دانشگاه صنعتی شریف

پروژه‌ی صرافی ارز دیجیتال: پایگاه داده - فاز اول

استاد: دکتر مهدی آخی

شمارهٔ تیم: ۱۴

مانی ابراهیمی	محمدامین حیدری	محمد جعفری‌پور
۴۰۱۱۷۰۴۹۱	۴۰۱۱۷۰۵۵۳	۴۰۱۱۰۵۷۹۷

بهار ۱۴۰۳

فهرست مطالب

فصل ۱

کلیت فاز اول پروژه

۱.۱ شرح

در این فاز تلاش شده تا یک پایگاه داده‌ی مرتبط با یک صرافی ارز دیجیتال طراحی شود. این پایگاه داده شامل موجودیت‌هایی مانند کاربر و کیف پول و تراکنش است. همچنین برای هر موجودیت روابطی با موجودیت‌های دیگر نیز تعریف شده است. در ادامه به توضیح هر یک از موجودیت‌ها و روابط آن‌ها با موجودیت‌های دیگر پرداخته‌ایم. همچنین در انتها پاسخ به ۱۰ پرسش جبر رابطه‌ای داده شده نیز آمده است. مخزن یا همان repository **این پروژه در اینجا[1] قابل مشاهده است.**

۱.۲ تقسیم وظایف

تیم این پروژه متشکل از سه نفر بود که برای سادگی در سند تقسیم وظایف، برای آن‌ها از اسم کوتاه استفاده کردیم:

نام کوتاه	نام کامل	شماره دانشجویی
Mani	مانی ابراهیمی	۴۰۱۱۷۰۴۹۱
Mamadamin	محمدامین حیدری	۴۰۱۱۷۰۵۵۳
Mamal	محمد جعفری‌پور	۴۰۱۱۰۵۷۹۷

جدول ۱.۱: جدول اعضای تیم در جدول تقسیم وظایف

جدول تقسیم وظایف نیز از اینجا[2] قابل مشاهده است.

[1] **در صورتی که لینک برای شما کار نمی‌کند، از آدرس** https://github.com/maniebra/dbms-exchange-project **استفاده نمایید.**

[2] **در صورتی که این لینک برای شما کار نمی‌کند، میتوانید از آدرس**
https://docs.google.com/spreadsheets/d/1x1Guh4HTWLyG9GTomZEtesp5cjIGez9m9Day3bS_kgM/edit?usp=sharing
استفاده نمایید.

فصل ۲

دیاگرام های ER

۲.۱ شرح

در این بخش تلاش بر این بود که کلیت پایگاه داده‌ی مورد نظر را با استفاده از دیاگرام‌های ER نمایش دهیم. ابتدا دیاگرام ER اصلی را نمایش داده‌ایم و سپس به تفکیک بخش‌های مختلف آن پرداخته‌ایم.

۲.۲ توضیح هر موجودیت

در ادامه، برای هر موجودیت حاضر در این دیاگرام توضیحی آمده:

۲.۲.۱ User

موجودیت کاربر یا همان user، که دارای صفات گفته شده از جمله نام و نام خانوادگی و شناسه ملی و شماره تماس و ایمیل و رمز عبور و سایر موارد است. این موجودیت برای کاربران اصلی‌ترین موجودیت بوده چرا که اطلاعات خود هر کاربر را در این موجودیت ذخیره می‌کنیم. همچنین به یک موجودیت کیف پول متصل است که باعث می‌شود هر کاربر یک کیف پول داشته باشد.

۲.۲.۲ Wallet

موجودیت کیف پول یا همان wallet، که دارای صفات گفته شده از جمله موجودیت کاربر و موجودی و ارز و سایر موارد است. این موجودیت برای ذخیره‌ی اطلاعات مربوط به کیف پول هر ارز از هر کاربر استفاده می‌شود. همچنین به یک موجودیت تراکنش متصل است که باعث می‌شود هر کیف پول دارای تراکنش باشد.

۲.۲.۳ Transactions

موجودیت تراکنش یا همان transactions، که دارای صفات گفته شده از جمله موجودیت کیف پول و نوع تراکنش و مبلغ و تاریخ و سایر موارد است. این موجودیت برای ذخیره‌ی اطلاعات مربوط به تراکنش‌های هر کیف پول استفاده می‌شود. در هر تراکنش مقداری ارز از کیف پول یک کاربر خارج شده و به کیف پول کاربری دیگر می‌رود. در نظر داشته باشید که هر تبادل، دو تراکنش است.

همچنین صفت fee در تراکنش با داشتن Market_id و بدست آوردن ارز پایه‌ی آن مارکت و قیمت لحظه‌ای آن ارز پایه به ریال محاسبه می‌شود.

۲.۲.۴ Orders

موجودیت سفارش ها یا همان Orders، که دارای صفات گفته شده از جمله تاریخ و وضعیت و نوع ارز و حجم و قیمت و سایر موارد است. این موجودیت برای ذخیره اطلاعات مربوط به سفارشات کاربران می باشد و دارای دو نوع خرید و فروش می باشد. همچنین به یک موجودیت تبادل متصل است در اصل ترکیب دو سفارش خرید و فروش می باشد.

۲.۲.۵ Trades

موجودیت تبادل ها یا همان Trades، که دارای صفات گفته شده از جمله تاریخ و حجم و مقدار و سایر موارد است. این موجودیت برای ذخیره اطلاعات مربوط به تبادل ها می باشد که تبادل ها میتواند بین یک کاربر و ادمین سایت و یا دو کاربر باشد که به ترتیب دو موجودیت OTC و P2Pرو تشکیل داده اند. همچنین این موجودیت دارای یک شناسه برای هر تبادل می باشد. صفت `min_fill_remainder` به این صورت عمل می‌کند که حجم باقی‌مانده‌ی کمینه‌ی دو سفارش خرید و فروش را ذخیره می‌کند.

OTC

شامل ID ادمین و مشتری می‌باشد که بوسیله‌ی شناسه‌ی Market به بازار مربوطه متصل شده است.

P2P

شامل دو ID و OrderID خریدار و فروشنده یا همان maker و taker می‌باشند که به وسیله‌ی شناسه‌ی صرافی یا همان Broker_ID به صرافی مربوطه متصل شده‌اند.

۲.۲.۶ OrderBooks

موجودیت لیست سفارشات یا همان OrderBooks، که دارای صفات گفته شده از جمله شناسه و شناسه ی بازار و و سایر موارد است. این موجودیت برای ذخیره اطلاعات مربوط به لیست های سفارشات هر فروشگاه میباشد، همچنین به یک موجودیت لیست که زیرمجموعه‌ی OrderBooks است متصل شده که شامل دو نوع لیست خرید و فروش می باشد و به موجودیت سفارشات که خود دو نوع خرید و فروش دارد نیز متصل است که در نهایت این دو نوع خرید و فروش با هم سفارشات را بتواند بسازد.

