGER AS $$
DECLARE
    current_status_name VARCHAR;
BEGIN
    -- Получаем текущее имя статуса по ID
    SELECT status_name INTO current_status_name
    FROM statuses
    WHERE id = OLD.status_id;

    -- Если текущий статус "banned", блокируем изменение
    IF current_status_name = 'banned' THEN
        RAISE EXCEPTION 'Cannot change status of a banned exchange';
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

```

### 3.4.10 Функция «prevent\_status\_change\_if\_banned»

```

CREATE OR REPLACE FUNCTION prevent_status_change_if_banned()
RETURNS TRIGGER AS $$
DECLARE
    current_status_name VARCHAR;
BEGIN

```

```

-- Получаем текущее имя статуса пользователя
SELECT status_name INTO current_status_name
FROM statuses
WHERE id = OLD.status_id;

-- Если текущий статус — "banned", блокируем изменение
IF current_status_name = 'banned' THEN
    RAISE EXCEPTION 'Cannot change status for banned user';
END IF;

RETURN NEW;
END;
$$ LANGUAGE plpgsql;

```

#### 3.4.11 Функция «prevent\_wallet\_status\_change\_if\_banned»

```

CREATE OR REPLACE FUNCTION
prevent_wallet_status_change_if_banned()
RETURNS TRIGGER AS $$
DECLARE
    current_status_name VARCHAR;
BEGIN
    -- Получаем текущее имя статуса кошелька
    SELECT status_name INTO current_status_name
    FROM statuses
    WHERE id = OLD.status_id;

    -- Если статус "banned", запрещаем менять статус
    IF current_status_name = 'banned' THEN
        RAISE EXCEPTION 'Cannot change status of a banned wallet';
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

```

#### 3.4.12 Функция «set\_balances\_updated\_at»

```

CREATE OR REPLACE FUNCTION set_balances_updated_at()
RETURNS TRIGGER AS $$
BEGIN
    NEW.updated_at := now();
    RETURN NEW;
END;

```

```
$$ LANGUAGE plpgsql;
```

#### **3.4.13 Функция «set\_exchange\_tx\_updated\_at»**

```
CREATE OR REPLACE FUNCTION set_exchange_tx_updated_at()  
RETURNS TRIGGER AS $$  
BEGIN  
    NEW.updated_at := now();  
    RETURN NEW;  
END;  
$$ LANGUAGE plpgsql;
```

#### **3.4.14 Функция «set\_exchanges\_updated\_at»**

```
CREATE OR REPLACE FUNCTION set_exchanges_updated_at()  
RETURNS TRIGGER AS $$  
BEGIN  
    NEW.updated_at := now();  
    RETURN NEW;  
END;  
$$ LANGUAGE plpgsql;
```

#### **3.4.15 Функция «set\_internal\_tx\_updated\_at»**

```
CREATE OR REPLACE FUNCTION set_internal_tx_updated_at()  
RETURNS TRIGGER AS $$  
BEGIN  
    NEW.updated_at := now();  
    RETURN NEW;  
END;  
$$ LANGUAGE plpgsql;
```

#### **3.4.16 Функция «set\_users\_updated\_at»**

```
CREATE OR REPLACE FUNCTION set_users_updated_at()  
RETURNS TRIGGER AS $$  
BEGIN  
    NEW.updated_at := now();  
    RETURN NEW;  
END;  
$$ LANGUAGE plpgsql;
```



### 3.4.17 Функция «sync\_wallets\_status\_with\_user»

```
CREATE OR REPLACE FUNCTION sync_wallets_status_with_user()
RETURNS TRIGGER AS $$
BEGIN
    -- Проверка: если изменился только status_id
    IF NEW.status_id IS DISTINCT FROM OLD.status_id THEN
        -- Обновляем статус всех кошельков пользователя
        UPDATE wallets
        SET status_id = NEW.status_id,
            updated_at = now()
        WHERE user_username = NEW.username;
    END IF;

    RETURN NULL; -- AFTER триггер
END;
$$ LANGUAGE plpgsql;
```

### 3.4.18 Функция «update\_balances\_after\_confirmed\_internal\_tx»

```
CREATE OR REPLACE FUNCTION
update_balances_after_confirmed_internal_tx()
RETURNS TRIGGER AS $$
DECLARE
    new_status VARCHAR;
BEGIN
    -- Получаем новый статус
    SELECT transaction_status_name INTO new_status
    FROM transaction_statuses
    WHERE id = NEW.transaction_status_id;

    -- Обрабатываем только переход в confirmed
    IF new_status = 'confirmed' AND OLD.transaction_status_id <>
NEW.transaction_status_id THEN
        -- Списание с отправителя (amount + fee)
        UPDATE balances
        SET amount = amount - (NEW.amount + NEW.fee),
            updated_at = now()
        WHERE wallet_address = NEW.sender_address
            AND token_id = NEW.token_id;

        IF NOT FOUND THEN
            RAISE EXCEPTION 'Sender balance not found for wallet % and token
%', NEW.sender_address, NEW.token_id;
```

```

END IF;

-- Начисление получателю (только amount)
UPDATE balances
SET amount = amount + NEW.amount,
    updated_at = now()
WHERE wallet_address = NEW.recipient_address
    AND token_id = NEW.token_id;

IF NOT FOUND THEN
    -- Если записи нет, можно создать её, или выбросить ошибку.
    Здесь выбросим ошибку:
    RAISE EXCEPTION 'Recipient balance not found for wallet % and
token %', NEW.recipient_address, NEW.token_id;
END IF;
END IF;

RETURN NULL; -- AFTER триггер
END;
$$ LANGUAGE plpgsql;

```

### 3.4.19 Функция «update\_wallets\_updated\_at»

```

CREATE OR REPLACE FUNCTION update_wallets_updated_at()
RETURNS TRIGGER AS $$
BEGIN
    NEW.updated_at := now();
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

```

### 3.4.20 Функция «validate\_exchange\_transaction»

```

CREATE OR REPLACE FUNCTION validate_exchange_transaction()
RETURNS TRIGGER AS $$
DECLARE
    wallet_status_name VARCHAR;
    exchange_status_name VARCHAR;
    available_balance NUMERIC(20, 8);
BEGIN
    -- Проверка статуса кошелька
    SELECT s.status_name INTO wallet_status_name
    FROM wallets w
    JOIN statuses s ON w.status_id = s.id

```

```

WHERE w.address = NEW.wallet_address;

IF wallet_status_name <> 'active' THEN
    RAISE EXCEPTION 'Wallet must be active for exchange transaction';
END IF;

-- Проверка статуса биржи
SELECT s.status_name INTO exchange_status_name
FROM exchanges e
JOIN statuses s ON e.status_id = s.id
WHERE e.id = NEW.exchange_id;

IF exchange_status_name <> 'active' THEN
    RAISE EXCEPTION 'Exchange must be active for exchange
transaction';
END IF;

-- Проверка баланса
SELECT b.amount INTO available_balance
FROM balances b
WHERE b.wallet_address = NEW.wallet_address AND b.token_id =
NEW.token_id;

IF available_balance IS NULL THEN
    RAISE EXCEPTION 'No balance available for this wallet and token';
END IF;

IF NEW.amount + NEW.fee > available_balance THEN
    RAISE EXCEPTION 'Insufficient balance: amount + fee exceeds wallet
balance';
END IF;

-- Проверка комиссии
IF NEW.fee > NEW.amount * 0.05 THEN
    RAISE EXCEPTION 'Fee cannot exceed 5%% of the amount';
END IF;

RETURN NEW;
END;
$$ LANGUAGE plpgsql;

```

### **3.4.21 Функция «validate\_internal\_transaction»**

