دانشگاه صنعتی شریف

بهار ۱۴۰۳

گزارش پروژه فاز ۲

طراحی پایگاه‌های داده

اعضای گروه:

سینا ایمانی - ۹۹۱۰۰۳۱۴

علی شاه‌علی - ۴۰۰۱۰۹۹۰۵

علیرضا حبیب زاده - ۹۹۱۰۹۳۹۳

بهار ۱۴۰۳

Index:

1: مواقع زیادی نیاز به سورت کردن سفارش‌ها بر حسب قیمت‌شان داریم. پس یک index روی ستون unit_price از ArchiveOrders میسازیم:

```
CREATE INDEX idx_archiveorders_unit_price ON ArchiveOrders(unit_price);
```

2: یکی دیگر از نیازمندی‌ها، پیدا کردن آخرین سفارش بر اساس بازار مربوطه است. برای بهبود این پرسمان روی زوج market_id, id از جدول ArchiveOrders یک index میسازیم. در این حالت پیدا کردن آخرین سفارش یک بازار معادل سرچ کردن برای راست‌ترین market_id روی یک آرایه است که توسط ایندکس انجام می‌شود.

```
CREATE INDEX idx_archiveorders_market_id_id ON ArchiveOrders(market_id, id);
```

3: یک سرچ دیگر پیدا کردن بازارهای با یک ارز خاص است. برای تسریع این فرایند هم نیاز به ایندکس روی ستون working_currency از جدول Markets داریم.

```
CREATE INDEX idx_markets_working_currency ON Markets(working_currency);
```

4: مانند مثال قبل، کوئری‌های مربوط به کیف‌پول‌های یک ارز خاص نیز ممکن است. پس روی ستون currency از جدول Wallets هم یک ایندکس میسازیم.

```
CREATE INDEX idx_wallets_currency ON Wallets(currency);
```

جدا از موارد گفته شده، برای foreign key ها هم ایندکس نیاز داریم. ولی در اینجا همه foreign keyها به یک primary key بود و پیشفرض گرفتیم که همه primary keyها ایندکس دارند.

کوئری‌ها:

:1

برای این کوئری، ابتدا به ازای هر بازار آخرین ArchiveOrder آنرا با استفاده از یک group by روی ستون market_id و تابع max روی ستون id پیدا کرده و در یک جدول جدید ذخیره می‌کنیم. سپس جدول جدید را با ArchiveOrders جوین کرده تا قیمت آخرین تراکنش هر بازار به دست آید.

:2

برای این کوئری، جداول ActiveOrders و ArchivedOrders را با هم جوین می‌کنیم تا اطلاعات سفارشات فعال به دست آید، سپس نتیجه را با Wallets روی wallet_id جوین می‌کنیم تا currency مربوط به سفارشات هم از Wallets به دست آید. در ادامه نتیجه را بر اساس زوج currency و unit_price دسته‌بندی می‌کنیم و مجموع remainder_number را به ازای سفارشات نوع 'Buy' و 'Sell' یه طور مستقل با تابع sum محاسبه می‌کنیم.
برای محاسبه مجموع مقدار باقی‌مانده روی سفارشات یک نوع خاص، به شکل زیر عمل می‌کنیم:

```
SUM(CASE WHEN o.type = 'Buy' THEN ao.remained_number ELSE 0 END)
```

:3

برای محاسبه جواب این کوئری چند مرحله طی می‌کنیم. در گام اول باید قیمت هر ارز را به ریال پیدا کرده، و سپس به ازای هر کاربر موجودی کیف پول او را حساب کنیم.

برای پیدا کردن قیمت هر ارز، ابتدا آخرین ArchivedOrder ای که به ازای هر currency ثبت شده را باید پیدا کنیم. پس جدول‌های ArchiveOrders و Markets را با هم جوین کرده و فقط بازارهایی که ارز پایه (base_currency) آنها ریال است نگه میداریم. سپس نتیجه را بر حسب working_currency بازارها group by کرده و در نهایت با تابع MAX آخرین سفارش هر ارز را پیدا می‌کنیم:

```
WITH LatestOrders AS (
    SELECT
        m.working_currency,
        MAX(ao.id) AS latest_archive_order_id
    FROM ArchiveOrders ao
    INNER JOIN Markets m ON ao.market_id = m.id
    WHERE m.base_currency = 'IRR'
    GROUP BY m.working_currency
),
```

سپس جدول LatestOrders محاسبه شده را با جدول ArchivedOrders جوین می‌کنیم تا اطلاعات آخرین سفارش هر ارز به دست آید. از بین آنها قیمت تبادل آن سفارش را برداشته و با ارز مد نظر در جدول LastPrices ذخیره می‌کنیم:

```
LatestPrices AS (
    SELECT
        lo.working_currency,
        ao.unit_price AS current_price
    FROM
        LatestOrders lo
        JOIN ArchiveOrders ao ON lo.latest_archive_order_id = ao.id
),
```

در گام بعد میزان موجودی را به ازار هر کیف پول پیدا می‌کنیم. برای این کار به ازای هر کیف پول باید آخرین BalanceUpdate آنرا برداشته و newBalance آن معادل موجودی کیف پول است. آخرین آپدیت هر کیف پول هم در ستون last_balance_update_number از جدول Wallets موجود است، پس کافی است این دو جدول را با این ستون جوین کنیم:

```
LatestBalancePerWallet AS (
    SELECT
        w.id AS wallet_id,
        w.user_id,
        w.currency,
        bu.newBalance AS balance
    FROM
        Wallets w
        JOIN BalanceUpdates bu ON w.last_balance_update_number = bu.number
)
```

در نهایت جدول Users را با جدول LastBalancePerWallet و LastPrices جوین می‌کنیم و پس از groupby کردن بر اساس کاربران، با استفاده از تابع SUM جمع هر ارز از هر کیف پول کاربر را ضربدر ارزش آن ارز حساب می‌کنیم:

```
SELECT
    u.username,
    u.fullname,
    COALESCE(SUM(lb.balance * lp.current_price), 0) AS total_balance_rials
FROM
    LatestBalancePerWallet lb
    INNER JOIN Users u ON u.id = lb.user_id
    LEFT JOIN LatestPrices lp ON lb.currency = lp.working_currency
GROUP BY u.id;
```

:4

برای این کوئری ابتدا هر کدام از ۱۰ بالاترین خرید و ۱۰ پایین‌ترین فروش را جدا پیدا کرده، و در انتها union می‌کنیم.