۲.۲.۷ Markets

موجودیت فروشگاه ها یا همان Markets، که دارای صفات گفته شده از جمله کارمزد و قیمت لحظه ای بازار و نوع ارز پایه و سایر موارد است. این موجودیت برای ذخیره اطلاعات مربوط به فروشگاه های خرید و فروش ارز دیجیتال برای کاربران می باشد. همچنین به یک موجودیت لیست سفارشات متصل است که شامل دو لیست خرید و فروش هر فروشگاه می باشد.

۲.۲.۸ Brokers

موجودیت صرافی ها یا همان Brokers، که دارای صفات گفته شده از جمله شناسه و سایر موارد است. این موجودیت برای ذخیره اطلاعت مربوط به صرافی های ارز دیجیتال می باشد. همچنین به موجودیت فروشگاه ها متصل می باشد که برای هر ارز پایه در صرافی یک فروشگاه وجود دارد و به یک یا چند admin متصل است که در ان ادمین های هر صرافی مشخص می شوند.

۲.۲.۹ CryptoCurrency

موجودیت کریپتو ها یاCryptoCurrency ارز هایی اند که در سایت وجود دارند و توسط افراد مبادله میشوند. این ارز ها ممکن است قیمت ثابت Stable coinباشند و یا قیمت انها هرلحظه عوض شود nonstable Currency.

۲.۲.۱۰ Network

هر ارز شامل چندین شبکه ی مجزا از هم است که تراکنشهای آنها روی بستر متفاوتی انحام میشود.این شبکه ها دارای کارمزد و زمان متفاوتی اند.

۲.۲.۱۱ Online Payments

تاریخچه‌ی تمامی واریزی‌های هر کاربر، مقدار آن و زمان انجام‌شده است.

۲.۲.۱۲ Wallet History

تاریخچه‌ای از تغییرات میزان هر کیف پول است و هر تراکنشی برای دو کیف پول یک Wallet History جدید می‌سازد.

۲.۲.۱۳ Crypto Histories

تاریخچه‌ی تغییرات قیمت یک رمزارز است که زمان آن تغییر و مقدار و قیمت آن در آن زمان (قیمت همان قیمت لحظه‌ای مارکت است) نشان می‌دهد. با انحام هر تراکنش یک CryptoHistory جدید ایجاد می‌شود چرا که قیمت لحظه‌ای ارز تغییر می‌کند.

فصل ۳

سوالات جبر رابطه‌ای

۳.۱ شرح

در این بخش پاسخ به ۱۰ سوال جبر رابطه‌ای[۱] آمده است.

۳.۲ پاسخ به سوالات

۳.۲.۱ سوال ۱

$$\Pi_{market\_id, fee}(Transactions \bowtie_{Transactions.market\_id=Market.market\_id \wedge Transactions.date=date} (\ _{market\_id}\mathcal{F}_{\max(date)}(Market \bowtie_{Market.market\_id=Transactions.market\_id} Transactions)))$$

۳.۲.۲ سوال ۲

$$_{owner\_id}\mathcal{F}_{\texttt{Sum}(total\_value \times in\_time\_price)}[Wallets \bowtie_{Market.market\_id=id} Markets]$$

۳.۲.۳ سوال ۳

$$_{crypto\_id}\mathcal{F}_{\texttt{Count}(order\_id)}[\sigma_{fill=" false"}(Orders)]$$

۳.۲.۴ سوال ۴

$$A = \rho_{user\_id, total}[_{owner\_id}\mathcal{F}_{\texttt{Sum}(fee) as total Sell}(Transactions \bowtie_{Transactions.origin\_wallet\_id=wallets.id} Wallet)]$$
$$B = \rho_{user\_id, total}[_{owner\_id}\mathcal{F}_{\texttt{Sum}(fee) as total Buy}(Transactions \bowtie_{Transactions.dest\_wallet\_id=wallets.id} Wallet)]$$

$$_{user\_id}\mathcal{F}_{\texttt{Sum}(Total)}$$

[۱] Relational Algebra

۳.۲.۵ سوال ۵

$$A =_{user\_id,cryptoid} \mathcal{F}_{\texttt{Count}(Transactions.id)}((Users \times Cryptocurrency) \ltimes_{users.user\_id=Transactions.SellerID} Transactions)$$

$$B =_{user\_id,cryptoid} \mathcal{F}_{\texttt{Count}(Transactions.id)}((Users \times Cryptocurrency) \ltimes_{users.user\_id=Transactions.BuyerID} Transactions)$$

$$_{user\_id,cryptoid}\mathcal{F}_{mathttSum(TotalCount)}(\rho_{user\_id,cryptoid/TotalCount(A)} \cup \rho_{user\_id,cryptoid/TotalCount(B)})$$

۳.۲.۶ سوال ۶

$$\mathcal{F}_{\texttt{Sum}(fee)}[\sigma_{Now-Date\geq\text{"}0000-00-30-00:00:00\text{"}}(Transactions)]$$

۳.۲.۷ سوال ۷

$$A = \Pi_{cryptoid,in\_time\_price}(Cryptocurrency)$$
$$B =_{cryptoid} \mathcal{F}_{mathttmax(Date)asDate}(\sigma_{Transactions.Date-Now()\leq\text{"}0000-00-30-00:00:00\text{"}}(Transactions \rtimes Cryptocurrency))$$
$$C = \Pi_{cryptoid,fee}[Cryptocurrency \bowtie_{Cryptocurrency.idTransactions.cryptoid} (Transactions \bowtie B)]$$

$$\Pi_{cryptoid,in\_time\_price-fee}(A \bowtie)C$$

۳.۲.۸ سوال ۸

$$A =_{owner\_id} \mathcal{F}_{\texttt{Sum}(Total\_value)assum}(Wallets)$$
$$B = \Pi_{owner\_id,cryptoid,Total\_value}(Wallets)$$
$$_{cryptoid}\mathcal{F}_{\texttt{count}(owner\_id)}[\sigma_{percentage\geq 0.05}(\rho_{cryptoid,owner\_id,percentage}[\Pi_{cryptoid,owner\_id,\frac{Total\_value}{Sum}}(A \bowtie B)])]$$

۳.۲.۹ سوال ۹

$$A = \Pi_{user\_id,Date}(\sigma_{Date-Now()\leq\text{"}0000-00-30;00:00:00\text{"}}(Online\_Payments))$$
$$B = \rho_{user\_id,paymentDate}(A) \bowtie WalletHistories$$
$$C =_{cryptoid,user\_id,paymentDate} \mathcal{F}_{\texttt{Max}(Date)}(\sigma_{Date<paymentDate}(B))$$
$$\rho_{user_id,paymentDate}(A) \times CryptoHistories$$
$$E =_{cryptoid,user\_id,paymentDate} \mathcal{F}_{\texttt{max}(Date)}(\sigma_{Date<paymentDate}(D))$$
$$X =_{user\_id,paymentDate} \mathcal{F}_{\texttt{Sum}(amount\times price)astotalValue}((C \times WalletHistories) \bowtie_{user\_id=user\_id\wedge paymentDate=paymentDate} E \bowtie CryptoHistories) \bowtie_{user\_id=user\_id\wedge paymentDate=paymentDate} Online\_Payments$$

$$\mathcal{F}_{\texttt{CountUnique(user\_id)}}(\sigma_{onlineamount\geq\frac{1}{5}totalValue}(X))$$