```

CREATE OR REPLACE FUNCTION validate_internal_transaction()

```

```

RETURNS TRIGGER AS $$
DECLARE
    sender_status TEXT;
    recipient_status TEXT;
    available_balance NUMERIC(20, 8);
BEGIN
    -- 1. Нельзя отправлять самому себе
    IF NEW.sender_address = NEW.recipient_address THEN
        RAISE EXCEPTION 'Cannot send tokens to yourself';
    END IF;

    -- 2. Статусы кошельков
    SELECT s.status_name INTO sender_status
    FROM wallets w
    JOIN statuses s ON w.status_id = s.id
    WHERE w.address = NEW.sender_address;

    SELECT s.status_name INTO recipient_status
    FROM wallets w
    JOIN statuses s ON w.status_id = s.id
    WHERE w.address = NEW.recipient_address;

    IF sender_status <> 'active' THEN
        RAISE EXCEPTION 'Sender wallet must be active';
    END IF;

    IF recipient_status <> 'active' THEN
        RAISE EXCEPTION 'Recipient wallet must be active';
    END IF;

    -- 3. Проверка баланса
    SELECT b.amount INTO available_balance
    FROM balances b
    WHERE b.wallet_address = NEW.sender_address
    AND b.token_id = NEW.token_id;

    IF available_balance IS NULL THEN
        RAISE EXCEPTION 'No balance available for sender wallet and token';
    END IF;

    IF NEW.amount + NEW.fee > available_balance THEN
        RAISE EXCEPTION 'Insufficient balance: amount + fee exceeds sender
balance';
    END IF;

```

```

-- 4. Проверка комиссии
IF NEW.fee > NEW.amount * 0.05 THEN
    RAISE EXCEPTION 'Fee cannot exceed 5%% of the amount';
END IF;

RETURN NEW;
END;
$$ LANGUAGE plpgsql;

```

### 3.4.22 Функция «*validate\_onion\_url*»

```

CREATE OR REPLACE FUNCTION validate_onion_url()
RETURNS TRIGGER AS $$
BEGIN
    -- Проверка, что url заканчивается на '.onion'
    IF RIGHT(NEW.url, 6) <> '.onion' THEN
        RAISE EXCEPTION 'Exchange URL must be a valid .onion address';
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

```

### 3.5 Создание пользователей и ролей

В *PostgreSQL* управление правами доступа реализовано через систему ролей. Каждая роль может выполнять функции отдельного пользователя (если для нее установлен атрибут *LOGIN*) или группы пользователей. Роли обладают широкими возможностями: они могут быть владельцами объектов базы данных (таблиц, представлений, процедур) и управлять доступом к этим объектам, предоставляя или отзывая разрешения у других ролей. Особенностью системы является возможность включения одной роли в состав другой — это позволяет делегировать права доступа через иерархию ролей.

Возможность создания и удаления ролей определяется наличием права *CREATEROLE*. Создание ролей производится при помощи оператора *CREATE ROLE*, а удаление при помощи *DROP ROLE*.

В рамках базы данных были созданы следующие роли:

1) Администратор базы данных (*design*)

Роль, обладающая доступом ко всем объектам базы данных. Является владельцем базы данных и всех объектов внутри неё, за исключением функций.

Атрибуты: *NOSUPERUSER*, *NOCREATEDB*, *CREATEROLE*, *NOINHERIT*, *LOGIN*, *NOREPLICATION*, *BYPASSRLS*, *CONNECTION LIMIT 1*;

2) Менеджер аккаунтов (*account\_manager*)

Групповая роль, управляет пользователями платформы. Имеет доступ к таблице *users*, *wallets*, *balances* и связанными с ними данными. Не обладает правами на изменение структуры базы данных или работу с ролями.

Атрибуты: *NOLOGIN*, *NOSUPERUSER*, *NOCREATEDB*, *NOCREATEROLE*, *INHERIT*, *NOREPLICATION*;

3) Менеджер обменников (*exchange\_manager*)

Групповая роль, отвечает за взаимодействие с внешними биржами. Имеет доступ к *exchange\_transactions*, информации о токенах, кошельках и биржах.

Атрибуты: *NOLOGIN*, *NOSUPERUSER*, *NOCREATEDB*, *NOCREATEROLE*, *INHERIT*, *NOREPLICATION*;

4) Менеджер внутренних переводов (*internal\_manager*)

Групповая роль, контролирует внутренние переводы между кошельками пользователей. Работает с таблицей *internal\_transactions* и связанными сущностями (кошельки, токены, статусы).

Атрибуты: *NOLOGIN*, *NOSUPERUSER*, *NOCREATEDB*, *NOCREATEROLE*, *INHERIT*, *NOREPLICATION*;

5) Помощник (*assistant*)

Групповая роль, имеет ограниченный доступ для выполнения вспомогательных задач: просмотр статусов, токенов, информации о пользователях и кошельках. Не может изменять данные или управлять объектами базы.

Атрибуты: *NOLOGIN*, *NOSUPERUSER*, *NOCREATEDB*, *NOCREATEROLE*, *INHERIT*, *NOREPLICATION*;

### 3.6 Назначение прав субъектам доступа

Для управления правами доступа к объектам базы данных в *PostgreSQL* используются операторы *GRANT* (предоставление прав) и *REVOKE* (отзыв прав). Основные виды привилегий для работы с таблицами и другими объектами включают:

1) *INSERT* – позволяет добавить новую строку в таблицу, представление. А также позволяет использовать *COPY FROM*.

2) *SELECT* – позволяет выводить данные из любого столбца или определенных столбцов таблицы, представления, материализованного представления или другого табличного объекта. Также позволяет использовать *COPY TO*;

3) *UPDATE* – разрешает обновление любого столбца или определенных столбцов таблицы, представления;

4) *DELETE* – позволяет удалять строку из таблицы, представления;

5) *TRUNCATE* – позволяет полностью очистить таблицу;

6) *REFERENCES* – позволяет создавать внешние ключи на таблицу;

7) *TRIGGER* – позволяет создать триггер для таблицы;

Перед настройкой прав доступа к пользовательским таблицам была выполнена операция отзыва всех привилегий в данной базе данных у *PUBLIC*.

#### 3.6.1 Описание привилегий ролей

Таблица 13 – описание привилегий *Design*

	<i>SELECT</i>	<i>INSERT</i>	<i>UPDATE</i>	<i>DELETE</i>	<i>TRUNCATE</i>	<i>REFERENCES</i>	<i>TRIGGER</i>
<i>balances</i>	*	*	*	*	*	*	*
<i>blockchains</i>	*	*	*	*	*	*	*
<i>exchange_transactions</i>	*	*	*	*	*	*	*
<i>exchanges</i>	*	*	*	*	*	*	*
<i>internal_transactions</i>	*	*	*	*	*	*	*
<i>statuses</i>	*	*	*	*	*	*	*
<i>tokens</i>	*	*	*	*	*	*	*
<i>transaction_statuses</i>	*	*	*	*	*	*	*
<i>users</i>	*	*	*	*	*	*	*
<i>wallets</i>	*	*	*	*	*	*	*
<i>view_account_data</i>	*	*	*	*	*	*	*
<i>view_exchange_transactions_full</i>	*	*	*	*	*	*	*
<i>view_internal_transactions_full</i>	*	*	*	*	*	*	*
<i>view_token_info</i>	*	*	*	*	*	*	*
<i>view_transactions_all</i>	*	*	*	*	*	*	*

Таблица 14 – описание привилегий *Account\_manager*

	SELE CT	INSE RT	UPDA TE	DELE TE	TRUNC ATE	REFEREN CES	TRIGG ER
<i>balances</i>	*	*	*	*			
<i>blockchains</i>	*						
<i>exchange_transactions</i>							
<i>exchanges</i>							
<i>internal_transactions</i>							
<i>statuses</i>	*						
<i>tokens</i>	*						
<i>transaction_statuses</i>							
<i>users</i>	*	*	*	*			
<i>wallets</i>	*	*	*	*			
<i>view_account_data</i>	*						
<i>view_exchange_transactions_full</i>							
<i>view_internal_transactions_full</i>							
<i>view_token_info</i>	*						
<i>view_transactions_all</i>							

Таблица 15 – описание привилегий *Exchange\_manager*

	SELE CT	INSE RT	UPDA TE	DELE TE	TRUNC ATE	REFEREN CES	TRIGG ER
<i>balances</i>	*		*				
<i>blockchains</i>	*						
<i>exchange_transactions</i>	*	*	*	*			
<i>exchanges</i>	*	*	*	*			
<i>internal_transactions</i>							
<i>statuses</i>	*						
<i>tokens</i>	*						
<i>transaction_statuses</i>	*						
<i>users</i>	*						
<i>wallets</i>	*						
<i>view_account_data</i>							
<i>view_exchange_transactions_full</i>	*						
<i>view_internal_transactions_full</i>							
<i>view_token_info</i>	*						
<i>view_transactions_all</i>							



Таблица 16 – описание привилегий *Internal\_manager*

	SELE CT	INSE RT	UPDA TE	DELE TE	TRUNC ATE	REFEREN CES	TRIGG ER
<i>balances</i>	*		*				
<i>blockchains</i>	*						
<i>exchange_transactions</i>							
<i>exchanges</i>							
<i>internal_transactions</i>	*	*	*	*			
<i>statuses</i>	*						
<i>tokens</i>	*						
<i>transaction_statuses</i>	*						
<i>users</i>	*						
<i>wallets</i>	*						
<i>view_account_data</i>	*						
<i>view_exchange_transactions_full</i>							
<i>view_internal_transactions_full</i>	*						
<i>view_token_info</i>	*						
<i>view_transactions_all</i>							

Таблица 17 – описание привилегий *Assistant*

	SELE CT	INSE RT	UPDA TE	DELE TE	TRUNC ATE	REFEREN CES	TRIGG ER
<i>balances</i>	*						
<i>blockchains</i>	*						
<i>exchange_transactions</i>	*						
<i>exchanges</i>	*						
<i>internal_transactions</i>	*						
<i>statuses</i>	*						
<i>tokens</i>	*						
<i>transaction_statuses</i>	*						
<i>users</i>	*						
<i>wallets</i>	*						
<i>view_account_data</i>	*						
<i>view_exchange_transactions_full</i>	*						
<i>view_internal_transactions_full</i>	*						
<i>view_token_info</i>	*						
<i>view_transactions_all</i>	*						

### 3.6.2 Назначение привилегий ролям

Для назначения привилегий роли *Design* используются следующие SQL запросы:

*GRANT ALL ON TABLE balances TO design WITH GRANT OPTION;*  
*GRANT ALL ON TABLE blockchains TO design WITH GRANT OPTION;*

GRANT ALL ON TABLE *exchange\_transactions* TO *design* WITH GRANT OPTION;  
 GRANT ALL ON TABLE *exchanges* TO *design* WITH GRANT OPTION;  
 GRANT ALL ON TABLE *internal\_transactions* TO *design* WITH GRANT OPTION;  
 GRANT ALL ON TABLE *statuses* TO *design* WITH GRANT OPTION;  
 GRANT ALL ON TABLE *tokens* TO *design* WITH GRANT OPTION;  
 GRANT ALL ON TABLE *transaction\_statuses* TO *design* WITH GRANT OPTION;  
 GRANT ALL ON TABLE *users* TO *design* WITH GRANT OPTION;  
 GRANT ALL ON TABLE *wallets* TO *design* WITH GRANT OPTION;  
 GRANT ALL ON TABLE *view\_account\_data* TO *design* WITH GRANT OPTION;  
 GRANT ALL ON TABLE *view\_exchange\_transactions\_full* TO *design* WITH GRANT OPTION;  
 GRANT ALL ON TABLE *view\_internal\_transactions\_full* TO *design* WITH GRANT OPTION;  
 GRANT ALL ON TABLE *view\_token\_info* TO *design* WITH GRANT OPTION;  
 GRANT ALL ON TABLE *view\_transactions\_all* TO *design* WITH GRANT OPTION;

Для назначения привилегий роли *Account\_manager* используются следующие SQL запросы:

GRANT SELECT, INSERT, UPDATE, DELETE ON TABLE *balances* TO *account\_manager*;  
 GRANT SELECT ON TABLE *blockchains* TO *account\_manager*;  
 GRANT SELECT ON TABLE *statuses* TO *account\_manager*;  
 GRANT SELECT ON TABLE *tokens* TO *account\_manager*;  
 GRANT SELECT, INSERT, UPDATE, DELETE ON TABLE *users* TO *account\_manager*;  
 GRANT SELECT, INSERT, UPDATE, DELETE ON TABLE *wallets* TO *account\_manager*;  
 GRANT SELECT ON TABLE *view\_account\_data* TO *account\_manager*;  
 GRANT SELECT ON TABLE *view\_token\_info* TO *account\_manager*;

Для назначения привилегий роли *Exchange\_manager* используются следующие SQL запросы:

GRANT SELECT, UPDATE ON TABLE *balances* TO *exchange\_manager*;  
 GRANT SELECT ON TABLE *blockchains* TO *exchange\_manager*;  
 GRANT SELECT, INSERT, UPDATE, DELETE ON TABLE *exchange\_transactions* TO *exchange\_manager*;  
 GRANT SELECT, INSERT, UPDATE, DELETE ON TABLE *exchanges* TO *exchange\_manager*;