برای این‌کار هم جدول Markets را با ArchiveOrders جوین کرده و با استفاده از تابع ROWNUMBER و PARTITION کردن بر اساس بازار و مرتب‌سازی بر اساس قیمت سفارش، رنک هر سفارش در بین سفارش‌های بازار خودش را پیدا می‌کنیم و در جدول‌های TopSells و TopBuys ذخیره می‌کنیم:

```
WITH TopSells AS (
    SELECT
        m.id AS market_id,
        ao.type AS type,
        ao.id AS order_id,
        ao.unit_price AS price,
        ROW_NUMBER() OVER (PARTITION BY m.id ORDER BY ao.unit_price DESC) AS rank
    FROM
        Markets m
    JOIN ArchiveOrders ao ON ao.market_id = m.id
    WHERE ao.type = 'Buy'
),
TopBuys AS (
    SELECT
        m.id AS market_id,
        ao.type AS type,
        ao.id AS order_id,
        ao.unit_price AS price,
        ROW_NUMBER() OVER (PARTITION BY m.id ORDER BY ao.unit_price ASC) AS rank
    FROM
        Markets m
    JOIN ArchiveOrders ao ON ao.market_id = m.id
    WHERE ao.type = 'Sell'
)
```

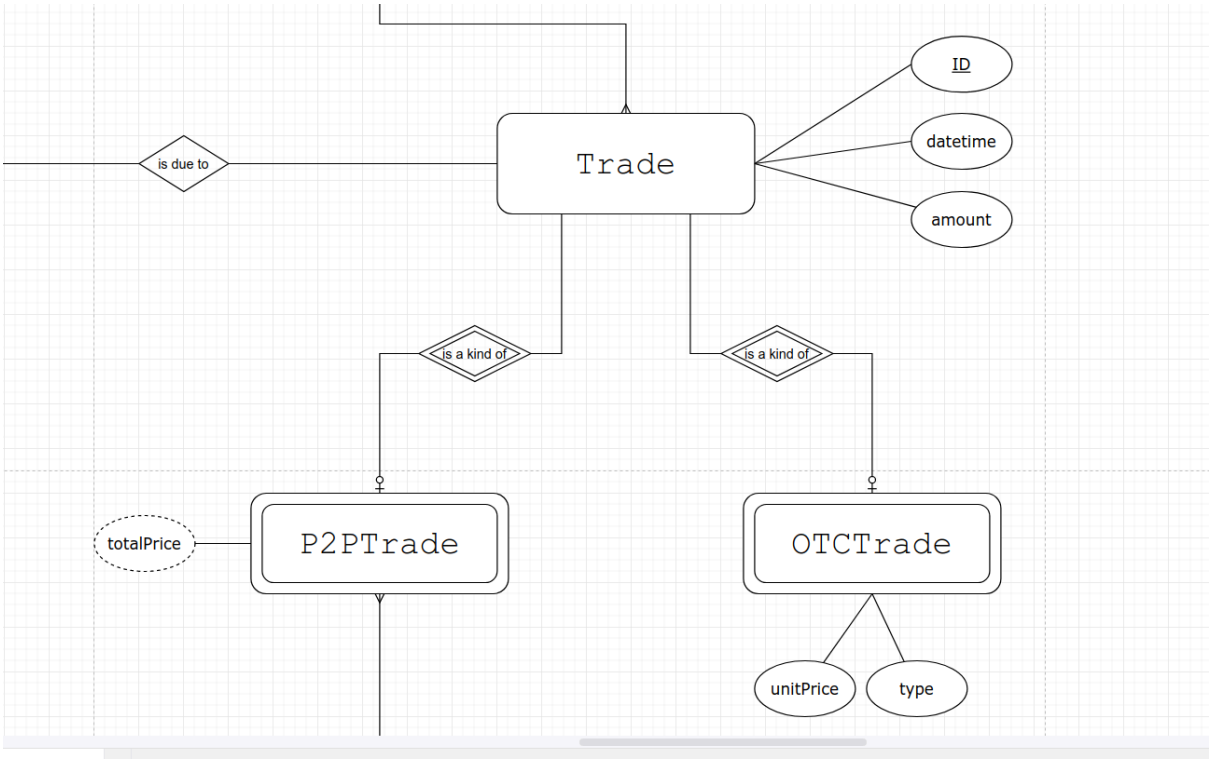
و در انتها هم از هر کدام این دو جدول آنهایی که rank حداکثر ۱۰ دارند را select کرده و نتایج را اجتماع میگیریم:

```sql
SELECT
    market_id,
    type,
    order_id,
    price
FROM TopSells
WHERE rank <= 10

UNION ALL

SELECT
    market_id,
    type,
    order_id,
    price
FROM TopBuys
WHERE rank <= 10
ORDER BY market_id, type, price DESC;
```