۳.۲.۱۰ سوال ۱۰

$$A = \rho_{cryptoid,price,totalSell}[_{cryptoid,price}\mathcal{F}_{\texttt{Sum(amount)}}((Cryptocurrency \times prices) \bowtie_{Cryptocurrency.id=sellOrders.cryptoid} sellOrders)]$$

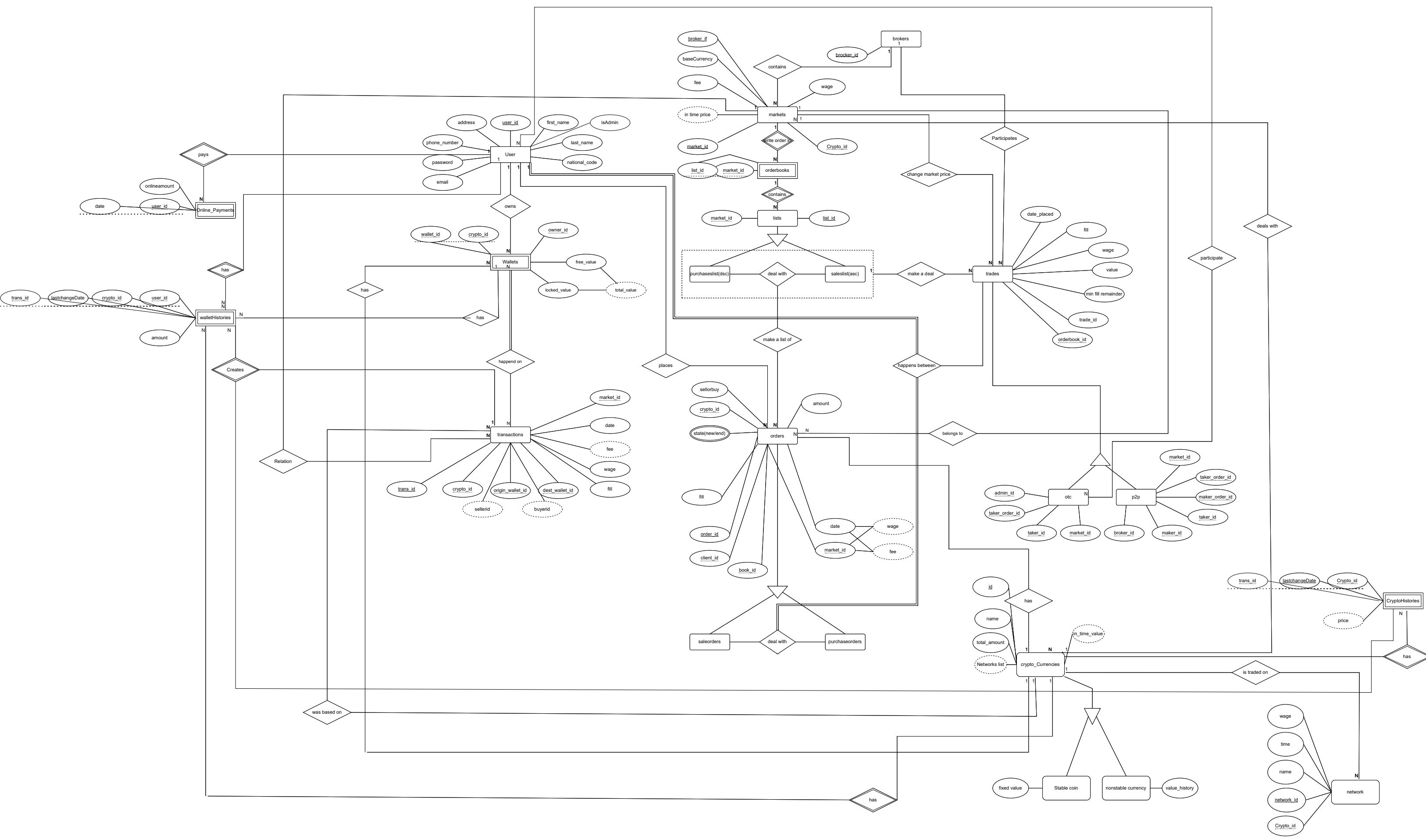
$$B = \rho_{cryptoid,price,totalSell}[_{cryptoid,price}\mathcal{F}_{\texttt{Sum(amount)}}((Cryptocurrency \times prices) \bowtie_{Cryptocurrency.id=purchaseOrders.cryptoid} purchaseOrders)]$$

$$A \cup B$$

فصل ۴

ضمیمه: تصویر دیاگرام ER

در انتهای فایل، ضمیمه‌ی تصویر دیاگرام مربوطه آمده است.
در صورتی که در مشاهده‌ی این تصویر مشکل دارید، فایل PDF را با مرورگرهای Edge یا Chrome باز نمایید.


User
address
user_id
first_name
isAdmin
phone_number
last_name
password
national_code
email
pays
onlineamount
date
user_id
Online_Payments
owns
wallet_id
crypto_id
owner_id
Wallets
free_value
locked_value
total_value
has
has
trans_id
lastchangeDate
crypto_id
user_id
walletHistories
amount
has
Creates
happend on
market_id
transactions
date
fee
wage
fill
trans_id
crypto_id
origin_wallet_id
dest_wallet_id
sellerid
buyerid
Relation
was based on
has
broker_if
baseCurrency
fee
in time price
market_id
markets
contains
brokers
brocker_id
wage
Crypto_id
write order in
list_id
market_id
orderbooks
contains
market_id
lists
list_id
purchaseslist(dsc)
deal with
saleslist(asc)
make a deal
trades
change market price
Participates
date_placed
fill
wage
value
min fill remainder
trade_id
orderbook_id
make a list of
places
happens between
deals with
participate
sellorbuy
crypto_id
amount
state(new/end)
orders
belongs to
fill
order_id
client_id
book_id
date
market_id
wage
fee
saleorders
deal with
purchaseorders
otc
p2p
admin_id
taker_order_id
taker_id
market_id
market_id
taker_order_id
maker_order_id
taker_id
broker_id
maker_id
id
has
name
total_amount
Networks list
in_time_value
crypto_Currencies
is traded on
Stable coin
nonstable currency
fixed value
value_history
trans_id
lastchangeDate
Crypto_id
CryptoHistories
price
has
wage
time
name
network_id
Crypto_id
network

فصل ۵

کلیت فاز دوم پروژه

۵.۱ شرح

ما در این فاز از پروژه چهار تغیر در نموار فاز قبلی خود ایجاد کردیم که به ترتیب عبارتند از بهینه کردن نمودار برای طراحی دیتابیس، ایجاد دیتابیس، نرمال سازی و ایندکس کردن دیتا بیس و در نهایت انحام هشت جستوجو در دیتابیس.

