Колледж Автономной некоммерческой образовательной организации

высшего образования

«Научно-технологический университет «Сириус»

**Пояснительная записка по результату выполнения работы в рамках курса**

**«Технология разработки баз данных»**

**Тема: «Биржа»**

|  |  |
| --- | --- |
| Работу подготовил: | |
| Студент группы | К0709-22 |
|  | Зайцев Алексей Вадимович |
| Проверил: | |
| преподаватель | Картавых Е.В. |

Сириус 2025

Оглавление

[1 Введение 3](#__RefHeading___Toc1548_1444848943)

[2 Описание предметной области 4](#__RefHeading___Toc1550_1444848943)

[3 Проектирование БД 6](#__RefHeading___Toc1552_1444848943)

[3.1 Первичный список отношений 6](#__RefHeading___Toc1554_1444848943)

[3.2 Анализ отношения «Студент» 8](#__RefHeading___Toc1556_1444848943)

[3.2.2 Вывод по анализу нормальных форм 10](#__RefHeading___Toc2123_1444848943)

[4 Таблицы базы данных 11](#__RefHeading___Toc1564_1444848943)

[5 Связи 15](#__RefHeading___Toc1566_1444848943)

[6 Пример данных 16](#__RefHeading___Toc1568_1444848943)

[7 Заключение 17](#__RefHeading___Toc1570_1444848943)

# Введение

В данном проекте разработана база данных для платформы по управлению цифровыми активами и торговлей ими. Система позволяет регистрировать пользователей, осуществлять их верификацию, вести учёт активов, фиксировать изменения их стоимости, управлять портфелями, а также обрабатывать заказы и сделки. Такая архитектура обеспечивает надёжное хранение и обработку информации, необходимой для функционирования торговой платформы, и способствует масштабированию при увеличении нагрузки.

# Описание предметной области

В предметной области проекта выделены следующие основные сущности:

* **Пользователь (users)** – лицо, зарегистрированное на платформе.
  + id (UUID): уникальный идентификатор пользователя.
  + username (VARCHAR(50)): имя пользователя, уникальное в системе.
  + hash\_password (TEXT): хэш пароля.
  + email (TEXT): адрес электронной почты (уникальный).
  + created\_at (TIMESTAMP): время создания записи.
* **Документы верификации пользователя (user\_verification\_docs)** – сведения, подтверждающие личность пользователя.
  + id (UUID): уникальный идентификатор документа.
  + user\_id (UUID): внешний ключ, связывающий с таблицей users (один к одному).
  + given\_name (TEXT): имя, указанное в документе.
  + surname (TEXT): фамилия, указанная в документе.
  + document\_type (ENUM document\_type): тип документа (passport, national\_id, drivers\_license, others).
  + document\_number (TEXT): номер документа.
  + is\_male (BOOLEAN): пол владельца документа.
  + issued\_by (TEXT): орган, выдавший документ.
  + date\_of\_issue (DATE): дата выдачи документа.
  + date\_of\_expiry (DATE): срок действия документа (с ограничением, что дата окончания не раньше даты выдачи).
  + issuing\_state (TEXT): страна или регион выдачи.
* **Актив (assets)** – цифровой актив или инструмент, торгуемый на платформе.
  + id (UUID): уникальный идентификатор актива.
  + name (VARCHAR(50)): наименование актива.
  + symbol (VARCHAR(10)): биржевой символ актива.
* **Цены активов (asset\_prices)** – история изменения стоимости активов.
  + id (UUID): уникальный идентификатор записи цены.
  + asset\_id (UUID): внешний ключ, ссылающийся на таблицу assets.
  + price (NUMERIC(10, 2)): стоимость актива.
  + created\_at (TIMESTAMP): время фиксации цены.
  + Дополнительно созданы индексы для оптимизации запросов по asset\_id и времени фиксации.
* **Портфели (portfolios)** – учет активов, принадлежащих пользователям.
  + id (UUID): уникальный идентификатор записи портфеля.
  + user\_id (UUID): внешний ключ, связывающий с пользователем.
  + asset\_id (UUID): внешний ключ, связывающий с активом.
  + amount (NUMERIC(38, 18)): общее количество актива в портфеле.
  + available\_amount (NUMERIC(38, 18)): доступное для торговли количество (с проверками на неотрицательное значение).
  + Уникальное ограничение на пару (user\_id, asset\_id) обеспечивает, что для каждого пользователя и актива существует единственная запись.
* **Транзакции портфеля (portfolio\_transacrions)** – история операций по портфелю.
  + id (UUID): уникальный идентификатор транзакции.
  + portfolio\_id (UUID): внешний ключ, связывающий с таблицей portfolios.
  + transaction\_type (ENUM transaction\_type): тип транзакции (buy, sell, fee).
  + balance\_before (NUMERIC(38, 18)): баланс до проведения операции.
  + balance\_after (NUMERIC(38, 18)): баланс после проведения операции.
  + amount (NUMERIC(38, 18)): сумма изменения баланса (с проверками на неотрицательное значение).
  + created\_at (TIMESTAMP): время проведения транзакции.
* **Заказы (orders)** – заявки пользователей на покупку или продажу активов.
  + id (UUID): уникальный идентификатор заказа.
  + user\_id (UUID): внешний ключ, связывающий с пользователем.
  + asset\_id (UUID): внешний ключ, связывающий с активом.
  + order\_type (ENUM order\_type): тип заказа (buy, sell).
  + order\_status (ENUM order\_status): статус заказа (pending, completed, cancelled).
  + amount (NUMERIC(38, 18)): суммарное количество актива для заказа (с проверкой на положительность).
  + filled\_amount (NUMERIC(38, 18)): количество, уже исполненное по заказу (не может превышать общее количество).
  + price (NUMERIC(10, 2)): цена, по которой размещён заказ.
* **Сделки (trades)** – записи о проведённых сделках, возникающих при согласовании заказов на покупку и продажу.
  + id (UUID): уникальный идентификатор сделки.
  + buy\_order\_id (UUID): внешний ключ, ссылающийся на заказ на покупку.
  + sell\_order\_id (UUID): внешний ключ, ссылающийся на заказ на продажу.
  + trade\_amount (NUMERIC(38, 18)): объём проведённой сделки (с проверкой на положительное значение).
  + traded\_at (TIMESTAMP): время совершения сделки.

# Проектирование БД

## Первичный список отношений

Согласно списку сущностей, представленных в разделе «Описание предметной области» получаем первичный список отношений базы данных:

1. **users**
   1. id (UUID)
   2. username (VARCHAR(50))
   3. hash\_password (TEXT)
   4. email (TEXT)
   5. created\_at (TIMESTAMP)
2. **user\_verification\_docs**
   1. id (UUID)
   2. user\_id (UUID)
   3. given\_name (TEXT)
   4. surname (TEXT)
   5. document\_type (ENUM document\_type) (passport, national\_id, drivers\_license, others).
   6. document\_number (TEXT)
   7. is\_male (BOOLEAN)
   8. issued\_by (TEXT)
   9. date\_of\_issue (DATE)
   10. date\_of\_expiry (DATE)
   11. issuing\_state (TEXT)
3. **assets**
   1. id (UUID)
   2. name (VARCHAR(50))
   3. symbol (VARCHAR(10))
4. **asset\_prices**
   1. id (UUID)
   2. asset\_id (UUID)
   3. price (NUMERIC(10, 2))
   4. created\_at (TIMESTAMP)
5. **portfolios**
   1. id (UUID)
   2. user\_id (UUID)
   3. asset\_id (UUID)
   4. amount (NUMERIC(38, 18))
   5. available\_amount (NUMERIC(38, 18))
6. **portfolio\_transacrions**
   1. id (UUID)
   2. portfolio\_id (UUID)
   3. transaction\_type (ENUM transaction\_type) (buy, sell, fee).
   4. balance\_before (NUMERIC(38, 18))
   5. balance\_after (NUMERIC(38, 18))
   6. amount (NUMERIC(38, 18)
   7. created\_at (TIMESTAMP)
7. **orders**
   1. id (UUID)
   2. user\_id (UUID)
   3. asset\_id (UUID)
   4. order\_type (ENUM order\_type) (buy, sell).
   5. order\_status (ENUM order\_status) (pending, completed, cancelled).
   6. amount (NUMERIC(38, 18))
   7. filled\_amount (NUMERIC(38, 18))
   8. price (NUMERIC(10, 2))
8. **trades**
   1. id (UUID)
   2. buy\_order\_id (UUID)
   3. sell\_order\_id (UUID)
   4. trade\_amount (NUMERIC(38, 18))
   5. traded\_at (TIMESTAMP)