۳.۱ منطبق‌سازی طراحی ER با SQL

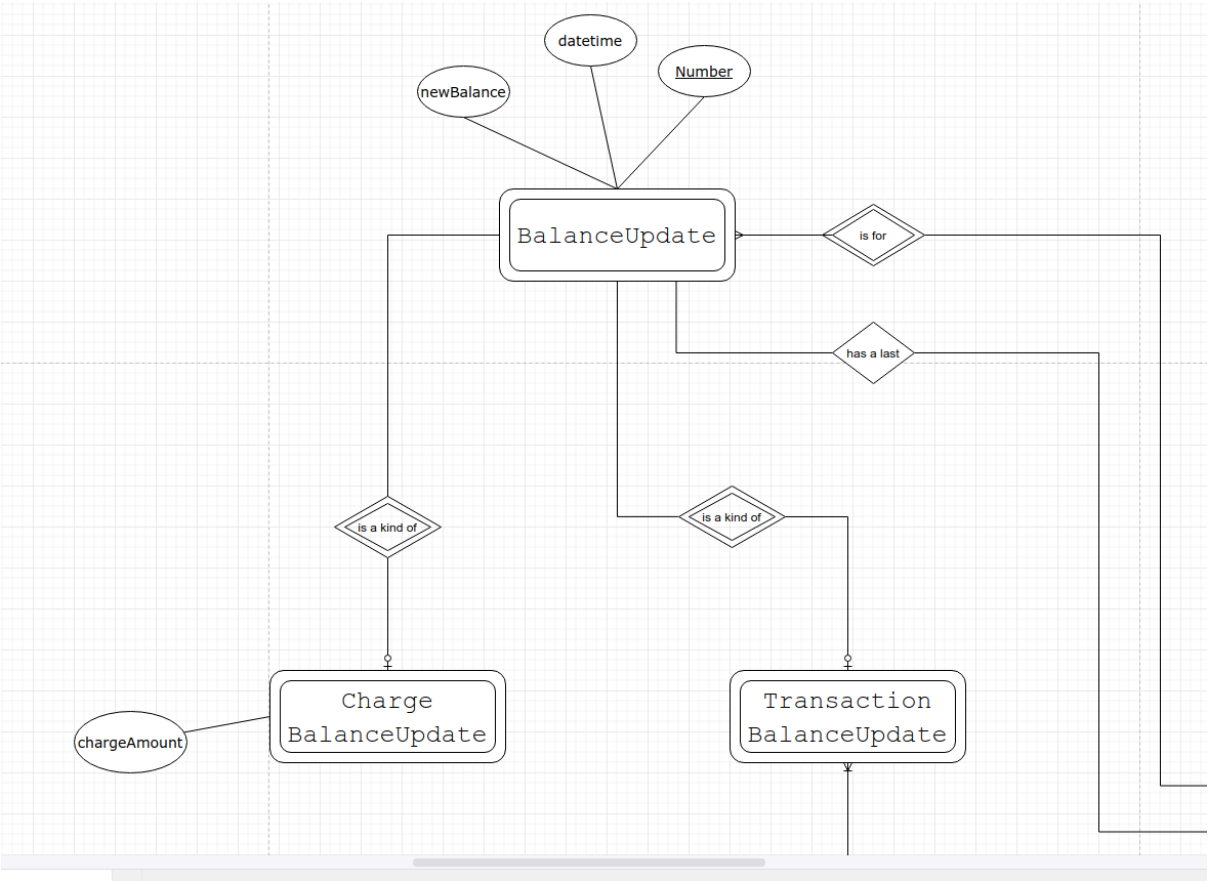
برای ایجاد انطباق بین طراحی انجام گرفته در فاز اول و SQL تغییراتی را در طراحی خود لحاظ کردیم:

۱. در طراحی اولیه یک موجودیت انتزاعی به نام Trade ساخته بودیم که دو موجودیت P2PTrade و OTCTrade از آن مشتق می‌شدند. برای آن که بتوانیم در SQL این مناسبات را پیاده‌سازی کنیم، یک موجودیت جداگانه برای Trade تعریف کردیم که به دو موجودیت دیگر رابطه ی one-to-zero-or-many دارد؛ در نتیجه در فرزندان این موجودیت، یک کلید خارجی به پدرشان نگه‌داری می‌شود، ولی در پدر به آن‌ها لینکی وجود نخواهد داشت. همچنین فرزندان یک موجودیت ضعیف وابسته به پدر محسوب می‌شوند.

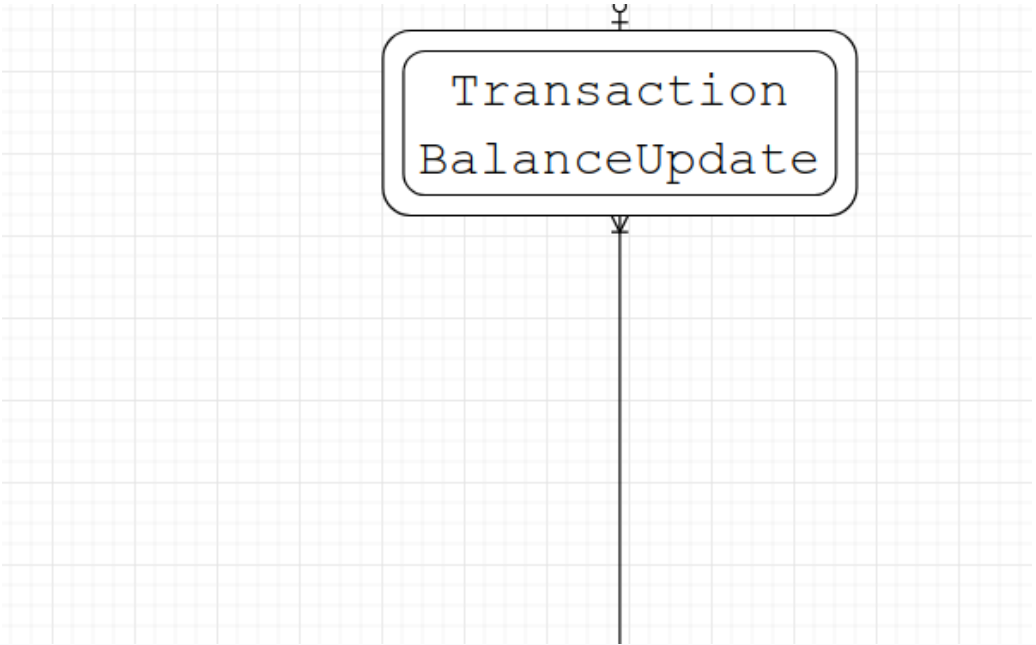


۲

. مشابه همین رویکرد را در مورد ChargeBalanceUpdate، TransactionBalanceUpdate و BalanceUpdate انجام دادیم.