فصل ۶

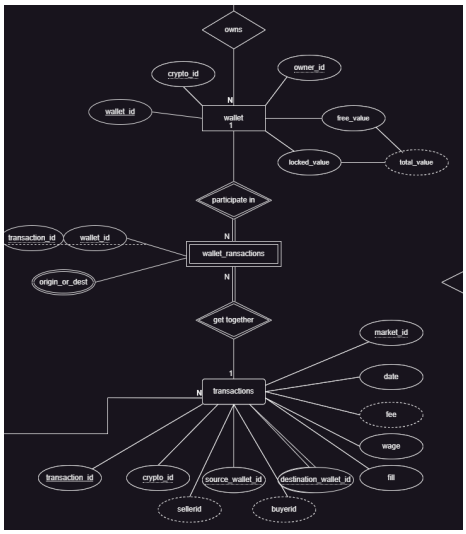
تبدیل نمودارهای فاز اول، به نمودارهای منطبق با SQL

۶.۱ چهار تغیر در نمودار برای بهینه سازی

در این بخش نمودار خود را تغیر دادیم تا مناسب درست کردن SQL باشد.

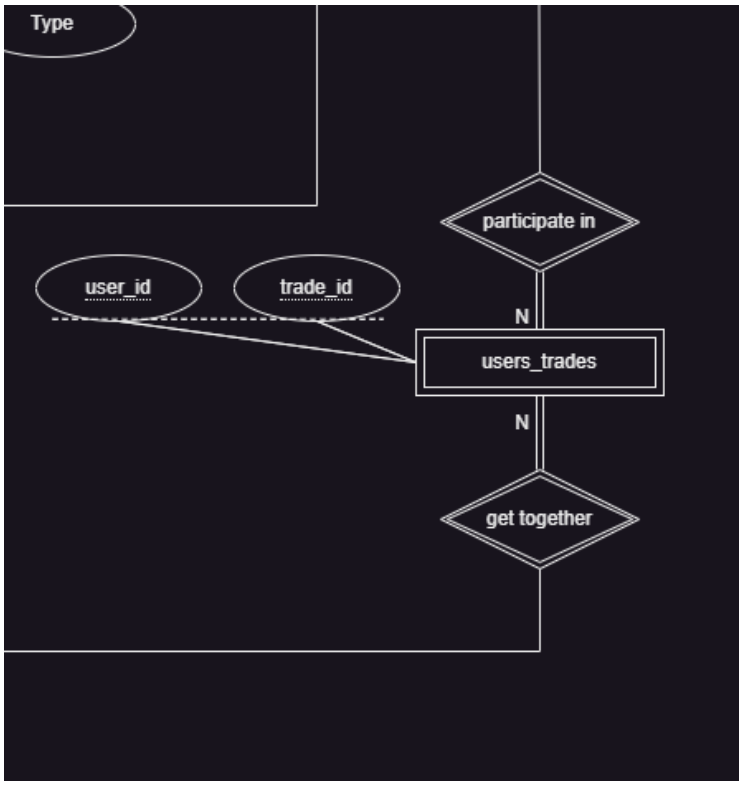
۶.۱.۱ رابطه ی چند به چند بین کیف پول و تراکنش‌ها

از انحایی که در هر تراکنش دو کیف پول استفاده میشد و هر کیف پول در چندین تراکنش شرکت میکرد، یک جدول جدید اضافه کردیم که رابطه ی چند به چند را به دو رابطه ی یک به چند تقسیم کند.



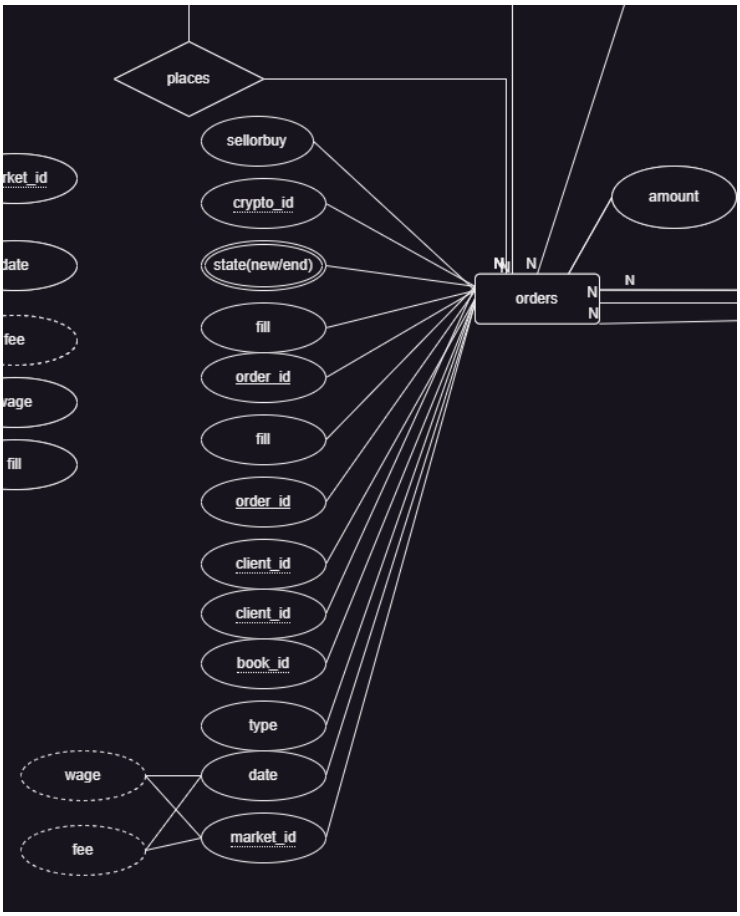
۶.۱.۲ رابطه ی چند به چند بین کاربرها و تبادل ها

از انجایی که هر تبادل از دو کاربر و هر کاربر در چندین تبادل شرکت میکرد، یک جدول جدول جدید اضافه کردیم که رابطه ی چند به چند را به دو رابطه ی یک به چند تقسیم بکند.