Далее выполним анализ отношений на предмет соответствия 1-3 нормальным формам.

## Анализ отношения «Студент»

#### *3.2.1 Анализ отношения «Пользователь» (users)*

**Проверка на соответствие 1НФ**

* Каждая запись имеет уникальный идентификатор – атрибут **id** (UUID), используемый в качестве первичного ключа.
* Все атрибуты (username, hash\_password, email, created\_at) содержат атомарные значения.

**Проверка на соответствие 2НФ**

* Поскольку первичный ключ – это простой атрибут (**id**), все остальные атрибуты зависят от него напрямую, и отношение удовлетворяет требованиям 2НФ.

**Проверка на соответствие 3НФ**

* Отсутствуют транзитивные зависимости: ни один неключевой атрибут не определяется через другой неключевой атрибут.
* Таким образом, отношение «Пользователь» находится в 3НФ.

#### 3.2.2 Анализ отношения «Документы верификации» (user\_verification\_docs)

**Проверка на соответствие 1НФ**

* Таблица имеет первичный ключ **id** (UUID).
* Атрибуты: user\_id, given\_name, surname, document\_type, document\_number, is\_male, issued\_by, date\_of\_issue, date\_of\_expiry, issuing\_state – все значения атомарны.

**Проверка на соответствие 2НФ**

* Поскольку первичный ключ – простой (**id**), все атрибуты зависят от него напрямую, что удовлетворяет требованиям 2НФ.

**Проверка на соответствие 3НФ**

* Нет транзитивных зависимостей между неключевыми атрибутами.
* Связь с таблицей **users** реализована через внешний ключ (**user\_id**), что не нарушает нормализацию.
* Таким образом, отношение находится в 3НФ.

#### 3.2.3 Анализ отношения «Активы» (assets)

**Проверка на соответствие 1НФ**

* Таблица имеет атрибут **id** (UUID) в качестве первичного ключа.
* Атрибуты **name** и **symbol** содержат атомарные значения, каждый из которых уникален.

**Проверка на соответствие 2НФ**

* Простой первичный ключ гарантирует, что все атрибуты зависят напрямую от **id**.

**Проверка на соответствие 3НФ**

* Отсутствуют транзитивные зависимости: ни один из атрибутов не определяется через другой неключевой атрибут.
* Отношение удовлетворяет 3НФ.

#### 3.2.4 Анализ отношения «Цены активов» (asset\_prices)

**Проверка на соответствие 1НФ**

* Первичный ключ – атрибут **id** (UUID), что обеспечивает уникальную идентификацию записей.
* Атрибуты **asset\_id**, **price** и **created\_at** являются атомарными.

**Проверка на соответствие 2НФ**

* Поскольку ключ – простой, все остальные атрибуты зависят непосредственно от **id**.

**Проверка на соответствие 3НФ**

* Нет транзитивных зависимостей: например, **price** и **created\_at** не зависят друг от друга, а лишь от записи с уникальным **id**.
* Отношение находится в 3НФ.

#### 3.2.5 Анализ отношения «Портфели» (portfolios)

**Проверка на соответствие 1НФ**

* Таблица имеет первичный ключ **id** (UUID).
* Атрибуты **user\_id**, **asset\_id**, **amount** и **available\_amount** – все атомарны.
* Уникальное ограничение на пару (**user\_id, asset\_id**) обеспечивает уникальность сочетания пользователя и актива.

