



دانشگاه صنعتی شریف

پروژهی صرافی ارز دیجیتال: پایگاه داده - فاز اول

استاد: دکتر مهدی آخی

شماره تیم: ۱۴

محمد جعفری پور
۴۰۱۱۰۵۷۹۷

محمد امین حیدری
۴۰۱۱۷۰۵۵۳

مانی ابراهیمی
۴۰۱۱۷۰۴۹۱

بهار ۱۴۰۳

فهرست مطالب

۵	۱	کلیت فاز اول پروژه
۵	۱.۱	شرح
۵	۱.۲	تقسیم وظایف
۷	۲	دیاگرام های ER
۷	۲.۱	شرح
۷	۲.۲	توضیح هر موجودیت
۷	۲.۲.۱	User
۷	۲.۲.۲	Wallet
۷	۲.۲.۳	Transactions
۸	۲.۲.۴	Orders
۸	۲.۲.۵	Trades
۸	۲.۲.۶	OrderBooks
۸	۲.۲.۷	Markets
۸	۲.۲.۸	Brokers
۹	۲.۲.۹	CryptoCurrency
۹	۲.۲.۱۰	Network
۹	۲.۲.۱۱	Online Payments
۹	۲.۲.۱۲	Wallet History
۹	۲.۲.۱۳	Crypto Histories
۱۱	۳	سوالات جبر رابطه ای
۱۱	۳.۱	شرح
۱۱	۳.۲	پاسخ به سوالات
۱۱	۳.۲.۱	سوال ۱
۱۱	۳.۲.۲	سوال ۲
۱۱	۳.۲.۳	سوال ۳
۱۱	۳.۲.۴	سوال ۴
۱۲	۳.۲.۵	سوال ۵
۱۲	۳.۲.۶	سوال ۶
۱۲	۳.۲.۷	سوال ۷
۱۲	۳.۲.۸	سوال ۸
۱۲	۳.۲.۹	سوال ۹
۱۳	۳.۲.۱۰	سوال ۱۰

۱۵	۴	ضمیمه: تصویر دیاگرام ER
۱۷	۵	کلیت فاز دوم پروژه
۱۷	۵.۱	شرح
۱۹	۶	تبدیل نمودارهای فاز اول، به نمودارهای منطبق با SQL
۱۹	۶.۱	چهار تغییر در نمودار برای بهینه سازی
۲۰	۶.۱.۱	رابطه ی چند به چند بین کیف پول و تراکنش ها
۲۱	۶.۱.۲	رابطه ی چند به چند بین کاربرها و تبادل ها
۲۲	۶.۱.۳	رابطه ی specification در سفارشات
۲۳	۶.۱.۴	رابطه ی جنرالیزیشن در تبادل ها
۲۵	۷	ساخت پایگاه داده
۲۷	۸	بهبود پایگاه داده
۲۷	۸.۱	نرمال تر سازی
۲۷	۸.۱.۱	حذف crypto_id از transactions
۲۸	۸.۱.۲	حذف orders_id , purchase_lists_id , sales_lists_id
۲۹	۸.۱.۳	نرمال کردن جدول trades
۲۹	۸.۲	index ها
۳۰	۸.۲.۱	ایندکس تاریخ بر روی p2p
۳۰	۸.۲.۲	ایندکس کیف پول بر روی تراکنش ها
۳۰	۸.۲.۳	ایندکس قیمت بر روی سفارش ها
۳۰	۸.۳	جستارهای پایگاه داده
۳۰	۸.۳.۱	جستار اول
۳۱	۸.۳.۲	جستار دوم
۳۲	۸.۳.۳	جستار سوم
۳۳	۸.۳.۴	جستار چهارم
۳۴	۸.۳.۵	جستار پنجم
۳۵	۸.۳.۶	جستار ششم
۳۷	۸.۳.۷	جستار هفتم

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^۱ 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^۲ قابل مشاهده است.

^۱ در صورتی که لینک برای شما کار نمی‌کند، از آدرس <https://github.com/maniebra/dbms-exchange-project> استفاده نمایید.