```

GRANT SELECT ON TABLE statuses TO exchange_manager;
GRANT SELECT ON TABLE tokens TO exchange_manager;
GRANT SELECT ON TABLE transaction_statuses TO exchange_manager;
GRANT SELECT ON TABLE users TO exchange_manager;
GRANT SELECT ON TABLE wallets TO exchange_manager;
GRANT SELECT ON TABLE view_exchange_transactions_full TO
exchange_manager;
GRANT SELECT ON TABLE view_token_info TO exchange_manager;

```

Для назначения привилегий роли *Internal\_manager* используются следующие SQL запросы:

```

GRANT SELECT, UPDATE ON TABLE balances TO internal_manager;
GRANT SELECT ON TABLE blockchains TO internal_manager;
GRANT SELECT, INSERT, UPDATE, DELETE ON TABLE
internal_transactions TO internal_manager;
GRANT SELECT ON TABLE statuses TO internal_manager;
GRANT SELECT ON TABLE tokens TO internal_manager;
GRANT SELECT ON TABLE transaction_statuses TO internal_manager;
GRANT SELECT ON TABLE users TO internal_manager;
GRANT SELECT ON TABLE wallets TO internal_manager;
GRANT SELECT ON TABLE view_account_data TO internal_manager;
GRANT SELECT ON TABLE view_internal_transactions_full TO
internal_manager;
GRANT SELECT ON TABLE view_token_info TO internal_manager;

```

Для назначения привилегий роли *Assistant* используются следующие SQL запросы:

```

GRANT SELECT ON TABLE balances TO assistant;
GRANT SELECT ON TABLE blockchains TO assistant;
GRANT SELECT ON TABLE exchange_transactions TO assistant;
GRANT SELECT ON TABLE exchanges TO assistant;
GRANT SELECT ON TABLE internal_transactions TO assistant;
GRANT SELECT ON TABLE statuses TO assistant;
GRANT SELECT ON TABLE tokens TO assistant;
GRANT SELECT ON TABLE transaction_statuses TO assistant;
GRANT SELECT ON TABLE users TO assistant;
GRANT SELECT ON TABLE wallets TO assistant;
GRANT SELECT ON TABLE view_account_data TO assistant;
GRANT SELECT ON TABLE view_exchange_transactions_full TO assistant;
GRANT SELECT ON TABLE view_internal_transactions_full TO assistant;
GRANT SELECT ON TABLE view_token_info TO assistant;
GRANT SELECT ON TABLE view_transactions_all TO assistant;

```

Для каждой групповой роли были созданы пользователи, SQL запрос:

```
CREATE ROLE user_account_1 LOGIN PASSWORD 'secure_password_1';
GRANT account_manager TO user_account_1 WITH set false;
```

```
CREATE ROLE user_exchange_1 LOGIN PASSWORD
'secure_password_2';
GRANT exchange_manager TO user_exchange_1 WITH set false;
```

```
CREATE ROLE user_internal_1 LOGIN PASSWORD 'secure_password_3';
GRANT internal_manager TO user_internal_1 WITH set false;
```

```
CREATE ROLE user_assistant_1 LOGIN PASSWORD 'secure_password_4';
GRANT assistant TO user_assistant_1 WITH set false;
```

### 3.6.3 Аудит определенных для ролей привилегий

Для обеспечения надежной защиты данных платформы был проведён комплексный аудит безопасности базы данных с акцентом на ролевую модель и управление правами доступа.

В ходе аудита были выполнены следующие действия:

#### 1. Проверка ролей и пользователей

Были проанализированы все существующие роли и пользователи в базе данных. В системе реализованы основные групповые роли: *design* (администратор базы данных), *account\_manager* (менеджер аккаунтов), *exchange\_manager* (менеджер обменников), *internal\_manager* (менеджер внутренних переводов) и *assistant* (помощник). Каждому пользователю назначена соответствующая групповая роль, что обеспечивает централизованное и удобное управление правами доступа.

#### 2. Анализ прав доступа

Проведена проверка привилегий, назначенных ролям и пользователям, с использованием системных представлений *information\_schema.role\_table\_grants* и *pg\_auth\_members*. Результаты показали, что права доступа чётко разграничены согласно бизнес-логике:

- Администратор базы данных (*design*) имеет полный доступ ко всем объектам базы, кроме функций и процедур.
- Менеджер аккаунтов (*account\_manager*) управляет данными пользователей, кошельков и балансами, не имея прав на изменение структуры базы и ролей.
- Менеджер обменников (*exchange\_manager*) отвечает за взаимодействие с внешними биржами и транзакциями, связанными с ними.
- Менеджер внутренних переводов (*internal\_manager*) контролирует внутренние переводы между кошельками пользователей.
- Помощник (*assistant*) обладает ограниченным доступом для просмотра информации и выполнения вспомогательных задач, не имея прав на изменение данных.

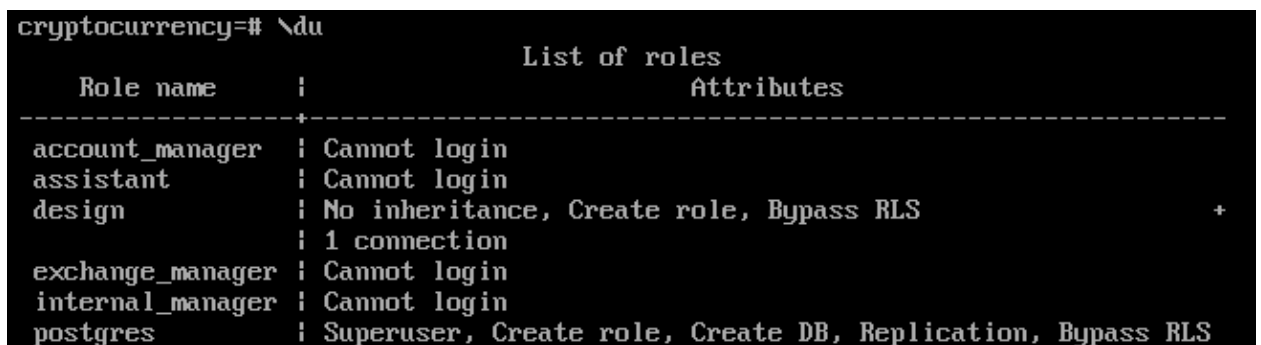
### 3. Отсутствие избыточных привилегий

Проведён анализ, подтверждающий, что ни один пользователь или роль не имеет прав, выходящих за рамки своей ответственности. Это снижает риск несанкционированного доступа или случайного изменения критически важных данных.

### 4. Проверка логирования и мониторинга

В базе данных настроено логирование действий пользователей и операций, что позволяет отслеживать активность и своевременно выявлять потенциальные угрозы.

Проведённый аудит безопасности показал, что система ролевого разграничения доступа реализована корректно и соответствует требованиям безопасности. Пользователи имеют только необходимые права для выполнения своих функций, что минимизирует риски несанкционированного доступа и обеспечивает надёжную защиту данных платформы (рисунок 13, 14).



```
cryptocurrency=# \du
```

Role name	Attributes
account_manager	Cannot login
assistant	Cannot login
design	No inheritance, Create role, Bypass RLS 1 connection
exchange_manager	Cannot login
internal_manager	Cannot login
postgres	Superuser, Create role, Create DB, Replication, Bypass RLS

Рисунок 13 – список всех пользователей базы данных

privilege_type	table_schema	table_name
account_manager	public	balances
account_manager	public	balances
account_manager	public	balances
account_manager	public	balances
account_manager	public	blockchains
account_manager	public	statuses
account_manager	public	tokens
account_manager	public	users
account_manager	public	users
account_manager	public	users
account_manager	public	users
account_manager	public	view_account_data
account_manager	public	view_token_info
account_manager	public	wallets
account_manager	public	wallets
account_manager	public	wallets
account_manager	public	wallets
assistant	public	balances
assistant	public	blockchains
assistant	public	exchange_transactions
assistant	public	exchanges
assistant	public	internal_transactions
assistant	public	statuses
assistant	public	tokens
assistant	public	transaction_statuses
assistant	public	users
assistant	public	view_account_data
assistant	public	view_exchange_transactions_full
assistant	public	view_internal_transactions_full
assistant	public	view_token_info
assistant	public	view_transactions_all
assistant	public	wallets
design	public	balances
design	public	balances
design	public	balances
design	public	balances
design	public	balances
design	public	balances
design	public	balances
design	public	blockchains
design	public	blockchains
design	public	blockchains
design	public	blockchains
design	public	blockchains
design	public	blockchains
design	public	blockchains
design	public	exchange_transactions
--More--		

Рисунок 14 – фрагмент перечня прав доступа всех ролей к таблицам пользовательских схем в базе данных

### 3.7 Аутентификация клиентов при соединении с базой данных

Аутентификация в СУБД *PostgreSQL* управляется с помощью конфигурационного файла *pg\_hba.conf*. Этот файл определяет правила доступа к серверу, регулируя, какие клиенты могут подключаться, к каким базам данных и с какими методами аутентификации.

Формат файла *pg\_hba.conf* представляет собой набор записей, каждая из которых занимает отдельную строку. Пустые строки, а также комментарии (обозначаемые символом #) игнорируются. Если запись слишком длинная, её можно перенести на следующую строку, поставив в конце обратную косую черту (\).

Каждая запись состоит из нескольких полей, разделённых пробелами или табуляцией. Эти поля определяют:

- 1) Тип соединения;
- 2) Имя базы данных, к которой запрашивается доступ;
- 3) Имя пользователя, пытающегося подключиться;
- 4) Диапазон IP-адресов клиента (если соединение сетевое);
- 5) Метод аутентификации, который будет использоваться для проверки подлинности;

При попытке подключения клиента *PostgreSQL* последовательно проверяет записи в *pg\_hba.conf* и применяет первую подходящую запись, соответствующую типу соединения, IP-адресу клиента, базе данных и пользователю. Если ни одна запись не совпадает, доступ автоматически отклоняется.

Для разрабатываемой базы данных конфигурационный файл был настроен следующим образом (рисунок 15).

local	cryptocurrency	design	scram-sha-256
hostssl	cryptocurrency	+account_manager 192.168.153.0/24	cert
hostssl	cryptocurrency	+exchange_manager 192.168.153.0/24	cert
hostssl	cryptocurrency	+internal_manager 192.168.153.0/24	cert
hostssl	cryptocurrency	+assistant 192.168.153.0/24	cert
hostssl	cryptocurrency	design 192.168.153.0/24	cert

Рисунок 15 – фрагмент перечня прав доступа всех ролей к таблицам пользовательских схем в базе данных

### 3.8 Защита канала связи между СУБД и клиентом

Для обеспечения безопасности передачи данных между клиентом и сервером *PostgreSQL* было настроено SSL-шифрование. Этот механизм предотвращает перехват информации при её передаче по сети, что особенно важно при работе в корпоративных системах. *PostgreSQL* поддерживает SSL-соединения на уровне протоколов TLS, предоставляя встроенные средства для защиты канала связи.

Реализация SSL-шифрования в *PostgreSQL* основана на библиотеке *libpq*, которая управляет подключениями и загружает настройки из конфигурационного файла *OpenSSL*.

Чтобы настроить защищенное соединение необходимо подготовить:

- 1) SSL-сертификат сервера;
- 2) Приватный ключ для шифрования данных;
- 3) Корневой сертификат;
- 4) Приватный ключ и сертификат клиента;

Для создания сертификата для сервера были использованы следующие команды:

`openssl genrsa -aes256 -out server.key 4096` – эта команда создает приватный ключ RSA длиной 4096 бит для сервера.

`openssl req -new -key server.key -out server.csr` – данная команда создает запрос на подпись сертификата для сервера.

`openssl x509 -req -days 3650 -in server.csr -CA caa.crt -CAkey caa.key -CAcreateserial -out server.crt` – с помощью данной команды был подписан CSR сервера.

Для создания клиентских сертификатов были использованы следующие команды:

`openssl req -new -nodes -out client_director.csr -keyout client_director.key -subj "/CN=director"`, где *CN* – это название роли в базе данных. Данная команда создает клиентский приватный ключ, а также создает запрос на подписание клиентского сертификата.

`openssl x509 -req -in client_director.csr -CA /var/lib/pgsql/17/data/ca.crt -CAkey /var/lib/pgsql/17/data/ca.key -CAcreateserial -out client_director.crt -days 365` – данная команда подписывает клиентский сертификат, который будет использоваться клиентом при подключении.

Затем была произведена настройка в конфигурационном файле *postgresql.conf* (рисунок 16).