**Проверка на соответствие 2НФ**

* Поскольку первичный ключ – простой (**id**), все неключевые атрибуты зависят от него напрямую.

**Проверка на соответствие 3НФ**

* Отсутствуют транзитивные зависимости: значения **amount** и **available\_amount** не вычисляются через другие неключевые атрибуты, а напрямую зависят от записи с уникальным **id**.
* Таблица удовлетворяет требованиям 3НФ.

#### 3.2.6 Анализ отношения «Транзакции портфелей» (portfolio\_transacrions)

**Проверка на соответствие 1НФ**

* Таблица имеет первичный ключ **id** (UUID).
* Все атрибуты (**portfolio\_id**, **transaction\_type**, **balance\_before**, **balance\_after**, **amount**, **created\_at**) представлены атомарно.

**Проверка на соответствие 2НФ**

* Простой первичный ключ обеспечивает, что все остальные атрибуты зависят напрямую от **id**.

**Проверка на соответствие 3НФ**

* Нет транзитивных зависимостей: ни один атрибут не определяется через другой неключевой атрибут.
* Таким образом, отношение находится в 3НФ.

#### 3.2.7 Анализ отношения «Заказы» (orders)

**Проверка на соответствие 1НФ**

* Таблица имеет уникальный идентификатор **id** (UUID) в качестве первичного ключа.
* Все атрибуты (**user\_id**, **asset\_id**, **order\_type**, **order\_status**, **amount**, **filled\_amount**, **price**) являются атомарными.
* Наличие проверок (например, filled\_amount ≤ amount) не нарушает атомарность данных.

**Проверка на соответствие 2НФ**

* Поскольку ключ – простой (**id**), все атрибуты зависят напрямую от него.

**Проверка на соответствие 3НФ**

* Отсутствуют транзитивные зависимости: все атрибуты зависят непосредственно от **id**, а вычисляемые ограничения (например, проверка количества заполненного заказа) не создают зависимости между неключевыми полями.
* Таблица удовлетворяет требованиям 3НФ.

#### 3.2.8 Анализ отношения «Сделки» (trades)

**Проверка на соответствие 1НФ**

* Таблица имеет атрибут **id** (UUID) в качестве первичного ключа.
* Атрибуты **buy\_order\_id**, **sell\_order\_id**, **trade\_amount** и **traded\_at** содержат атомарные значения.

**Проверка на соответствие 2НФ**

* Простой ключ (**id**) гарантирует, что все остальные атрибуты зависят от него напрямую.

**Проверка на соответствие 3НФ**

* Отсутствуют транзитивные зависимости: данные о сделке (например, trade\_amount) зависят напрямую от уникальной записи с **id** и не вычисляются через другие атрибуты.
* Таким образом, отношение находится в 3НФ.

### Вывод по анализу нормальных форм

Все рассмотренные таблицы – **users**, **user\_verification\_docs**, **assets**, **asset\_prices**, **portfolios**, **portfolio\_transacrions**, **orders** и **trades** – удовлетворяют требованиям 1НФ, 2НФ и 3НФ. Это гарантирует, что:

* Каждая запись имеет уникальный идентификатор.
* Все атрибуты имеют атомарные значения.
* Нет избыточных или транзитивных зависимостей, что способствует минимизации дублирования данных и упрощению поддержки целостности базы данных.

# Таблицы базы данных

Скрипт для создания таблиц в базе данных представлен ниже:

-- Удаление таблиц и типов (при наличии)

drop table if EXISTS users, user\_verification\_docs, assets, asset\_prices, portfolios, portfolio\_transacrions, orders, trades CASCADE;

drop type if EXISTS document\_type, transaction\_type, order\_type, order\_status;

-- Таблица пользователей

CREATE TABLE users (

    id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

    username VARCHAR(50) NOT NULL UNIQUE,

    hash\_password TEXT NOT NULL,

    email TEXT NOT NULL UNIQUE,

    created\_at TIMESTAMP with time zone DEFAULT now()