^۲ در صورتی که این لینک برای شما کار نمی‌کند، می‌توانید از آدرس https://docs.google.com/spreadsheets/d/1x1Guh4HTWLyG9GTomZEsp5cjIGez9m9Day3bS_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:

User ۲.۲.۱

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.

Wallet ۲.۲.۲

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

Transactions ۲.۲.۳

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 ارزش 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 ارزش پایه‌ی آن مارکت و قیمت لحظه‌ای آن ارزش پایه به ریال محاسبه می‌شود.

Orders ۲.۲.۴

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.

Trades ۲.۲.۵

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 P2P را تشکیل داده اند. همچنین این موجودیت دارای یک شناسه برای هر تبادل می باشد. صفت min_fill_remainder به این صورت عمل می کند که حجم باقی مانده ی کمینه ی دو سفارش خرید و فروش را ذخیره می کند.

OTC

شامل ID ادمین و مشتری می باشد که بوسیله ی شناسه ی Market به بازار مربوطه متصل شده است.

P2P

شامل دو ID و OrderID خریدار و فروشنده یا همان maker و taker می باشند که به وسیله ی شناسه ی صرافی یا همان Broker_ID به صرافی مربوطه متصل شده اند.

OrderBooks ۲.۲.۶

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 زیرمجموعه ی 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.

Markets ۲.۲.۷

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.

Brokers ۲.۲.۸

موجودیت صرافی ها یا همان Brokers، که دارای صفات گفته شده از جمله شناسه و سایر موارد است. این موجودیت برای ذخیره اطلاعات 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 یا CryptoCurrency ارز هایی اند که در سایت وجود دارند و توسط افراد مبادله میشوند. این ارز ها ممکن است قیمت ثابت Stable coin باشند و یا قیمت آنها هر لحظه عوض شود nonstable Currency.

۲.۲.۱۰ Network

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

۲.۲.۱۱ Online Payments

تاریخچه ی تمامی واریزی های هر کاربر، مقدار آن و زمان انجام شده است.

۲.۲.۱۲ Wallet History

تاریخچه ای از تغییرات میزان هر کیف پول است و هر تراکنشی برای دو کیف پول یک Wallet History جدید می سازد.

۲.۲.۱۳ Crypto Histories

تاریخچه ی تغییرات قیمت یک رمزارز است که زمان آن تغییر و مقدار و قیمت آن در آن زمان (قیمت همان قیمت لحظه ای مارکت است) نشان می دهد. با انجام هر تراکنش یک CryptoHistory جدید ایجاد می شود چرا که قیمت لحظه ای ارز تغییر می کند.

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

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

۳.۲.۲ سوال ۲

$$\text{owner_id} \mathcal{F}_{\text{Sum}(\text{total_value} \times \text{in_time_price})} [Wallets \bowtie_{\text{Market.market_id}=id} Markets]$$

۳.۲.۳ سوال ۳

$$\text{crypto_id} \mathcal{F}_{\text{Count}(\text{order_id})} [\sigma_{\text{fill}=\text{"false"}} (Orders)]$$

۳.۲.۴ سوال ۴

$$A = \rho_{\text{user_id}, \text{total}} [\text{owner_id} \mathcal{F}_{\text{Sum}(\text{fee})} \text{as totalSell} (\text{Transactions} \bowtie_{\text{Transactions.origin_wallet_id}=\text{wallets.id}} \text{Wallet})]$$

$$B = \rho_{\text{user_id}, \text{total}} [\text{owner_id} \mathcal{F}_{\text{Sum}(\text{fee})} \text{as totalBuy} (\text{Transactions} \bowtie_{\text{Transactions.dest_wallet_id}=\text{wallets.id}} \text{Wallet})]$$

$$\text{user_id} \mathcal{F}_{\text{Sum}(\text{Total})}$$

Relational Algebra^۱

سوال ۳.۲.۵

$$A =_{\text{user_id, cryptoid}} \mathcal{F}_{\text{Count}}(\text{Transactions.id}) ((\text{Users} \times \text{Cryptocurrency}) \bowtie_{\text{users.user_id}=\text{Transactions.SellerID}} \text{Transactions})$$