```
# - SSL -

ssl = on
ssl_ca_file = '/var/lib/pgsql/17/data/caa.crt'
ssl_cert_file = '/var/lib/pgsql/17/data/server.crt'
#ssl_crl_file = ''
#ssl_crl_dir = ''
ssl_key_file = '/var/lib/pgsql/17/data/server.key'
#ssl_ciphers = 'HIGH:MEDIUM:+3DES:!aNULL'<-----># allowed SSL ciphers
#ssl_prefer_server_ciphers = on
#ssl_ecdh_curve = 'prime256v1'
#ssl_min_protocol_version = 'TLSv1.2'
#ssl_max_protocol_version = ''
#ssl_dh_params_file = ''
#ssl_passphrase_command = ''
#ssl_passphrase_command_supports_reload = off
```

Рисунок 16 – Настройка SSL в *postgresql.conf*

Для проверки SSL соединения была использована программа *WireShark* (рисунок 17).

21	2.850832	192.168.153.140	192.168.153.1	TLSv1.3	119 Application Data
22	2.851150	192.168.153.1	192.168.153.140	TLSv1.3	148 Application Data
23	2.851418	192.168.153.140	192.168.153.1	TLSv1.3	169 Application Data
24	2.852590	192.168.153.1	192.168.153.140	TLSv1.3	201 Application Data
25	2.853703	192.168.153.140	192.168.153.1	TLSv1.3	551 Application Data
26	2.854092	192.168.153.1	192.168.153.140	TLSv1.3	155 Application Data
27	2.854387	192.168.153.140	192.168.153.1	TLSv1.3	101 Application Data
28	2.854544	192.168.153.1	192.168.153.140	TLSv1.3	176 Application Data
29	2.854716	192.168.153.140	192.168.153.1	TLSv1.3	146 Application Data
30	2.855165	192.168.153.1	192.168.153.140	TLSv1.3	200 Application Data
31	2.855396	192.168.153.140	192.168.153.1	TLSv1.3	170 Application Data
32	2.855625	192.168.153.1	192.168.153.140	TLSv1.3	275 Application Data
33	2.856672	192.168.153.140	192.168.153.1	TLSv1.3	921 Application Data
34	2.856924	192.168.153.1	192.168.153.140	TLSv1.3	161 Application Data
35	2.857200	192.168.153.140	192.168.153.1	TLSv1.3	167 Application Data
36	2.857364	192.168.153.1	192.168.153.140	TLSv1.3	97 Application Data
37	2.857423	192.168.153.1	192.168.153.140	TLSv1.3	94 Application Data
38	2.857437	192.168.153.1	192.168.153.140	TLSv1.3	94 Application Data

Рисунок 17 – Проверка шифрования при помощи утилиты *WireShark*

### 3.9 Организация журналирования

При обычной работе сервера происходит постоянная, но последовательная запись журнальных файлов.

Журналирование в *PostgreSQL* выполняет несколько важных функций, необходимых для стабильной и безопасной работы базы данных. В первую очередь, оно позволяет отслеживать все подключения к СУБД, фиксируя информацию о том, кто, когда получил доступ к системе. Это помогает администраторам выявлять несанкционированные попытки входа, анализировать активность пользователей и оперативно реагировать на потенциальные угрозы безопасности.

Кроме того, журнал операций сохраняет историю изменений данных, что особенно важно при расследовании инцидентов или выявлении причин ошибок.

Журналирование также используется для оптимизации производительности – анализ запросов помогает находить "узкие места" в работе базы данных и выявлять наиболее ресурсоемкие операции.

В *PostgreSQL* механизмы журналирования гибко настраиваются, позволяя выбирать уровень детализации логируемых событий в зависимости от конкретных задач.

Грамотно настроенное журналирование является неотъемлемой частью эксплуатации промышленных баз данных, сочетая в себе функции мониторинга, безопасности, восстановления и анализа производительности системы.

Для детальной настройки журналирования необходимо отредактировать файл конфигурации *postgresql.conf*, задав следующие параметры:

- 1) *log\_destination* – параметр, отвечающий за формат журнальных файлов. В данной базе данных используется формат *stderr*;
- 2) *logging\_collector* – включает сборщик сообщений, который собирает отправленные в *stderr* сообщения и перенаправляет их в журнальные файлы;
- 3) *log\_directory* – определяет каталог, в котором создаются журнальные файлы;
- 4) *log\_filename* – определяет имена журнальных файлов;
- 5) *log\_file\_mode* – определяет права доступа к журнальным файлам, при включенном *logging\_collector*. В рамках данной базы данных право на чтение и запись в журнальные файлы имеет только администратор;
- 6) *log\_rotation\_age* – определяет время жизни отдельного журнального файла, по истечении которого создается новый файл;
- 7) *log\_rotation\_size* – определяет максимальный размер отдельного журнального файла, при достижении которого создается новый файл;
- 8) *log\_line\_prefix* – параметр, определяющий строку, отображаемую в каждой строке в начале;
- 9) *log\_connections* – включает протоколирование всех попыток подключения к серверу;
- 10) *log\_disconnections* – включает протоколирование завершения сеанса;
- 11) *log\_statement* – управляет тем, какие *SQL*-команды будут записываться в журнал. Большую ценность представляют команды, отвечающие за манипуляцию данными;

Параметры, измененные в файле конфигурации *postgresql.conf*:

- *log\_statement* = *all*;
- *log\_connections* = *on*;
- *log\_disconnections* = *on*;
- *log\_directory* = *'log'*;
- *log\_filename* = *'postgresql-%a.log'*.

Результат на рисунке 17.