);

-- Тип документов

CREATE TYPE document\_type AS ENUM (

    'passport',

    'national\_id',

    'drivers\_license',

    'others'

);

-- Документы верификации пользователей

CREATE TABLE user\_verification\_docs (

    id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

    user\_id UUID NOT NULL UNIQUE,

    given\_name TEXT NOT NULL,

    surname TEXT NOT NULL,

    document\_type document\_type NOT NULL,

    document\_number TEXT NOT NULL,

    is\_male BOOLEAN NOT NULL,

    issued\_by TEXT NOT NULL,

    date\_of\_issue DATE NOT NULL,

    date\_of\_expiry DATE NOT NULL,

    issuing\_state TEXT NOT NULL,

    CHECK (date\_of\_expiry >= date\_of\_issue)

);

ALTER TABLE user\_verification\_docs ADD FOREIGN KEY (user\_id) REFERENCES users (id) ON DELETE CASCADE;

-- Таблица активов

CREATE TABLE assets (

    id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

    name VARCHAR(50) NOT NULL UNIQUE,

    symbol VARCHAR(10) NOT NULL UNIQUE

);

-- Таблица цен активов

CREATE TABLE asset\_prices (

    id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

    asset\_id UUID NOT NULL,

    price NUMERIC(10, 2) NOT NULL,

    created\_at TIMESTAMP with time zone DEFAULT now()

);

CREATE INDEX idx\_asset\_prices\_asset\_id ON asset\_prices (asset\_id);

CREATE INDEX idx\_asset\_prices\_created\_at ON asset\_prices (created\_at);

ALTER TABLE asset\_prices ADD FOREIGN KEY (asset\_id) REFERENCES assets (id) ON DELETE CASCADE;

-- Таблица портфелей

CREATE TABLE portfolios (

    id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

    user\_id UUID NOT NULL,

    asset\_id UUID NOT NULL,

    amount NUMERIC(38, 18) NOT NULL,

    available\_amount NUMERIC(38, 18) NOT NULL,

    CHECK (amount >= 0),

    CHECK (available\_amount >= 0),

    UNIQUE (user\_id, asset\_id)

);

CREATE INDEX idx\_portfolios\_user\_id ON portfolios (user\_id);

CREATE INDEX idx\_portfolios\_asset\_id ON portfolios (asset\_id);

ALTER TABLE portfolios ADD FOREIGN KEY (user\_id) REFERENCES users (id) ON DELETE RESTRICT;

ALTER TABLE portfolios ADD FOREIGN KEY (asset\_id) REFERENCES assets (id) ON DELETE RESTRICT;

-- Таблица транзакций портфелей

CREATE TYPE transaction\_type AS ENUM (

    'buy',

    'sell',

    'fee'

);

CREATE TABLE portfolio\_transacrions (

    id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

    portfolio\_id UUID NOT NULL,

    transaction\_type transaction\_type NOT NULL,

    balance\_before NUMERIC(38, 18) NOT NULL,

    balance\_after NUMERIC(38, 18) NOT NULL,

    amount NUMERIC(38, 18) NOT NULL,

    created\_at TIMESTAMP with time zone DEFAULT now(),

    CHECK (amount >= 0),

    CHECK (balance\_before >= 0),

    CHECK (balance\_after >= 0)

);

CREATE INDEX idx\_portfolio\_transactions\_portfolio\_id ON portfolio\_transacrions (portfolio\_id);

CREATE INDEX idx\_portfolio\_transactions\_transaction\_type ON portfolio\_transacrions (transaction\_type);

CREATE INDEX idx\_portfolio\_transaction\_created\_at ON portfolio\_transacrions (created\_at);

ALTER TABLE portfolio\_transacrions ADD FOREIGN KEY (portfolio\_id) REFERENCES portfolios (id) ON DELETE CASCADE;

-- Таблица заказов

CREATE TYPE order\_type AS ENUM (

    'buy',

    'sell'

);

CREATE TYPE order\_status AS ENUM (

    'pending',

    'completed',

    'cancelled'