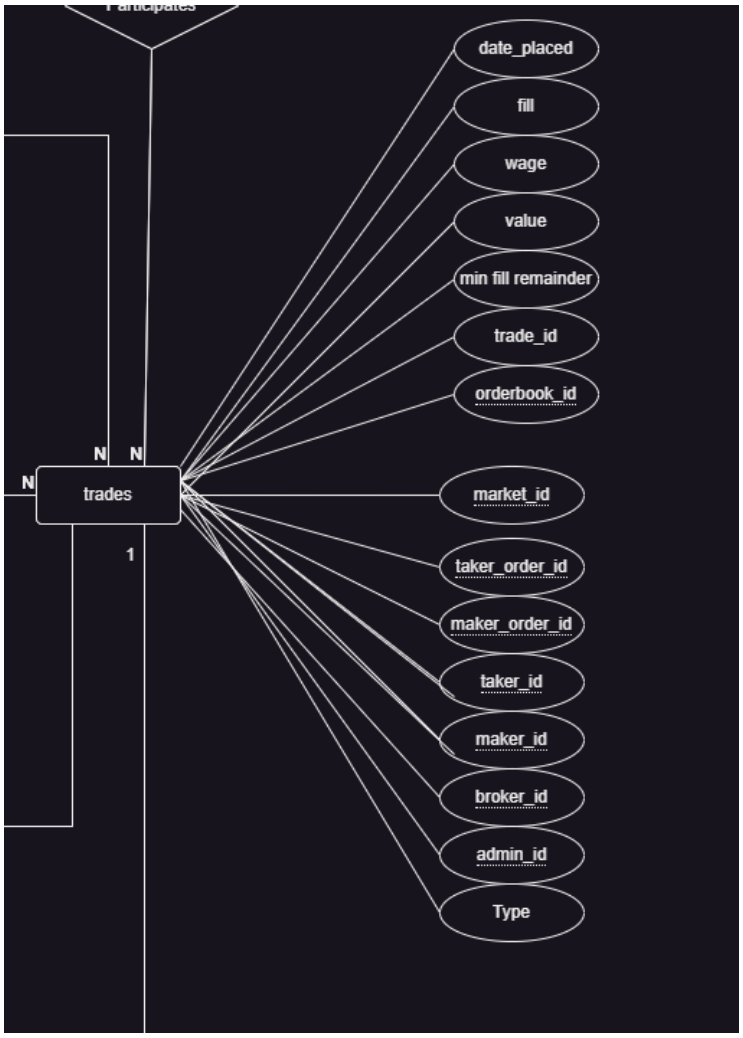
۶.۱.۳ رابطه ی specification در سفارشات

در فاز قبلی سفارشات به دو دسته‌ی سفارشت خرید و سفارشات فروش تقسیم می‌شدند که ما در این فاز این دو رابطه را در یک جدول سفارشات از طریق ستونی به نام type تفکیک کرده ایم.



۶.۱.۴ رابطه ی جنرالیزیشن در تبادل‌ها

در فاز قبلی تبادل‌ها شامل دو نوع $\mathrm{p2p}$ و otc می‌شدند که ما در این فاز آن دو نوع تبادل را در یک جدول تبادل قرار دادیم که با ستون type از هم تفکیک می شوند.



فصل ۷

ساخت پایگاه داده

بر اساس نمودار بخش قبل دو فایل SQL قرار دادیم که در یکی دستورات ایجاد جدول‌ها و دیگری تست کیس برای هر جدول ایجاد شده و در مسیر Phase2/SQL Files قرار داده شده است.

فصل ۸

بهبود پایگاه داده

۸.۱ نرمال‌تر سازی

در این بخش ما تمام جداول پایگاه داده‌ی خود را به فرم نرمال در اوردیم و تغیرات نمودار و دستورات ایجاد پایگاه داده را در دو فایل normalized_sqlform_Integrated.drawio **و** Normalized Create Tables.sql **در مسیر** Phase2/Normalize Files **قرار دادیم.**

۸.۱.۱ حذف crypto_id از transactions

$$transactions(\underline{transaction\_id}, crypto\_id, source\_wallet\_id,$$
$$destination\_wallet\_id, fill, wage, date, market\_id)$$
$$F.D = \{transaction\_id \rightarrow all\ attributes, market\_id \rightarrow crypto\_id\}$$

از انجایی که در $market\_id \rightarrow crypto\_id$ **یک** non prime attribute **به یک** non prime attribute **دیگر اشاره کرده این دیپندنسی را باید در یک جدول دیگر قرار دهیم تا از دومین فرم نرمال به سومین فرم نرمال انتقال پیدا کنیم.**

$$transactions(\underline{transaction\_id}, source\_wallet\_id, destination\_wallet\_id, fill, wage, date, market\_id)$$
$$F.D = \{transaction\_id \rightarrow all\ attributes\}$$

$$R(\underline{market\_id}, crypto\_id)$$
$$F.D = \{market\_id \rightarrow crypto\_id\}$$

حال دو جدول ما دارای سومین فرم نرمال هستند که همانطور که میبینید جدول R **زیر مجموعه ای از جدول** Markets **در پایگاه داده اصلی میباشد و نیازی به ساختن جدول اضافه نیست.**

Anomaly: چون ارتباط $market\_id \rightarrow crypto\_id$ **در جدول فروشگاه هم وجود داشت، برای تغیر دادن ویا حذف کرد یک رمزارز از یک فروشگاه مجبور بودیم که دو جدول را تغیر دَهیم و ممکن بود اطلاعات اشتباه تغیر دهیم ولی در حالت جدید این رابطه فقط در یک جدول وجود دارد .**

۸.۱.۲ حذف sales__lists__id , purchase__lists__id از orders

$$orders(\underline{order\_id}, sales\_lists\_id, purchase\_lists\_id, is\_sell, state, fill, client\_id,$$
$$date, market\_id, amount)$$
$$F.D = \{order\_id \to all\ attributes, (market\_id, is\_sell) \to (sales\_lists\_id, purchase\_lists\_id)\}$$

از انجایی که در $(market\_id, is\_sell) \to (sales\_lists\_id, purchase\_lists\_id)$ **دو** non prime attribute **به دو** non prime attribute **دیگر اشاره می‌کند، این دیپندنسی باعث می‌شود که جدول ما سومین فرم نرمال نباشد و ما باید ان را به یک جدول دیگر انتقال بدهیم.**
ما می‌توانیم $(market\_id, is\_sell) \to (sales\_lists\_id, purchase\_lists\_id)$
به دو رابطه‌ی $(market\_id, is\_sell) \to sales\_lists\_id$ **و** $(market\_id, is\_sell) \to purchase\_lists\_id$ **بشکانیم و با فرض اینکه** is__sell **یک** boolean **با دو مقدار می باشد، دو جدول برای روابط بالا بسازیم**

$$orders(\underline{order\_id}, is\_sell, state, fill, client\_id, date, market\_id, amount)$$
$$F.D = \{order\_id \to all\ attributes\}$$

$$is\_sell == TRUE : sale\_table(\underline{market\_id}, sales\_lists\_id)$$
$$F.D = \{market\_id \to sales\_lists\_id\}$$

$$is\_sell == FALSE : purchase\_table(\underline{market\_id}, purchase\_lists\_id)$$
$$F.D = \{market\_id \to purchase\_lists\_id\}$$