```

025-06-18 23:26:11.564 +07 [3096] LOG: execute <unnamed>: select * from balances
025-06-18 23:26:11.771 +07 [3096] LOG: execute <unnamed>: select * from balances
025-06-18 23:26:11.927 +07 [3096] LOG: execute <unnamed>: select * from balances
025-06-18 23:26:12.062 +07 [3096] LOG: execute <unnamed>: select * from balances
025-06-18 23:26:12.219 +07 [3096] LOG: execute <unnamed>: select * from balances
025-06-18 23:26:12.369 +07 [3096] LOG: execute <unnamed>: select * from balances
025-06-18 23:26:12.543 +07 [3096] LOG: execute <unnamed>: select * from balances

```

Рисунок 17 – Просмотр одного из файла журналирования

### 3.10 Организация шифрования данных

Шифрование пользовательских данных, реализуется непосредственно на стороне клиента, до их передачи в систему. Таким образом, данные передаются уже в зашифрованном виде, что исключает необходимость отдельного шифрования при хранении на сервере.

Это решение обеспечивает конфиденциальность аутентификационной информации и снижает потенциальные риски, связанные с утечкой данных на стороне сервера.

### 3.11 Резервное копирование и восстановление данных

Корректное резервное копирование базы данных *PostgreSQL* компактно сохраняет метаданные и данные базы данных в файле на жестком диске. Настоятельно рекомендуется перезаписывать файлы резервных копий на сменные носители и хранить в физически защищенном месте в стороне от сервера.

Существует разные подходы к резервному копированию данных в *PostgreSQL*:

- выгрузка в *SQL*;
- копирование на уровне файлов;

Первый способ заключается в генерации текстового файла с командами *SQL*, которые при выполнении на сервере пересоздадут базу данных в том же самом состоянии, в котором она была на момент выгрузки.

Второй способ заключается в непосредственном копировании файлов, в которых *PostgreSQL* хранит содержимое базы данных.

*PostgreSQL* предоставляет для этой цели вспомогательную утилиту *pg\_dump*. Утилита *pg\_dump* является для *PostgreSQL* обычным клиентским приложением, это означает, что существует возможность выполнять процедуру резервного копирования с любого удалённого компьютера, если имеете доступ к нужной базе данных.

Программа *pg\_dump* выгружает только одну базу данных в один момент времени и не включает в дампы информацию о ролях и табличных пространствах (так как это информация уровня кластера, а не самой базы данных). Для удобства создания дампа всего содержимого кластера баз данных предоставляется программа *pg\_dumpall*, которая делает резервную

копию всех баз данных кластера, а также сохраняет данные уровня кластера, такие как роли и определения табличных пространств.

Чтобы создать резервную копию, необходимо выполнить команду, показанную на рисунке 18.

```
C:\Program Files\PostgreSQL\17\bin>pg_dump.exe -U design cryptocurrency > C:\Users\User\Desktop\backp.dump
Пароль:
```

Рисунок 18 – Команда для создание резервной копии

Для восстановления базы данных из резервной копии с необходимо выполнить команду, показанную на рисунке 19.