);

CREATE TABLE orders (

    id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

    user\_id UUID NOT NULL,

    asset\_id UUID NOT NULL,

    order\_type order\_type NOT NULL,

    order\_status order\_status NOT NULL,

    amount NUMERIC(38, 18) NOT NULL,

    filled\_amount NUMERIC(38, 18) NOT NULL,

    price NUMERIC(10, 2) NOT NULL,

    CHECK (amount > 0),

    CHECK (filled\_amount >= 0),

    CHECK (filled\_amount <= amount),

    CHECK (price >= 0)

);

create index idx\_orders\_user\_id on orders (user\_id);

create index idx\_orders\_asset\_id on orders (asset\_id);

create index idx\_orders\_price on orders (price);

create index idx\_orders\_order\_type on orders (order\_type);

create index idx\_orders\_order\_status on orders (order\_status);

create index idx\_orders\_order\_type\_asset\_id\_price on orders (order\_type, asset\_id, price);

CREATE index idx\_orders\_order\_status\_asset\_id on orders (order\_status, asset\_id);

ALTER TABLE orders ADD FOREIGN KEY (user\_id) REFERENCES users (id) ON DELETE RESTRICT;

ALTER TABLE orders ADD FOREIGN KEY (asset\_id) REFERENCES assets (id) ON DELETE RESTRICT;

-- Таблица сделок

CREATE TABLE trades (

    id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

    buy\_order\_id UUID NOT NULL,

    sell\_order\_id UUID NOT NULL,

    trade\_amount NUMERIC(38, 18) NOT NULL,

    traded\_at TIMESTAMP with time zone DEFAULT now(),

    CHECK (trade\_amount > 0)

);

CREATE INDEX idx\_trades\_buy\_order\_id ON trades (buy\_order\_id);

CREATE INDEX idx\_trades\_sell\_order\_id ON trades (sell\_order\_id);

CREATE INDEX idx\_trades\_created\_at ON trades (traded\_at);

ALTER TABLE trades ADD FOREIGN KEY (buy\_order\_id) REFERENCES orders (id) ON DELETE RESTRICT;

ALTER TABLE trades ADD FOREIGN KEY (sell\_order\_id) REFERENCES orders (id) ON DELETE RESTRICT;

# Связи

В базе данных имеются следующие связи между отношениями:

* **Пользователь и Документы верификации:**  
  Каждый пользователь может иметь один документ верификации. Эта связь реализована посредством внешнего ключа в таблице user\_verification\_docs, где user\_id ссылается на id таблицы users.
* **Активы и Цены активов:**  
  Один актив может иметь множество записей об изменении цены, что реализовано через внешний ключ asset\_id в таблице asset\_prices.
* **Пользователь и Портфели:**  
  Таблица portfolios связывает пользователей и активы (связь многие ко многим) с уникальным ограничением на пару (user\_id, asset\_id).
* **Портфели и Транзакции портфелей:**  
  Каждая запись транзакции относится к конкретному портфелю, что обеспечивает внешний ключ portfolio\_id.
* **Пользователь, Активы и Заказы:**  
  Заказы размещаются пользователями для определённых активов. Таблица orders содержит внешние ключи user\_id и asset\_id.
* **Заказы и Сделки:**  
  При согласовании заказов на покупку и продажу формируются сделки. Таблица trades связывает два заказа посредством полей buy\_order\_id и sell\_order\_id.

# Пример данных

1. Пример данных для таблицы users:

INSERT INTO users (username, hash\_password, email)

VALUES

  ('user1', 'hash\_password1', 'user1@example.com'),

  ('user2', 'hash\_password2', 'user2@example.com');

1. Пример данных для таблицы users\_verification\_docs:

INSERT INTO user\_verification\_docs (user\_id, given\_name, surname, document\_type, document\_number, is\_male, issued\_by, date\_of\_issue, date\_of\_expiry, issuing\_state)

VALUES

  (

    (SELECT id FROM users WHERE username = 'user1'),

    'Иван',

    'Иванов',

    'passport',

    '123456789',

    true,

    'ОВД',

    '2020-01-01',

    '2030-01-01',

    'RU'