همانطور که می دانیم دو جدول جدید ایجاد شده هر دو زیر مجموعه‌ای از جدول Orderbooks **می‌باشند و ما فقط باید دو ستون از جدول سفارشات حذف کنیم.**

Anomaly: چون ارتباط $market\_id \to *\_list\_id$ **در جدول دفتر سفارشات هم وجود داشت، برای تغیر دادن ویا حذف کرد یک دفتر شفارش از یک سفارش مجبور بودیم که دو جدول را تغیر دهیم و ممکن بود اطلاعات اشتباه تغیر دهیم ولی در حالت جدید این رابطه فقط در یک جدول وجود دارد .**

۸.۱.۳ نرمال کردن جدول trades

$trades(\underline{trade\_id}, date\_placed, fill, wage, value, min\_fill\_remainder, type, market\_id,$
$taker\_order\_id, maker\_order\_id, taker\_id, maker\_id, broker\_id, admin\_id)$
$F.D = \{trade\_id \to all\, attribute, taker\_order\_id \to taker\_id, maker\_order\_id \to$
$maker\_id,$
$type \to (maker\_order\_id, admin\_id, min\_fill\_remainder)\}$
$maker\_order\_id \to$ و $taker\_order\_id \to taker\_id$ در مرحله‌ی اول ما دو دیپندسی
$maker\_id$ باعث می شوند جدول ما در سومین فرم نرمال قرار نگیرد به دلیل مثال قبلی باید در جداولی دیگر قرار می دهیم که این دو جدول هر دو زیر مجموعه از جدول سفارشات هستند که ما نیازی به ساختن آن‌ها نداریم.

$trades(\underline{trade\_id}, date\_placed, fill, wage, value, min\_fill\_remainder,$
$type, market\_id,$
$taker\_order\_id, maker\_order\_id, broker\_id, admin\_id)$
$F.D = \{trade\_id \to all\, attribute, type \to (maker\_order\_id,$
$admin\_id, min\_fill\_remainder)\}$

رابطه‌ی $type \to (maker\_order\_id, admin\_id)$ یک دیپندنسی کامل نیست و به طور مثال ما اگر بدانیم که $\text{type} == \text{'otc'}$ هست می توانیم نتیجه بگیرم که $\text{maker\_order\_id} == \text{NULL}$ می‌باشد و ما پس از مشورت با هد پروژه تصمیم گرفتیم که جدول را برا اساس type به دو جدول p2p و otc تقسیم بکنیم.

$p2p(\underline{p2p\_id}, date\_placed, fill, wage, value, min\_fill\_remainder, market\_id,$

$taker\_order\_id, maker\_order\_id)$

$$F.D = \{p2p\_id \to all\, attribute\}$$

$otc(\underline{otc\_id}, date\_placed, fill, wage, value, taker\_id, broker\_id, admin\_id)$

$$F.D = \{otc\_id \to all\, attribute\}$$

Anomaly: در مرحله‌ی اول نرمال سازی همان مشکلات دو تا نرمال سازی قبلی (تغیر دادن و حذف کردن در دو جدول) را برطرف کردیم و در مرحله‌ی دوم اگر جدول تبادلات را به دو جدول کوچکتر تبدیل نمی‌کردیم ان وقت یک جدول بزرگ با حجم زیاد داشتیم که مقدار بعضی از ستون ها در بعضی از سطر ها NULL می شد که هم حافظه‌ی زیادی مصرف میکرد و هم کوئری زدن رو کندتر می‌کرد.

۸.۲ indexها

برای سه جدول خود index قرار دادیم تا در جستار ها به ما کمک بکند.

۸.۲.۱ ایندکس تاریخ بر روی p2p

از انحایی که ما نیاز داریم تا قیمت لحظه‌ای هر بازار را بر اساس اخرین تبادل ثبت شده حساب کنیم. مرتب کردن این جدول بر اساس تاریخ به محاسبه‌ی قیمت بازار بسیار کمک می‌کند.

```
CREATE INDEX IF NOT EXISTS date_placed_idx
    ON p2p(date_placed);
```

۸.۲.۲ ایندکس کیف پول بر روی تراکنش‌ها

از انحایی که ما نیازمند محاسبه‌ی موجودی کیف پول هر کاربر هستیم پس بهتر است تراکنش های هر کیف پول را مرتب و درکنار هم در جدول تبادل ها قرار دهیم.

```
CREATE INDEX IF NOT EXISTS walle_id_idx
    ON transactions(source_wallet_id);
```

۸.۲.۳ ایندکس قیمت بر روی سفارش‌ها

از انحایی که درهر تبادل که ساخته می‌شود کم قیمت‌ترین سفارش فروش مورد استفاده قرار می‌گیرد بهتر است که بر اساس قیمت سفارشات خود را مرتب کنیم تا در هر لحظه دسترسی سریعی به کم قیمت‌ترین پیشنهاد فروش داشته باشیم.

```
CREATE INDEX IF NOT EXISTS fill_idx ON orders(fill);
```

۸.۳ جستارهای پایگاه داده

۸.۳.۱ جستار اول

به صورت زیر بود:

```
WITH LatestTransactions AS (
    SELECT
        market_id,
        MAX(date_placed) AS latest_date
    FROM
        p2p
    GROUP BY
        market_id
)
```

```
SELECT
    p.market_id, p.date_placed,
    p.fill * o.amount AS price
FROM
    p2p p
JOIN
    orders o ON p.maker_order_id = o.order_id
JOIN
    LatestTransactions lt
    ON p.market_id = lt.market_id
    AND p.date_placed = lt.latest_date;
```

نمونه خروجی:

	market_id integer	date_placed timestamp without time zone	price bigint
1	1	2023-11-28 00:00:00	57600
2	2	2023-11-27 00:00:00	52900
3	6	2023-11-26 00:00:00	48400
4	3	2023-11-25 00:00:00	44100
5	4	2023-11-24 00:00:00	40000
6	5	2023-11-17 00:00:00	16900
7	7	2023-11-13 00:00:00	8100

۸.۳.۲ جستار دوم

به صورت زیر بود:

```
WITH LatestTransactions AS (
    SELECT
        market_id,
        MAX(date_placed) AS latest_date
    FROM
        p2p
    GROUP BY
        market_id
),
MarketPrices AS (
```

```
    SELECT
        p.market_id,
        p.fill * o.amount AS price
    FROM
        p2p p
    JOIN
        orders o ON p.maker_order_id = o.order_id
    JOIN
        LatestTransactions lt ON p.market_id = lt.market_id AND p.date_placed = lt.latest_date
),
SupplyDemand AS (
    SELECT
        m.market_id,
        c.name AS crypto_name,
        SUM(CASE WHEN o.is_sell THEN o.amount ELSE 0 END) AS total_supply,
        SUM(CASE WHEN NOT o.is_sell THEN o.amount ELSE 0 END) AS total_demand
    FROM
        markets m
    JOIN
        orders o ON m.market_id = o.market_id
    JOIN
        cryptocurrencies c ON m.crypto_id = c.crypto_id
    GROUP BY
        m.market_id, c.name
)
SELECT
    sd.crypto_name,
    sd.total_supply,
    sd.total_demand,
    mp.price
FROM
    SupplyDemand sd
JOIN
    MarketPrices mp ON sd.market_id = mp.market_id;
;
```

