Практика 3: Биржа криптовалюты

Языки программирования: C, C++, Rust, Go, Python, Java, C#

Сложность: 3/10

Срок: 3 недели

Задание

Требуется реализовать рабочий прототип биржи на котором пользователи смогут торговать криптовалютой или другими активами.

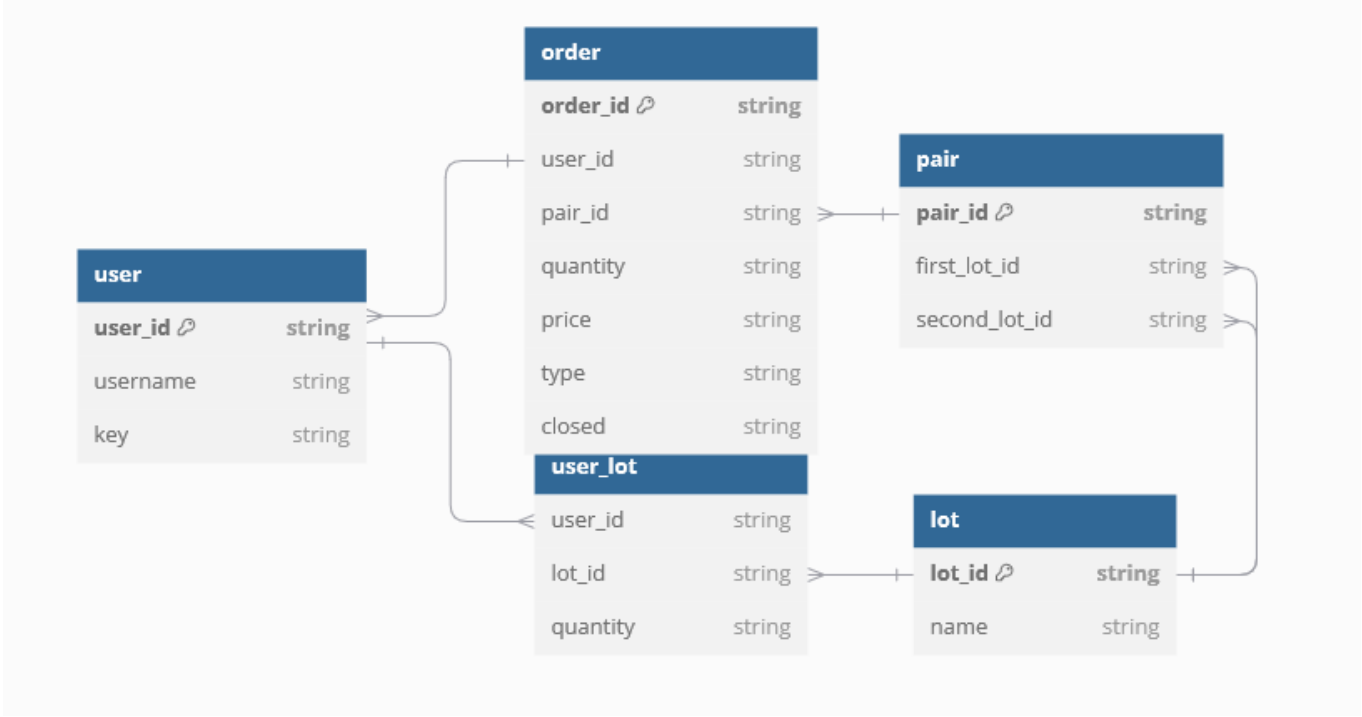
Для реализации биржи нужно использовать СУБД из первой практики, подключение к базе данных требуется осуществлять по сети, для этого выполнялась вторая практика.

В ходе реализации нужно отталкиваться от следующих основных сущностей:

1. Order - заявка на продажу или покупку какого либо актива от пользователя биржи
2. Lot - лот (актив) который торгуется на бирже
3. User - пользователь биржи

Описание БД

Минимальная структура базы данных будет выглядит следующим образом:



Ниже описаны поля таблиц базы данных

```
Table user {  // Данные о пользователях биржи
  user_id string [primary key]  // Числовой идентификатор пользователя
  username string  // Имя пользователя
  key string  // Ключ пользователя
}

table user_lot {  // Данные о активах пользователей и их количичистве
  user_id string  // Числовой идентификатор пользователя
  lot_id string  // Числовой идентификатор актива
  quantity string  // Количество
}

Table order {  // Данные о заявках на покупку или продажу активов
  order_id string [primary key]  // Числовой идентификатор заявки
  user_id string  // Числовой идентификатор пользователя который создал заявку
  pair_id string  // Числовой идентификатор актива
  quantity string  // Количество лотов в заявке
  price string  // Цена за 1 лот
  type string  // Тип заявки - покупка или продажа
  closed string  // Время закрытия заявки, если не закрыта, то пустая строка
}
```

```
Table lot {  // Данные об активах, которые торгуются на бирже
  lot_id string [primary key]  // Числовой идентификатор лота
  name string  // Название актива
}

Table pair {  // Данные о парах лотов (валют/авктивов)
  pair_id string [primary key]  // Числовой идентификатор пары
  first_lot_id string  // Продаваемый лот
  second_lot_id string  // Покупаемый лот
}

Ref: "user"."user_id" < "user_lot"."user_id"

Ref: "lot"."lot_id" < "user_lot"."lot_id"

Ref: "order"."user_id" < "user"."user_id"

Ref: "lot"."lot_id" < "pair"."first_lot_id"

Ref: "lot"."lot_id" < "pair"."second_lot_id"

Ref: "pair"."pair_id" < "order"."pair_id"
```