۳. یک رابطه ی سه‌گانه بین تراکنش، TransactionBalanceUpdate و مجددا TransactionBalanceUpdate وجود داشت که آن را به یک رابطه ی یک به چند از تراکنش به TransactionBalanceUpdate تبدیل کردیم. (در واقع این رابطه همواره یک به دو است و به همین خاطر یک رابطه ی سه‌تایی می‌تواند آن را مدل کند، اما برای پیاده‌سازی آن در SQL از این نظم ویژه در رابطه صرف نظر کرده و آن را در فرم کلی تر یک به چند می‌بینیم).



در کنار این تغییرات، چند اصلاح جزئی نیز انجام دادیم که در فاز اول از چشم دور مانده بودند:

۱. در موجودیت کاربر چند ویژگی مانند ایمیل، کدملی و حساب بانکی را جا انداخته بودیم.

۲. پیش از این موجودیت P2PTrade به ActiveOrder متصل بود که در صورت fulfill شدن آن ActiveOrder و حذف شدن آن، کلید خارجی مربوطه در P2PTrade نامعتبر می‌شد. در نتیجه P2PTrade را به خود ArchivedOrder متصل کردیم.

۳. موجودیت ActiveOrder ضعیف در نظر گرفته نشده بود.

۴. موجودیت‌های Charge و ChargeBalanceUpdate را ادغام کردیم، زیرا جدا بودن آن‌ها سودی نداشت و ادغام آن‌ها همچنان یک موجودیت در حد قابل قبولی با معنا را ایجاد می‌کرد.

۵. رابطه ی has a last بین کیف پول و BalanceUpdate به صورت یک به یک بود که اشتباه است (هر کیف پول یک «آخرین به‌روزرسانی موجودی» دارد؛ اما این طور نیست که هر «به‌روزرسانی موجودی»، آخرین به‌روزرسانی موجودی یک کیف پول باشد). این رابطه به

One-to-zero-or-one

اصلاح شد.

۳.۳ نرمال‌ترسازی به 3NF

با بررسی طراحی خود، به این نتیجه رسیدیم که بدون انجام هیچ تغییری، جداول ایجاد شده در فرم نرمال سوم قرار خواهند داشت. پدیده‌ای که مشاهده کردیم این بود که اگر یک

وابستگی عملکردی غیر بدیهی در جدولی وجود داشت، دترمینان آن وابستگی می‌توانست کل اطلاعات سطر مورد نظر را تعیین کند، در نتیجه طراحی در فرم نرمال سوم قرار داشت.

با این وجود، در صورت طراحی پایگاه داده به شکلی دیگر، می‌توانست چنین اتفاقی نیفتد. برای مثال در صورتی که به جای جدا کردن کاربر و کیف پول‌هایش، اطلاعات کاربر را مستقیماً در هر کیف پول نگه‌داری کنیم، در این صورت کد ملی کاربر می‌توانست نام کاربری او را مشخص کند؛ اما فیلد دیگر مانند موجودی آن کیف پول را تعیین نمی‌کرد. در نتیجه در 3NF قرار نمی‌گرفتیم. در صورت اجرای چنین طراحی‌ای مشکلاتی هم در به‌روزرسانی پیش می‌آمد. برای مثال، برای تغییر اطلاعات یک کاربر (مثل رمز عبور)، بایستی تمام کیف‌پول های او را بیابیم و اطلاعات موجود در آن را تغییر دهیم.

همچنین در صورتی که به جای جداسازی موجودیت بازار، اطلاعات بازار را در سفارش‌های مربوط به آن نگه‌داری می‌کردیم، آی‌دی بازار (یا رمزارز آن) می‌توانست مثلا کارمزد آن را تعیین کند، ولی اطلاعات مربوط به سفارش (مانند قیمت واحد) را مشخص نمی‌کرد. بنابراین مجددا در 3NF قرار نمی‌گرفتیم. در این شرایط، مجددا مشکلاتی در به روز رسانی یا حذف اتفاق می‌افتاد. برای مثال برای تغییر میزان کارمزد یک بازار، بایستی مقدار فیلد «کارمزد» را در تمام سفارش های مربوط به آن تغییر می‌دادیم. همچنین در صورتی که یک بازار از سفارش تهی شود، در این سناریو اطلاعات آن به کلی از پایگاه داده پاک خواهد شد!

📝 فاز ۲ پروژه

> *Database Design*
>
> *Alireza Habibzadeh* `99109393`
> *Sina Imani* `99100314`
> *Ali Shahali* `400109905`
>
> *Spring 2024*

🌱 فصل ۴: بخش امتیازی: NoSQL + API

⛁ پیاده‌سازی دیتابیس

ابتدا سرور MongoDB را دریافت و نصب می‌کنیم و نهایتا سرویس آن را اجرا می‌کنیم:

```
brew tap mongodb/brew
brew install mongodb-community@7.0
brew services start mongodb/brew/mongodb-community
mongod
```

حال وارد شل مونگو می‌شویم تا کالکشن‌های لازم را بسازیم:

```
alireza@Alirezas-MacBook-Pro ~ % mongosh
Current Mongosh Log ID: 66802939bbba1d025bc80023
Connecting to: mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.2.10
Using MongoDB: 7.0.12
Using Mongosh: 2.2.10

For mongosh info see: https://docs.mongodb.com/mongodb-shell/


To help improve our products, anonymous usage data is collected and sent to MongoDB periodically (https://www.mongodb.com/legal/privacy-policy).
You can opt-out by running the disableTelemetry() command.

------
   The server generated these startup warnings when booting
   2024-06-29T19:01:02.250+03:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
------

test>
```

می‌بینیم که به درستی دیتابیس روی پورت پیش‌فرض ۲۷۰۱۷ بالا آمده است.