نمونه خروجی:

	crypto_name text	total_supply numeric	total_demand numeric	price bigint
1	BNB	130	130	16900
2	Tether	350	350	44100
3	Shiba	320	320	48400
4	Doge	380	380	52900
5	Ethereum	200	200	40000
6	rial	1530	1530	57600
7	notcoin	90	90	8100

۸.۳.۳ جستار سوم

به صورت زیر بود:

```
WITH LatestPrices AS (
    SELECT
        m.market_id,
        m.crypto_id,
        (p.fill * o.amount) / NULLIF(o.amount, 0) AS latest_price,
        ROW_NUMBER() OVER (PARTITION BY m.crypto_id ORDER BY p.date_placed DESC) AS rn
    FROM
        p2p p
    JOIN
        orders o ON p.maker_order_id = o.order_id
    JOIN
        markets m ON p.market_id = m.market_id
),
FilteredPrices AS (
    SELECT
        market_id,
        crypto_id,
        latest_price
    FROM
        LatestPrices
    WHERE
        rn = 1
),
UserWalletValues AS (
    SELECT
        w.owner_id,
        w.crypto_id,
        (w.free_value + w.locked_value) * fp.latest_price AS total_value
    FROM
        wallets w
    JOIN
        FilteredPrices fp ON w.crypto_id = fp.crypto_id
)
SELECT
    u.user_id,
    u.first_name,
    u.last_name,
    COALESCE(SUM(uwv.total_value), 0) AS total_money_in_rials
FROM
    users u
LEFT JOIN
    UserWalletValues uwv ON u.user_id = uwv.owner_id
GROUP BY
    u.user_id, u.first_name, u.last_name;
```

۸.۳.۴ جستار چهارم

به صورت زیر بود:

```
WITH TopBuyOffers AS (
    SELECT
```

	crypto_name text	total_supply numeric	total_demand numeric	price bigint
1	BNB	130	130	16900
2	Tether	350	350	44100
3	Shiba	320	320	48400
4	Doge	380	380	52900
5	Ethereum	200	200	40000
6	rial	1530	1530	57600
7	notcoin	90	90	8100

```
        order_id,
        is_sell,
        state,
        fill,
        client_id,
        date,
        market_id,
        amount,
        ROW_NUMBER() OVER (PARTITION BY market_id ORDER BY amount DESC) AS rank
    FROM
        orders
    WHERE
        is_sell = FALSE
),
TopSellOffers AS (
    SELECT
        order_id,
        is_sell,
        state,
        fill,
        client_id,
        date,
        market_id,
        amount,
        ROW_NUMBER() OVER (PARTITION BY market_id ORDER BY amount ASC) AS rank
    FROM
        orders
    WHERE
        is_sell = TRUE
)
SELECT crypto_id, x.* FROM (
SELECT
    'Buy' AS offer_type,
    order_id,
    is_sell,
    state,
    fill,
    client_id,
    date,
    market_id,
    amount
FROM
    TopBuyOffers
```

```
WHERE
    rank <= 10
UNION ALL
SELECT
    'Sell' AS offer_type,
    order_id,
    is_sell,
    state,
    fill,
    client_id,
    date,
    market_id,
    amount
FROM
    TopSellOffers
WHERE
    rank <= 10
ORDER BY
    market_id, offer_type, amount DESC) x
JOIN markets ON markets.market_id = x.market_id;
```

نمونه خروجی:

	crypto_id integer	offer_type text	order_id integer	is_sell boolean	state text	fill bigint	client_id integer	date timestamp without time zone	market_id integer	amount bigint
1	1	Buy	9	false	end	2	3	2020-06-07 00:00:00	1	240
2	1	Buy	14	false	end	6	5	2021-06-07 00:00:00	1	190
3	1	Buy	15	false	fill	8	5	2021-07-07 00:00:00	1	180
4	1	Buy	16	false	fill	10	3	2023-06-07 00:00:00	1	170
5	1	Buy	25	false	end	5	1	2020-06-05 00:00:00	1	160
6	1	Buy	29	false	end	11	4	2020-06-05 00:00:00	1	120
7	1	Buy	30	false	end	10	5	2021-06-05 00:00:00	1	110
8	1	Buy	41	false	end	3	3	2020-06-03 00:00:00	1	80
9	1	Buy	42	false	end	4	3	2021-06-03 00:00:00	1	70
10	1	Buy	43	false	end	5	4	2021-07-03 00:00:00	1	60
11	1	Sell	21	true	end	2	7	2020-06-06 00:00:00	1	120
12	1	Sell	22	true	end	6	2	2021-06-06 00:00:00	1	110
13	1	Sell	33	true	end	7	4	2020-06-04 00:00:00	1	80
14	1	Sell	34	true	end	6	6	2021-06-04 00:00:00	1	70
15	1	Sell	35	true	end	4	6	2021-07-04 00:00:00	1	60
16	1	Sell	36	true	end	3	3	2023-06-04 00:00:00	1	50
17	1	Sell	37	true	end	2	2	2020-06-04 00:00:00	1	40
18	1	Sell	38	true	end	1	6	2021-06-04 00:00:00	1	30
19	1	Sell	39	true	fill	1	3	2021-07-04 00:00:00	1	20
20	1	Sell	40	true	fill	2	2	2023-06-04 00:00:00	1	10