```
C:\Program Files\PostgreSQL\17\bin>psql.exe -U design < C:\Users\User\Desktop\backp.dump
Пароль пользователя design:
```

Рисунок 19 – Команда для восстановления базы данных

Существует возможность создавать резервные копии БД (рисунок 20) и восстанавливать БД из резервных копий (рисунок 21) с помощью самой СУБД *dBeaver*.

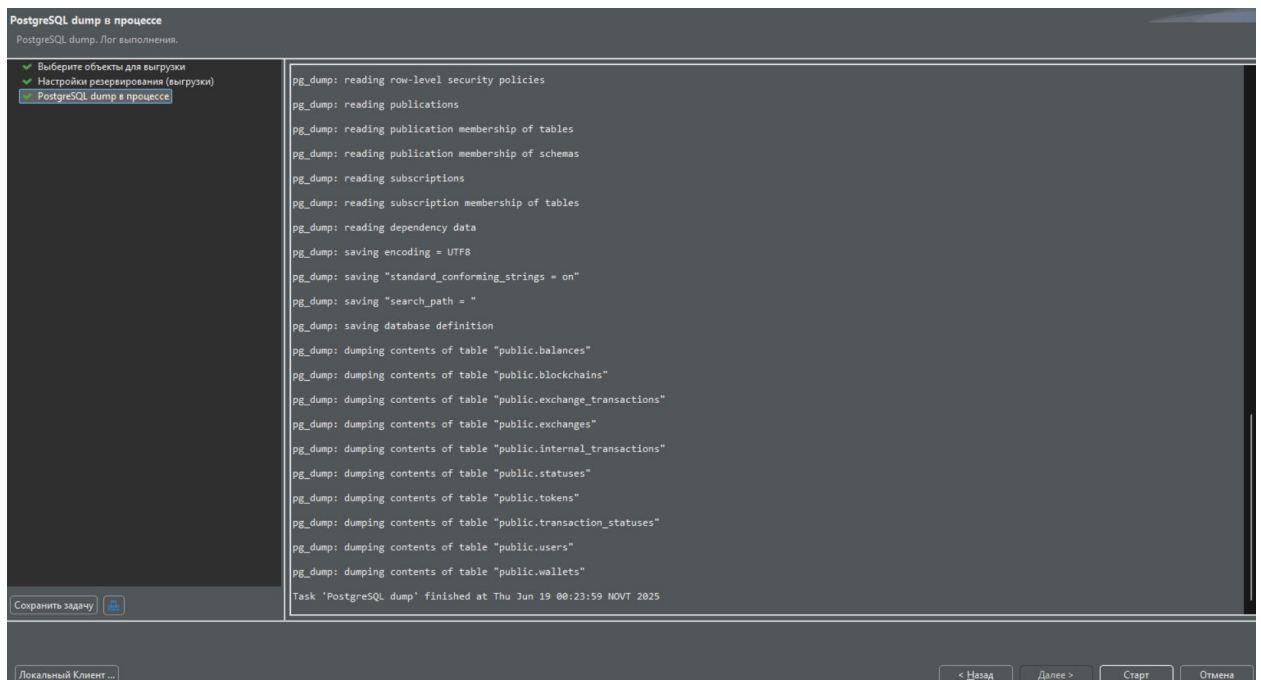


Рисунок 20 – Настройка резервной копии базы данных

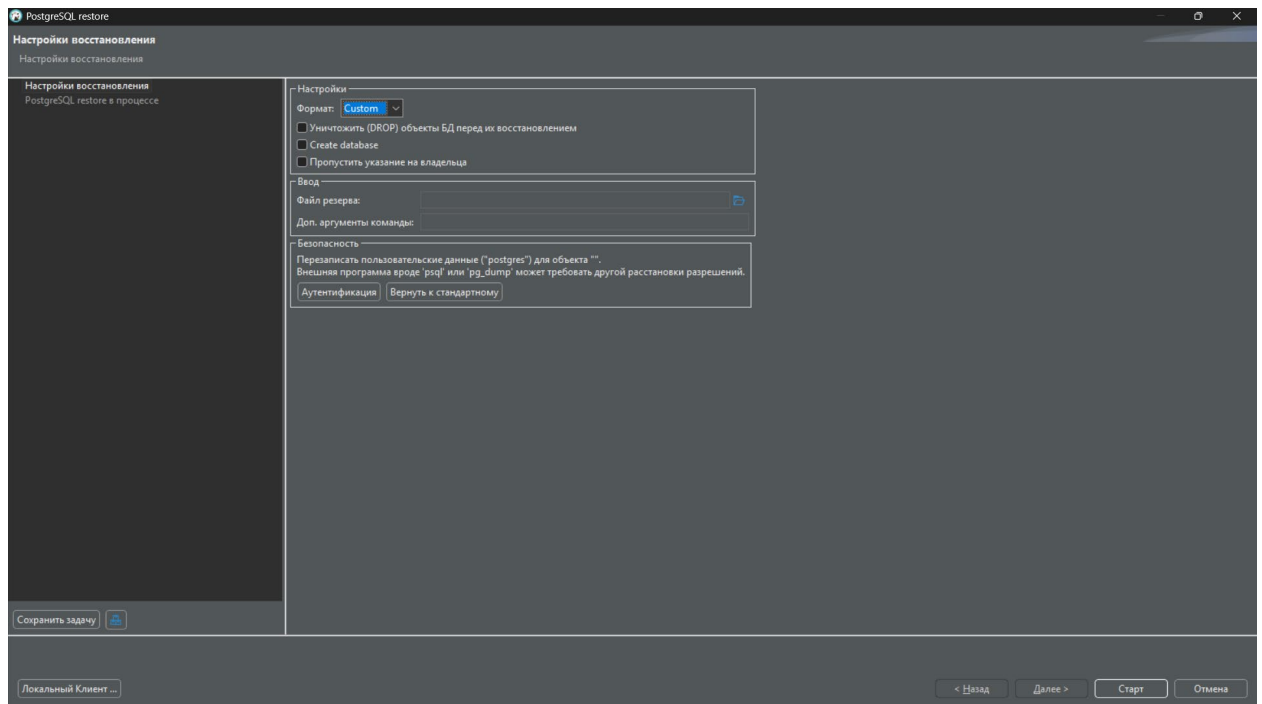


Рисунок 21 – Процесс восстановления БД из резервной копии

## **ЗАКЛЮЧЕНИЕ**

Целью курсовой работы является автоматизация процесса для автоматизации учета криптовалютных кошельков и осуществляемых транзакций, с акцентом на обеспечение целостности данных.

В ходе курсовой работы данная цель была достигнута в полном объеме и выполнены следующие задачи:

- разработаны и описаны внутримашинная информационная база;
- разработаны и описаны немашинная информационная база;
- реализованы меры по защите БД.

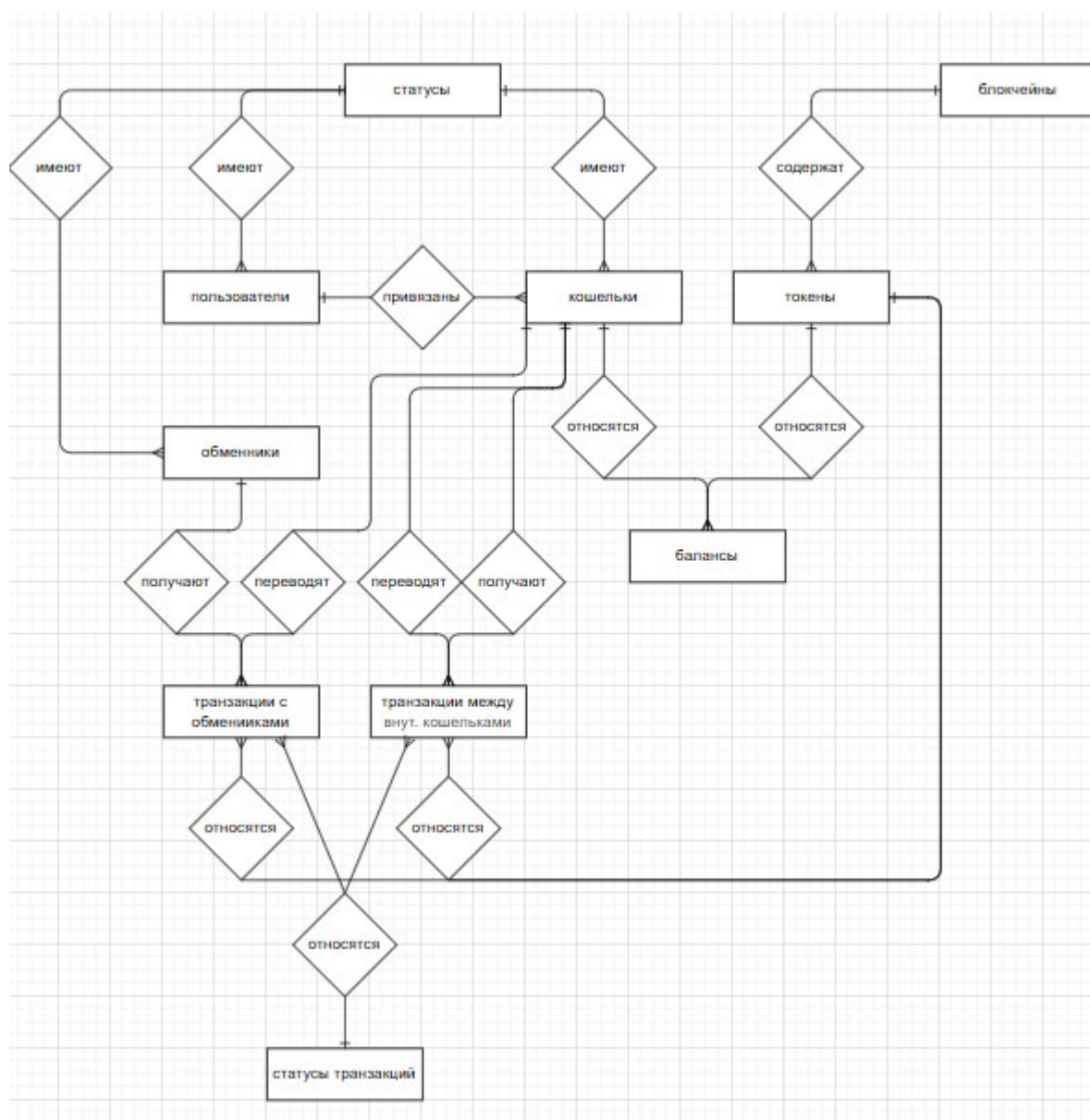
## БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Composite Primary Keys in PostgreSQL* – сайт. URL: <https://www.commandprompt.com/education/composite-primary-keys-in-postgresql/> (дата обращения: 17.05.2025).
2. *Postgres: How to do Composite keys?* – сайт. URL: <https://stackoverflow.com/questions/1285967/postgres-how-to-do-composite-keys> (дата обращения: 17.05.2025).
3. *PostgreSQL : Документация: 14: Глава 8. Типы данных* – сайт. URL: <https://postgrespro.ru/docs/postgresql/14/datatype?ysclid=lxbl52jheq302681342> (дата обращения: 17.05.2025).
4. *Обеспечение безопасности базы данных PostgreSQL* – сайт. URL: <https://habr.com/ru/post/550882/> (дата обращения: 17.05.2025).
5. *PostgreSQL: Документация: 16: 43.10. Триггерные функции* – сайт. URL: <https://postgrespro.ru/docs/postgresql/16/plpgsql-trigger> (дата обращения: 17.05.2025).

## **ПРИЛОЖЕНИЯ**

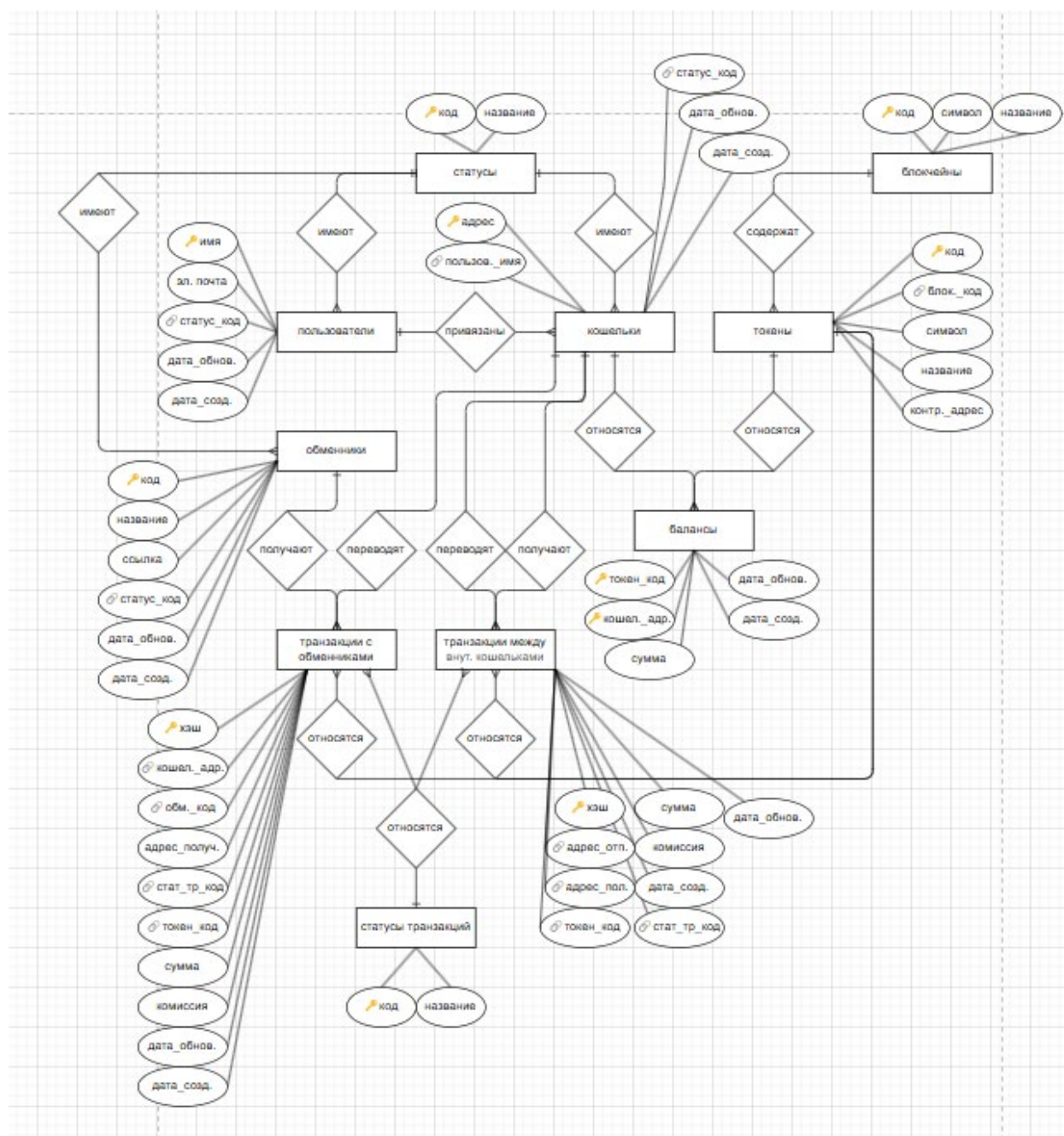


# **ПРИЛОЖЕНИЕ А** **ER-модель объекта моделирования**



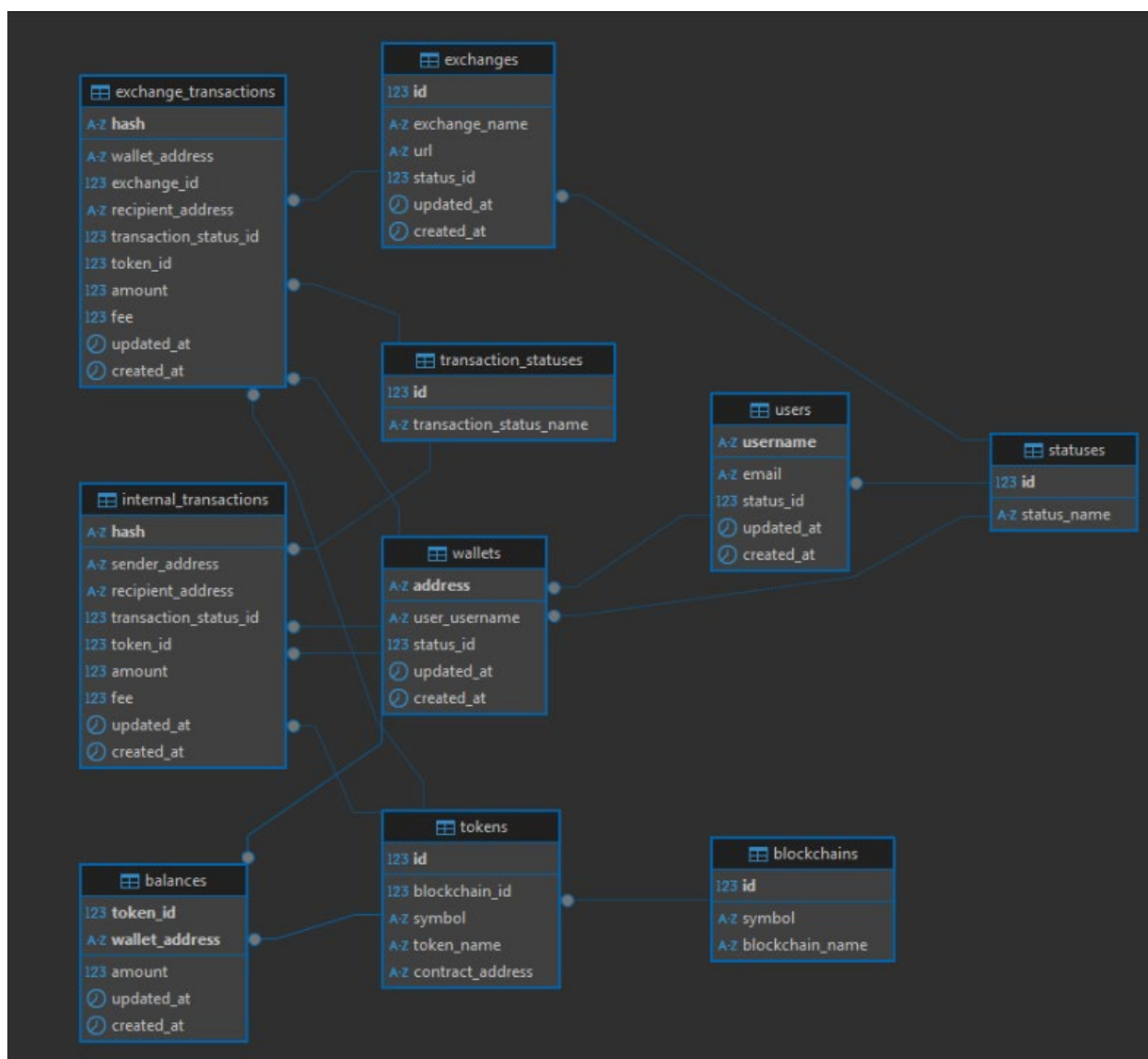
## ПРИЛОЖЕНИЕ Б

### Подробная ER-модель объекта моделирования



## ПРИЛОЖЕНИЕ В

### IDEF1X модель объекта моделирования



## ПРИЛОЖЕНИЕ Г

### Реляционная модель объекта моделирования