حال در این شل کالکشن‌های لازم را می‌سازیم:

```
test> disableTelemetry()  // تا یک وقت اطلاعات خیلی حساسمان به مونگو ارسال نشود :)
Telemetry is now disabled.
test> use exchange
```

```
switched to db exchange
exchange> db.createCollection('otc_prices')
{ ok: 1 }
exchange> db.createCollection('markets')
{ ok: 1 }
exchange> db.createCollection('otc_orders')
{ ok: 1 }
exchange> db.createCollection('p2p_orders')
{ ok: 1 }
exchange> db.createCollection('wallets')
{ ok: 1 }
exchange> db.createCollection('transactions')
{ ok: 1 }
exchange> db.createCollection('logs')
{ ok: 1 }
```

حال کمی دیتای تست اضافه می‌کنیم:

```
exchange> db.otc_prices.insertMany([
...   {currency: 'BTC', buy_price: 45000, sell_price: 45500},
...   {currency: 'ETH', buy_price: 3000, sell_price: 3100}
... ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('66802b56bbba1d025bc80024'),
    '1': ObjectId('66802b56bbba1d025bc80025')
  }
}
exchange> db.markets.insertMany([
...   {marketId: 'btc_usd', base_currency: 'BTC', quote_currency: 'USD', last_price: 45000, tradin
g_volume: 1000},
...   {marketId: 'eth_usd', base_currency: 'ETH', quote_currency: 'USD', last_price: 3000, trading
_volume: 5000}
... ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('66802b5fbbba1d025bc80026'),
    '1': ObjectId('66802b5fbbba1d025bc80027')
  }
}
```

فعلا هیچ authenticationای برای دیتابیس لحاظ نشده چون کلا لوکال است و دیتای خاصی هم نداریم ولی در صورتی که قرار است روی سرور باشد کار کند قطعا باید authetication برای آن لحاظ کنیم.

تنظیم وب‌سرور

حال می‌خواهیم اپ فلاسکی که زدیم را اجرا کنیم. به صورت پیش‌فرض اپ روی localhost اجرا می‌شود و نیازی به baseurl نداریم. در صورتی که سرور ما url داشته باشد این بخش را وب‌سرور یعنی احتمالا nginx هندل می‌کند و TLS را می‌زند و ریکوئست‌های ساده را به Flask می‌دهد. مثلا می‌توانیم از این کانفیگ nginx استفاده کنیم:

```
server {
```

```
  listen 80;
  server_name {{base_url}};
  location / {
      proxy_pass http://127.0.0.1:5000;
      proxy_set_header Host $host;
      proxy_set_header X-Real-IP $remote_addr;
      proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
      proxy_set_header X-Forwarded-Proto $scheme;
  }
}
server {
  listen 443 ssl;
  server_name {{base_url}};
  ssl_certificate /etc/letsencrypt/live/{{base_url}}/fullchain.pem;
  ssl_certificate_key /etc/letsencrypt/live/{{base_url}}/privkey.pem;

  location / {
      proxy_pass http://127.0.0.1:5000;
      proxy_set_header Host $host;
      proxy_set_header X-Real-IP $remote_addr;
      proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
      proxy_set_header X-Forwarded-Proto $scheme;
  }
}
```

که یعنی درخواست‌های پورت ۸۰ را به سادگی فروارد می‌کند و درخواست‌های پورت ۴۴۳ را با سرتیفیکیت base_url ما رمزنگاری کرده و سپس به همان پورت ۵۰۰۰ لوکال که وب‌سرویس فلاسک ما گوش می‌دهد می‌فرستد. در این صورت دیگر نیازی نیست فلاسک TLS را هندل کند. البته کمی گشتم و ظاهرا خود فلاسک هم می‌تواند این کار را انجام دهد که البته من همان روش ترکیب با nginx را ترجیح می‌دهم. برای این کار کافی است هنگام اجرا کردن فلاسک:

```
if __name__ == '__main__':
    context = ('/etc/letsencrypt/live/{{base_url}}/fullchain.pem',
               '/etc/letsencrypt/live/{{base_url}}/privkey.pem')  #certificate and key files
    app.run(debug=True, ssl_context=context)
```

برای تست حتی می‌توانیم از یک سرتیفیکیت self-signed هم استفاده کنیم:

```
if __name__ == '__main__':
    app.run(debug=True, ssl_context='adhoc')
```

🚀 اجرای سرویس و تست آن

سرویس فلاسک را در فایل `app.py` پیاده کرده‌ایم و همراه فایل‌های پروژه تحویل داده شده. آن را ابتدا بدون TLS اجرا می‌کنیم و سپس چند تست انجام می‌دهیم:

```
alireza@Alirezas-MacBook-Pro ~ % curl http://127.0.0.1:5000/api/otc/prices
[
  {
    "buy_price": 45000,
    "currency": "BTC",
    "sell_price": 45500
  },
  {
    "buy_price": 3000,
```

```
    "currency": "ETH",
    "sell_price": 3100
  }
]
```

که می‌بینیم به درستی قیمت‌ها بازگردانده شده‌اند. همچنین درخواستمان را در لاگ فلاسک می‌بینیم (به دلیل روشن بودن حالت debug):

```
/opt/homebrew/bin/python3.12 /Users/alireza/PycharmProjects/db-spring2024/app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 334-430-471
127.0.0.1 - - [29/Jun/2024 19:16:46] "GET /api/otc/prices HTTP/1.1" 200 -
```

این یعنی هم وب‌سرویس به درستی کار می‌کند و هم ارتباط با دیتابیس به درستی برقرار می‌شود. یک بار هم با TLS خود فلاسک اجرا کردم (و با سرتیفیکیت self-signed):

```
* Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on https://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 334-430-471
127.0.0.1 - - [29/Jun/2024 20:05:44] "GET /api/otc/prices HTTP/1.1" 200 -
```

که همان‌طور که می‌بینید از https استفاده می‌کند. (اینجا مثلا nginx دیگر دو پروتکل روی دو پورت نداریم و حتما باید با https روی پورت ۵۰۰۰ کار کنیم.)

```
alireza@Alirezas-MacBook-Pro ~ % curl https://127.0.0.1:5000/api/otc/prices
curl: (60) SSL certificate problem: self signed certificate
More details here: https://curl.se/docs/sslcerts.html

curl failed to verify the legitimacy of the server and therefore could not
establish a secure connection to it. To learn more about this situation and
how to fix it, please visit the web page mentioned above.
```

که مطابق انتظار است چون سرتیفیکیت self-signed است و trust شده نیست. با اسکیپ کردن این چک:

```
alireza@Alirezas-MacBook-Pro ~ % curl https://127.0.0.1:5000/api/otc/prices --insecure
[
  {
    "buy_price": 45000,
    "currency": "BTC",
    "sell_price": 45500
  },
  {
```

```
    "buy_price": 3000,
    "currency": "ETH",
    "sell_price": 3100
  }
]
```

و تامام.