۸.۳.۵ جستار پنجم

به صورت زیر بود:

```
DROP FUNCTION get_best_sell_orders(bigint,integer);

CREATE OR REPLACE FUNCTION get_best_sell_orders(target_price BIGINT, market_id_param INT)
RETURNS TABLE (
    order_id INT,
    fill BIGINT,
```

```
    amount BIGINT,
    price BIGINT,
    market_id INT,
    date TIMESTAMP
) AS $$
BEGIN
    RETURN QUERY
    WITH RankedOrders AS (
        SELECT
            o.order_id,
            o.fill,
            o.amount,
            o.market_id,
            o.date,
            ROW_NUMBER() OVER (ORDER BY o.fill ASC) AS rn,
            SUM(o.fill * o.amount) OVER (ORDER BY o.fill ASC) AS cumulative_price
        FROM
            orders o
        WHERE
            o.is_sell = TRUE
            AND o.market_id = market_id_param
    ),
    OrdersWithPrev AS (
        SELECT
            ro.*,
            LAG(ro.cumulative_price) OVER (ORDER BY ro.rn) AS prev_cumulative_price
        FROM
            RankedOrders ro
    ),
    SelectedOrders AS (
        SELECT
            ow.order_id,
            ow.fill,
            ow.amount,
            ow.market_id,
            ow.date,
            ow.rn,
            ow.cumulative_price,
            ow.prev_cumulative_price
        FROM
            OrdersWithPrev ow
        WHERE
            ow.cumulative_price <= target_price
            OR (ow.cumulative_price > target_price AND ow.prev_cumulative_price <= target_price)
    )
    SELECT
        so.order_id,
        so.fill,
        so.amount,
        (so.fill * so.amount) AS price,
        so.market_id,
        so.date
    FROM
        SelectedOrders so
    ORDER BY so.cumulative_price;
END;
$$ LANGUAGE plpgsql;
```

```
SELECT * FROM get_best_sell_orders(2000,1);
```

	order_id integer	fill bigint	amount bigint	price bigint	market_id integer	date timestamp without time zone
1	38	1	30	30	1	2021-06-04 00:00:00
2	39	1	20	20	1	2021-07-04 00:00:00
3	17	2	160	320	1	2020-06-06 00:00:00
4	21	2	120	240	1	2020-06-06 00:00:00
5	37	2	40	80	1	2020-06-04 00:00:00
6	40	2	10	20	1	2023-06-04 00:00:00
7	36	3	50	150	1	2023-06-04 00:00:00
8	35	4	60	240	1	2021-07-04 00:00:00
9	22	6	110	660	1	2021-06-06 00:00:00

۸.۳.۶ جستار ششم

به صورت زیر بود:

```
DROP FUNCTION get_best_volume_sell_orders(bigint,integer);

CREATE OR REPLACE FUNCTION get_best_volume_sell_orders(target_volume BIGINT, market_id_param INT)
RETURNS TABLE (
    order_id INT,
    fill BIGINT,
    amount BIGINT,
    price BIGINT,
    market_id INT,
    date TIMESTAMP
) AS $$
BEGIN
    RETURN QUERY
    WITH RankedOrders AS (
        SELECT
            o.order_id,
            o.fill,
            o.amount,
            o.market_id,
            o.date,
            (o.fill * o.amount) AS price,
            ROW_NUMBER() OVER (ORDER BY o.amount ASC) AS rn,
            SUM(o.amount) OVER (ORDER BY o.amount ASC) AS cumulative_amount
        FROM
            orders o
        WHERE
            o.is_sell = TRUE
            AND o.market_id = market_id_param
ORDER BY o.fill ASC
    ),
    FinalOrders AS (
        SELECT
            ro.order_id,
            ro.fill,
```

```
            ro.amount,
            ro.price,
            ro.market_id,
            ro.date,
            ro.cumulative_amount,
            CASE
                WHEN ro.cumulative_amount >= target_volume THEN 'sufficient'
                ELSE 'insufficient'
            END AS status
        FROM
            RankedOrders ro
    ),
    SufficientOrders AS (
        SELECT
            fo.*
        FROM
            FinalOrders fo
        WHERE
            fo.status = 'sufficient'
        ORDER BY
            fo.fill, fo.cumulative_amount
        LIMIT 1
    )
    SELECT
        fo.order_id,
        fo.fill,
        fo.amount,
        fo.price,
        fo.market_id,
        fo.date
    FROM
        FinalOrders fo
    WHERE
        fo.status = 'insufficient'
    UNION ALL
    SELECT
        so.order_id,
        so.fill,
        so.amount,
        so.price,
        so.market_id,
        so.date
    FROM
        SufficientOrders so;
END;
$$ LANGUAGE plpgsql;


SELECT * FROM get_best_volume_sell_orders(110,1);
```

نمونه خروجی:

	order_id integer	fill bigint	amount bigint	price bigint	market_id integer	date timestamp without time zone
1	38	1	30	30	1	2021-06-04 00:00:00
2	39	1	20	20	1	2021-07-04 00:00:00
3	37	2	40	80	1	2020-06-04 00:00:00
4	40	2	10	20	1	2023-06-04 00:00:00
5	21	2	120	240	1	2020-06-06 00:00:00

۸.۳.۷ جستار هفتم

به صورت زیر بود:

```
DROP FUNCTION IF EXISTS compare_p2p_and_otc(bigint, integer, numeric);

CREATE OR REPLACE FUNCTION compare_p2p_and_otc(target_volume BIGINT, market_id_param INT, otc_price NUMERIC
RETURNS TABLE (
    source TEXT,
    average_price NUMERIC
) AS $$
DECLARE
    total_price NUMERIC;
    final_adjustment BIGINT;
BEGIN
    -- Retrieve and adjust the order data
    WITH OrderData AS (
        SELECT
            gt.amount,
            gt.fill,
            SUM(gt.amount) OVER (ORDER BY gt.fill ASC) AS cumulative_volume,
            ROW_NUMBER() OVER (ORDER BY gt.fill ASC) AS rn
        FROM get_best_volume_sell_orders(target_volume, market_id_param) gt
    ),
    AdjustedData AS (
        SELECT
            od.amount,
            od.fill,
            CASE
                WHEN od.cumulative_volume > target_volume THEN od.amount - (od.cumulative_volume - target_v
                ELSE od.amount
            END AS adjusted_amount,
            od.rn
        FROM OrderData od
    ),
    FinalData AS (
        SELECT
            ad.amount,
            ad.fill,
            ad.adjusted_amount,
            (ad.fill * ad.adjusted_amount) AS weighted_price
        FROM AdjustedData ad
    ),
    Adjustment AS (
        SELECT
            amount,
            fill,
            adjusted_amount,
            weighted_price
        FROM FinalData
```

```
        UNION ALL
        SELECT
            0 AS amount,
            MAX(fill) AS fill,
            -(SUM(adjusted_amount) - target_volume) AS adjusted_amount,
            (MAX(fill) * -(SUM(adjusted_amount) - target_volume)) AS weighted_price
        FROM FinalData
    )
    SELECT
        SUM(a.weighted_price)::NUMERIC / target_volume AS average_price
    INTO total_price
    FROM Adjustment a;

    -- Compare the weighted average price with OTC price
    IF total_price IS NULL OR total_price > otc_price THEN
        RETURN QUERY SELECT 'OTC' AS source, otc_price AS average_price;
    ELSE
        RETURN QUERY SELECT 'P2P' AS source, total_price AS average_price;
    END IF;
END;
$$ LANGUAGE plpgsql;



-- Call the function with your desired parameters
--the third parameter is the price for over the
--counter deal asked from the admin in
-- the moment of the sale.

SELECT * FROM compare_p2p_and_otc(110, 1, 1.75);
```

نمونه خروجی:

	source text	average_price numeric
1	P2P	1.545454545454