Описание API

Требуется реализовать API работающий по протоколу HTTP содержащий следующие ресурсы:

Создание пользователей:

```
POST /user
{
        "username": string
}

Response
{
```

```
        "key": "a8f5f167f44f4964e6c998dee827110c"
}
```

Создание ордера:

```
POST /order

X-USER-KEY: string

{
        "pair_id": int,
        "quantity": float,
        "price": float,
        "type": "sell" | "buy"
}

Response
{
        "order_id": int
}
```

Получение списка ордеров:

```
GET /order

Response
[
        {
                "order_id": int,
                "user_id": int,
                "lot_id": int,
                "quantity": float,
                "type": "sell" | "buy",
                "price": float,
                "closed": string
        },
        ...
]
```

Удаление ордера:

```
DELETE /order

X-USER-KEY: string

{
        "order_id": int
}
```

Получение информации о лотах:

```
GET /lot

Response
[
        {
                "lot_id": int,
                "name": string
        },
        ...
]
```

Получение информации о парах:

```
GET /pair

Response
[
        {
                "pair_id": int,
                "sale_lot_id": int,
                "buy_lot_id": int
        }
]
```

Получение информации об активах пользователя:

```
GET /balance

X-USER-KEY: string
```

```
Response
[
        {
                "lot_id": int,
                "quantity": float
        },
        ...
]
```

Конфигурация

Также биржа должна содержать файл (`config.json`) конфигурации в формате JSON. Файл конфигурации должен содержать следующие поля:

```
{
        "lots": [  // Список активов, ваши активы могут отличаться от примера
                "RUB",
                "BTC",
                "ETH",
                "USDT",
                "USDC"
        ],
        "database_ip": "localhost",  // Адрес СУБД
        "database_port": 7432  // Порт СУБД
}
```

При запуске биржи лоты из конфигурации должны положиться в базу данных в таблицу `lot`, а также формируются валютные пары (по методу каждый с каждым) и кладутся в таблицу `pair`.

При создании пользователя его баланс автоматически пополняется на 1000 единиц каждого доступного лота (актива).

Для выполнения операций при которых нужно идентифицировать пользователя требуется отправлять HTTP заголовок `X-USER-KEY` в значения которого указывается ключ пользователя от лица которого выполняется запрос (примитивная система аутентификации). К таким операциям относятся: создание/удаление ордеров и получение информации о балансе пользователя. Ключ пользователя должна вернуть биржа при создании пользователя, ключ должен быть сгенерирован случайным образом.

Бизнес-логика на примере

У нас есть 3 лота: RUB, USD и BTC.

Данные лоты формируют 3 валютные пары: RUB/USD, RUB/BTC, USD/BTC.

Если мы хотим купить RUB за USD, мы выставляем ордер на покупку по валютной паре RUB/USD, запрос для этого будет выглядеть следующим образом:

```
POST /order

X-USER-KEY: string

{
        "pair_id": 1,  // Идентификатор пары RUB/USD
        "quantity": 300,  // Количество RUB которое хотим купить
        "price": 0.015,  // Цена 1 RUB в USD
        "type": "buy"  // Тип ордера – покупка
}
```

При отправке запроса в базе данных должна сформироваться следующая запись:

order_id	user_id	pair_id	quantity	price	type	closed
1	1	1	300	0.015	buy	

При выставлении данной заявки с баланса пользователя должно сняться 4.5 USD (из расчета что пользователь покупает 300 RUB по цене 0.015 = 300 * 0.015 = 4.5). Если пользователь отменил заявку, средства в полном объеме должны вернуться на его счет.

Теперь другой пользователь выставляет ордер по той же валютной паре RUB/USD на продажу 200 RUB по цене 0.01. Данная цена удовлетворяет первого пользователя для покупки RUB.

Теперь в базе данных буду следующие записи:

order_id	user_id	pair_id	quantity	price	type	closed
1	1	1	200	0.01	buy	1730738627
2	2	1	200	0.01	sell	1730738627
3	1	1	100	0.015	buy	

Мы видим, что первая заявка первого пользователя на покупку была разбита на 2 заявки: 1 - кол-во 200 по цене 0.01, 2 - кол-во 100 по цене 0.015. Мы частично удовлетворили первый ордер первого пользователя и полностью удовлетворили ордер второго пользователя.

Первый пользователь получил на баланс +200 RUB и -2 USD, второй пользователь получил -200 RUB и +2 USD.

Стоит помнить, что при добавлении нового ордера в БД, сначала нужно проверить существующие обратные ордера и удовлетворить их, после этого добавлять новый ордер.