🧪 تست بیشتر

قابلیت‌های خواسته شده در داک را تست می‌کنیم و البته دیتای تست بیشتری نیز نیاز داریم.

OTC Prices

در بخش قبل تست شد.

P2P Markets

```
alireza@Alirezas-MacBook-Pro ~ % curl http://127.0.0.1:5000/api/markets
[
  {
    "base_currency": "BTC",
    "last_price": 45000,
    "marketId": "btc_usd",
    "quote_currency": "USD",
    "trading_volume": 1000
  },
  {
    "base_currency": "ETH",
    "last_price": 3000,
    "marketId": "eth_usd",
    "quote_currency": "USD",
    "trading_volume": 5000
  }
]
```

OTC Order

اضافه کردن

```
alireza@Alirezas-MacBook-Pro-6 ~ % curl -X POST http://127.0.0.1:5000/api/otc/orders \
    -H "Content-Type: application/json" \
    -d '{
        "userId": "12345",
        "currency": "BTC",
        "amount": 1.0,
        "price": 45000.00,
        "type": "buy"
    }'
{
  "message": "OTC order created successfully",
  "orderId": "6680585290f9c1289560c647",
  "status": "created"
}
```

مشاهده‌ی سفارشات قبلی

```
alireza@Alirezas-MacBook-Pro ~ % curl http://127.0.0.1:5000/api/otc/orders
{
  "error": "userId is required"
}
alireza@Alirezas-MacBook-Pro ~ % curl 'http://127.0.0.1:5000/api/otc/orders?userId=999'
[]
alireza@Alirezas-MacBook-Pro ~ % curl 'http://127.0.0.1:5000/api/otc/orders?userId=12345'
[
  {
    "amount": 1.0,
    "currency": "BTC",
    "price": 45000.0,
    "status": "pending",
    "timestamp": "Sat, 29 Jun 2024 18:54:10 GMT",
    "type": "buy",
    "userId": "12345"
  }
]
```

کنسل کردن

```
alireza@Alirezas-MacBook-Pro ~ % curl -X DELETE 'http://127.0.0.1:5000/api/otc/orders'
{
  "error": "orderId is required"
}
alireza@Alirezas-MacBook-Pro ~ % curl -X DELETE 'http://127.0.0.1:5000/api/otc/orders?orderId=123'
{
  "error": "Invalid orderId format"
}
alireza@Alirezas-MacBook-Pro ~ % curl -X DELETE 'http://127.0.0.1:5000/api/otc/orders?orderId=6680585290f9c1289560c647'
{
  "message": "OTC order cancelled successfully",
  "orderId": "6680585290f9c1289560c647",
  "status": "cancelled"
}
alireza@Alirezas-MacBook-Pro ~ % curl -X DELETE 'http://127.0.0.1:5000/api/otc/orders?orderId=999999999999999999999'
{
  "error": "Order not found"
}
```

P2P Order

اضافه کردن

```
alireza@Alirezas-MacBook-Pro ~ % curl -X POST http://127.0.0.1:5000/api/p2p/orders \
    -H "Content-Type: application/json" \
    -d '{
        "userId": "12345",
        "marketId": "btc_usd",
        "amount": 1.0,
        "price": 45000.00,
```

```
        "type": "buy"
    }'
{
  "message": "P2P order created successfully",
  "orderId": "66805b9f374cd76950c2a42f",
  "status": "created"
}
```

کنسل کردن

```
alireza@Alirezas-MacBook-Pro ~ % curl -X DELETE 'http://127.0.0.1:5000/api/p2p/orders?orderId=123'
{
  "error": "Invalid orderId format"
}
alireza@Alirezas-MacBook-Pro ~ % curl -X DELETE 'http://127.0.0.1:5000/api/p2p/orders?orderId=66805b9f374cd76950c2a42f'
{
  "message": "P2P order cancelled successfully",
  "orderId": "66805b9f374cd76950c2a42f",
  "status": "cancelled"
}
alireza@Alirezas-MacBook-Pro ~ % curl -X DELETE 'http://127.0.0.1:5000/api/p2p/orders?orderId=999999999999999999999999'
{
  "error": "Order not found"
}
```

مشاهده‌ی سفارشات قبلی

```
alireza@Alirezas-MacBook-Pro ~ % curl 'http://127.0.0.1:5000/api/p2p/orders?userId=12345'

[
  {
    "amount": 1.0,
    "marketId": "btc_usd",
    "price": 45000.0,
    "status": "pending",
    "timestamp": "Sat, 29 Jun 2024 19:07:09 GMT",
    "type": "buy",
    "userId": "12345"
  },
  {
    "amount": 1.0,
    "marketId": "btc_usd",
    "price": 45000.0,
    "status": "canceled",
    "timestamp": "Sat, 29 Jun 2024 19:08:15 GMT",
    "type": "buy",
    "userId": "12345"
  }
]
```

```
alireza@Alirezas-MacBook-Pro ~ % curl 'http://127.0.0.1:5000/api/p2p/orders?userId=12345&status=canceled'
[
  {
    "amount": 1.0,
    "marketId": "btc_usd",
    "price": 45000.0,
    "status": "canceled",
    "timestamp": "Sat, 29 Jun 2024 19:08:15 GMT",
    "type": "buy",
    "userId": "12345"
  }
]
```

نکته‌ی دیگر این که API کد http درستی هم برمی‌گرداند مثلا برای خطاها کد 400 و یا 404 بسته به خطا برگردانده می‌شود.