  ),

  (

    (SELECT id FROM users WHERE username = 'user2'),

    'Мария',

    'Петрова',

    'national\_id',

    '987654321',

    false,

    'МВД',

    '2019-05-20',

    '2029-05-20',

    'RU'

  );

1. Пример данных для таблицы assets:

INSERT INTO assets (name, symbol)

VALUES

  ('Bitcoin', 'BTC'),

  ('Ethereum', 'ETH');

1. Пример данных для таблицы asset\_prices:

INSERT INTO asset\_prices (asset\_id, price)

VALUES

  (

    (SELECT id FROM assets WHERE symbol = 'BTC'),

    45000.00

  ),

  (

    (SELECT id FROM assets WHERE symbol = 'BTC'),

    45500.00

  ),

  (

    (SELECT id FROM assets WHERE symbol = 'ETH'),

    3000.00

  ),

  (

    (SELECT id FROM assets WHERE symbol = 'ETH'),

    3050.00

  );

1. Пример данных для таблицы portfolios:

INSERT INTO portfolios (user\_id, asset\_id, amount, available\_amount)

VALUES

  (

    (SELECT id FROM users WHERE username = 'user1'),

    (SELECT id FROM assets WHERE symbol = 'BTC'),

    1.5,

    1.5

  ),

  (

    (SELECT id FROM users WHERE username = 'user2'),

    (SELECT id FROM assets WHERE symbol = 'ETH'),

    10,

    8

  );

1. Пример данных для таблицы portfolio\_transactions:

INSERT INTO portfolio\_transacrions (portfolio\_id, transaction\_type, balance\_before, balance\_after, amount)

VALUES

  (

    (SELECT id FROM portfolios WHERE user\_id = (SELECT id FROM users WHERE username = 'user1')

      AND asset\_id = (SELECT id FROM assets WHERE symbol = 'BTC')),

    'buy',

    0,

    1.5,

    1.5

  ),

  (

    (SELECT id FROM portfolios WHERE user\_id = (SELECT id FROM users WHERE username = 'user2')

      AND asset\_id = (SELECT id FROM assets WHERE symbol = 'ETH')),

    'buy',

    0,

    10,

    10

  );

1. Пример данных для таблицы orders:

INSERT INTO orders (user\_id, asset\_id, order\_type, order\_status, amount, filled\_amount, price)

VALUES

  (

    (SELECT id FROM users WHERE username = 'user1'),

    (SELECT id FROM assets WHERE symbol = 'BTC'),

    'buy',

    'pending',

    1.0,

    0,

    45000.00

  ),

  (

    (SELECT id FROM users WHERE username = 'user2'),

    (SELECT id FROM assets WHERE symbol = 'ETH'),

    'sell',

    'pending',

    5.0,

    0,

    3050.00

  );

1. Пример данных для таблицы trades:

INSERT INTO trades (buy\_order\_id, sell\_order\_id, trade\_amount)

VALUES

  (

    (SELECT id FROM orders WHERE user\_id = (SELECT id FROM users WHERE username = 'user1') AND order\_type = 'buy'),

    (SELECT id FROM orders WHERE user\_id = (SELECT id FROM users WHERE username = 'user2') AND order\_type = 'sell'),

    0.5

  );

# Заключение

В результате проведённого проектирования создана структурированная база данных для платформы управления цифровыми активами. Разработанная модель охватывает ключевые аспекты работы платформы: регистрацию и верификацию пользователей, учёт активов, мониторинг их цен, управление портфелями, а также обработку заказов и сделок. Все отношения приведены к третьей нормальной форме, что обеспечивает отсутствие избыточности и поддержание целостности данных. Реализованные связи посредством внешних ключей способствуют сохранению согласованности данных при выполнении операций добавления, обновления и удаления. В ходе работы были получены практические знания в области проектирования БД для финансовых и торговых систем, а также в обеспечении производительности за счёт использования индексов и оптимальных ограничений