Wallets

کمی داده‌ی تست برای wallets درست می‌کنیم:

```
use exchange

db.wallets.insertMany([
    {
        "_id": ObjectId(),  // Automatically generate an ObjectId for the wallet ID
        "userId": "12345",
        "address": "1A2b3C4d5E6F7G8H9I0J",
        "balance": 10.5,
        "currency": "BTC"
    },
    {
        "_id": ObjectId(),  // Automatically generate an ObjectId for the wallet ID
        "userId": "12345",
        "address": "2B3C4D5E6F7G8H9I0J1A",
        "balance": 25.0,
        "currency": "ETH"
    },
    {
        "_id": ObjectId(),  // Automatically generate an ObjectId for the wallet ID
        "userId": "12345",
        "address": "3C4D5E6F7G8H9I0J1A2B",
        "balance": 50.0,
        "currency": "USDT"
    },
    {
        "_id": ObjectId(),  // Automatically generate an ObjectId for the wallet ID
        "userId": "67890",
        "address": "J0I9H8G7F6E5D4C3B2A1",
        "balance": 5.75,
        "currency": "ETH"
    },
    {
        "_id": ObjectId(),  // Automatically generate an ObjectId for the wallet ID
```

```
        "userId": "11121",
        "address": "K9L8M7N6O5P4Q3R2S1T",
        "balance": 20.0,
        "currency": "BTC"
    },
    {
        "_id": ObjectId(),  // Automatically generate an ObjectId for the wallet ID
        "userId": "67890",
        "address": "E1D2C3B4A5F6G7H8I9J0",
        "balance": 12.5,
        "currency": "BTC"
    }
])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('66805e8cbbba1d025bc80028'),
    '1': ObjectId('66805e8cbbba1d025bc80029'),
    '2': ObjectId('66805e8cbbba1d025bc8002a'),
    '3': ObjectId('66805e8cbbba1d025bc8002b'),
    '4': ObjectId('66805e8cbbba1d025bc8002c'),
    '5': ObjectId('66805e8cbbba1d025bc8002d')
  }
}
```

حال:

```
alireza@Alirezas-MacBook-Pro ~ % curl 'http://127.0.0.1:5000/api/wallets?userId=12345'
[
  {
    "address": "1A2b3C4d5E6F7G8H9I0J",
    "balance": 10.5,
    "currency": "BTC",
    "userId": "12345"
  },
  {
    "address": "2B3C4D5E6F7G8H9I0J1A",
    "balance": 25,
    "currency": "ETH",
    "userId": "12345"
  },
  {
    "address": "3C4D5E6F7G8H9I0J1A2B",
    "balance": 50,
    "currency": "USDT",
    "userId": "12345"
  }
]
alireza@Alirezas-MacBook-Pro ~ % curl 'http://127.0.0.1:5000/api/wallets?userId=999'
[]
```

Transfer and Transaction History

انتقال پول

```
alireza@Alirezas-MacBook-Pro ~ % curl -X POST http://127.0.0.1:5000/api/transactions \
  -H "Content-Type: application/json" \
  -d '{
    "fromUserId": "11121",
    "toUserId": "12345",
    "amount": 2.0,
    "currency": "BTC"
  }'

{
  "message": "Money transferred successfully",
  "status": "success",
  "transactionId": "668060124e124f5a827a1478"
}
alireza@Alirezas-MacBook-Pro ~ % curl -X POST http://127.0.0.1:5000/api/transactions \
  -H "Content-Type: application/json" \
  -d '{
    "fromUserId": "11121",
    "toUserId": "12345",
    "amount": 100.0,
    "currency": "BTC"
  }'

{
  "error": "Insufficient funds"
}
```

می‌بینیم که مقدار به ولت بیتکوین ۱۲۳۴۵ آمده:

```
alireza@Alirezas-MacBook-Pro ~ % curl 'http://127.0.0.1:5000/api/wallets?userId=12345'
[
  {
    "address": "1A2b3C4d5E6F7G8H9I0J",
    "balance": 12.5,
    "currency": "BTC",
    "userId": "12345"
  },
  {
    "address": "2B3C4D5E6F7G8H9I0J1A",
    "balance": 25,
    "currency": "ETH",
    "userId": "12345"
  },
  {
    "address": "3C4D5E6F7G8H9I0J1A2B",
    "balance": 50,
    "currency": "USDT",
    "userId": "12345"
```

```
  }
]
```

و از 11121 کم شده: (مقادیر را با بخش قبل مقایسه کنید)

```
alireza@Alirezas-MacBook-Pro ~ % curl 'http://127.0.0.1:5000/api/wallets?userId=11121'
[
  {
    "address": "K9L8M7N6O5P4Q3R2S1T",
    "balance": 18.0,
    "currency": "BTC",
    "userId": "11121"
  }
]
```

تاریخچه‌ی مالی

کوئری دیتابیس را این‌طوری می‌زنیم تا هم در صورتی که فرستنده باشد و هم گیرنده بیاید:

```
def get_transaction_history(user_id, limit=10, offset=0):
    return list(db.transactions.find({'$or': [{'fromUserId': user_id}, {'toUserId': user_id}]})
                .skip(offset).limit(limit).sort('timestamp', -1))

```

حال کلی انتقال از 11121 به 12345 و برعکس درست کردم.

```
alireza@Alirezas-MacBook-Pro ~ % curl 'http://127.0.0.1:5000/api/transactions/history'

{
  "error": "userId is required"
}
alireza@Alirezas-MacBook-Pro ~ % curl 'http://127.0.0.1:5000/api/transactions/history?userId=1234
5'
[
  {
    "_id": "668062deb35a0a111b7843b8",
    "amount": 2.0,
    "currency": "BTC",
    "fromUserId": "12345",
    "timestamp": "Sat, 29 Jun 2024 19:39:10 GMT",
    "toUserId": "11121"
  },
  {
    "_id": "668062ceb35a0a111b7843b6",
    "amount": 2.0,
    "currency": "BTC",
    "fromUserId": "11121",
    "timestamp": "Sat, 29 Jun 2024 19:38:54 GMT",
    "toUserId": "12345"
  },
  {
    "_id": "668062cdb35a0a111b7843b4",
    "amount": 2.0,
```

```
    "currency": "BTC",
    "fromUserId": "11121",
    "timestamp": "Sat, 29 Jun 2024 19:38:53 GMT",
    "toUserId": "12345"
  },
  {
    "_id": "668062ccb35a0a111b7843b2",
    "amount": 2.0,
    "currency": "BTC",
    "fromUserId": "11121",
    "timestamp": "Sat, 29 Jun 2024 19:38:52 GMT",
    "toUserId": "12345"
  },
  {
    "_id": "668062cbb35a0a111b7843b0",
    "amount": 2.0,
    "currency": "BTC",
    "fromUserId": "11121",
    "timestamp": "Sat, 29 Jun 2024 19:38:51 GMT",
    "toUserId": "12345"
  },
  {
    "_id": "668062cab35a0a111b7843ae",
    "amount": 2.0,
    "currency": "BTC",
    "fromUserId": "11121",
    "timestamp": "Sat, 29 Jun 2024 19:38:50 GMT",
    "toUserId": "12345"
  },
  {
    "_id": "668062c9b35a0a111b7843ac",
    "amount": 2.0,
    "currency": "BTC",
    "fromUserId": "11121",
    "timestamp": "Sat, 29 Jun 2024 19:38:49 GMT",
    "toUserId": "12345"
  },
  {
    "_id": "668062c6b35a0a111b7843aa",
    "amount": 2.0,
    "currency": "BTC",
    "fromUserId": "11121",
    "timestamp": "Sat, 29 Jun 2024 19:38:46 GMT",
    "toUserId": "12345"
  },
  {
    "_id": "668060124e124f5a827a1478",
    "amount": 2.0,
    "currency": "BTC",
    "fromUserId": "11121",
```

```
    "timestamp": "Sat, 29 Jun 2024 19:27:14 GMT",
    "toUserId": "12345"
  }
]
```

می‌بینیم وقتی هم فرستنده بوده و هم گیرنده آمده. حال تست offset و limit:

```
alireza@Alirezas-MacBook-Pro ~ % curl 'http://127.0.0.1:5000/api/transactions/history?userId=12345
&limit=2&offset=1'
[
  {
    "_id": "668062ceb35a0a111b7843b6",
    "amount": 2.0,
    "currency": "BTC",
    "fromUserId": "11121",
    "timestamp": "Sat, 29 Jun 2024 19:38:54 GMT",
    "toUserId": "12345"
  },
  {
    "_id": "668062cdb35a0a111b7843b4",
    "amount": 2.0,
    "currency": "BTC",
    "fromUserId": "11121",
    "timestamp": "Sat, 29 Jun 2024 19:38:53 GMT",
    "toUserId": "12345"
  }
]
```

Logs

همه‌ی کارهایی که کردیم در logs ذخیره شده. فایل کاملش را در کنار پروژه در `logs.json` قرار دادم. یک نمونه‌ی کوتاه شده:

```
exchange> db.logs.find()
[
  {
    _id: ObjectId('6680552909881e203c9d2f33'),
    timestamp: ISODate('2024-06-29T18:40:41.762Z'),
    event: 'set_otc_order',
    details: {
      userId: '12345',
      currency: 'BTC',
      amount: 1,
      price: 45000,
      type: 'buy',
      timestamp: ISODate('2024-06-29T18:40:41.757Z'),
      _id: ObjectId('6680552909881e203c9d2f32')
    }
  },
  {
    _id: ObjectId('66805a22e88ca3d3729fc873'),
    timestamp: ISODate('2024-06-29T19:01:54.261Z'),
    event: 'cancel_otc_order',
    details: { orderId: '6680585290f9c1289560c647' }
```

```
  },
  {
    _id: ObjectId('66805ab1e88ca3d3729fc877'),
    timestamp: ISODate('2024-06-29T19:04:17.220Z'),
    event: 'set_p2p_order',
    details: {
      userId: '12345',
      marketId: 'btc_usd',
      amount: 1,
      price: 45000,
      type: 'buy',
      status: 'pending',
      timestamp: ISODate('2024-06-29T19:04:17.219Z'),
      _id: ObjectId('66805ab1e88ca3d3729fc876')
    }
  },
  {
    _id: ObjectId('66805c614e124f5a827a1477'),
    timestamp: ISODate('2024-06-29T19:11:29.694Z'),
    event: 'cancel_p2p_order',
    details: { orderId: '66805b9f374cd76950c2a42f' }
  },
  {
    _id: ObjectId('668060124e124f5a827a1479'),
    timestamp: ISODate('2024-06-29T19:27:14.610Z'),
    event: 'transfer_money',
    details: {
      fromUserId: '11121',
      toUserId: '12345',
      amount: 2,
      currency: 'BTC',
      timestamp: ISODate('2024-06-29T19:27:14.608Z'),
      _id: ObjectId('668060124e124f5a827a1478')
    }
  },
  {
    _id: ObjectId('668062c6b35a0a111b7843ab'),
    timestamp: ISODate('2024-06-29T19:38:46.269Z'),
    event: 'transfer_money',
    details: {
      fromUserId: '11121',
      toUserId: '12345',
      amount: 2,
      currency: 'BTC',
      timestamp: ISODate('2024-06-29T19:38:46.268Z'),
      _id: ObjectId('668062c6b35a0a111b7843aa')
    }
  }
]
```

📁 خروجی

با دستورات مشابه برای همه‌ی کالکشن‌ها خروجی json آن‌ها در کنار پروژه ضمیمه شده:

```
mongoexport --collection=transactions --db=exchange --out=transactions.json
```