$$B =_{\text{user_id, cryptoid}} \mathcal{F}_{\text{Count}}(\text{Transactions.id}) ((\text{Users} \times \text{Cryptocurrency}) \bowtie_{\text{users.user_id}=\text{Transactions.BuyerID}} \text{Transactions})$$

$$_{\text{user_id, cryptoid}} \mathcal{F}_{\text{math}}(\text{Sum}(\text{TotalCount})) (\rho_{\text{user_id, cryptoid}/\text{TotalCount}}(A) \cup \rho_{\text{user_id, cryptoid}/\text{TotalCount}}(B))$$

سوال ۳.۲.۶

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

سوال ۳.۲.۷

$$\begin{aligned} A &= \Pi_{\text{cryptoid, in_time_price}}(\text{Cryptocurrency}) \\ B &=_{\text{cryptoid}} \mathcal{F}_{\text{math}}(\text{max}(\text{Date}) \text{ as Date}) (\sigma_{\text{Transactions.Date-Now}() \leq "0000-00-30-00:00:00"}(\text{Transactions} \times \text{Cryptocurrency})) \\ C &= \Pi_{\text{cryptoid, fee}}[\text{Cryptocurrency} \bowtie_{\text{Cryptocurrency.id}=\text{Transactions.cryptoid}} (\text{Transactions} \bowtie B)] \\ &\Pi_{\text{cryptoid, in_time_price-fee}}(A \bowtie C) \end{aligned}$$

سوال ۳.۲.۸

$$\begin{aligned} A &=_{\text{owner_id}} \mathcal{F}_{\text{Sum}}(\text{Total_value}) \text{ assum}(\text{Wallets}) \\ B &= \Pi_{\text{owner_id, cryptoid, Total_value}}(\text{Wallets}) \\ &\text{cryptoid} \mathcal{F}_{\text{count}}(\text{owner_id})[\sigma_{\text{percentage} \geq 0.05}(\rho_{\text{cryptoid, owner_id, percentage}}[\\ &\Pi_{\text{cryptoid, owner_id, } \frac{\text{Total_value}}{\text{Sum}}}(A \bowtie B)])] \end{aligned}$$

سوال ۳.۲.۹

$$\begin{aligned} A &= \Pi_{\text{user_id, Date}}(\sigma_{\text{Date-Now}() \leq "0000-00-30;00:00:00"}(\text{Online_Payments})) \\ B &= \rho_{\text{user_id, paymentDate}}(A) \bowtie \text{WalletHistories} \\ C &=_{\text{cryptoid, user_id, paymentDate}} \mathcal{F}_{\text{Max}}(\text{Date})(\sigma_{\text{Date} < \text{paymentDate}}(B)) \\ &\rho_{\text{user_id, paymentDate}}(A) \times \text{CryptoHistories} \\ E &=_{\text{cryptoid, user_id, paymentDate}} \mathcal{F}_{\text{max}}(\text{Date})(\sigma_{\text{Date} < \text{paymentDate}}(D)) \\ X &=_{\text{user_id, paymentDate}} \mathcal{F}_{\text{Sum}}(\text{amount} \times \text{price}) \text{ as totalValue}((C \times \text{WalletHistories}) \\ &\bowtie_{\text{user_id}=\text{user_id} \wedge \text{paymentDate}=\text{paymentDate}} E \bowtie \text{CryptoHistories}) \\ &\bowtie_{\text{user_id}=\text{user_id} \wedge \text{paymentDate}=\text{paymentDate}} \text{Online_Payments} \end{aligned}$$

$$\mathcal{F}_{\text{CountUnique}}(\text{user_id})(\sigma_{\text{onlineamount} \geq \frac{1}{5} \text{totalValue}}(X))$$

۳.۲.۱۰ سوال ۱۰

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

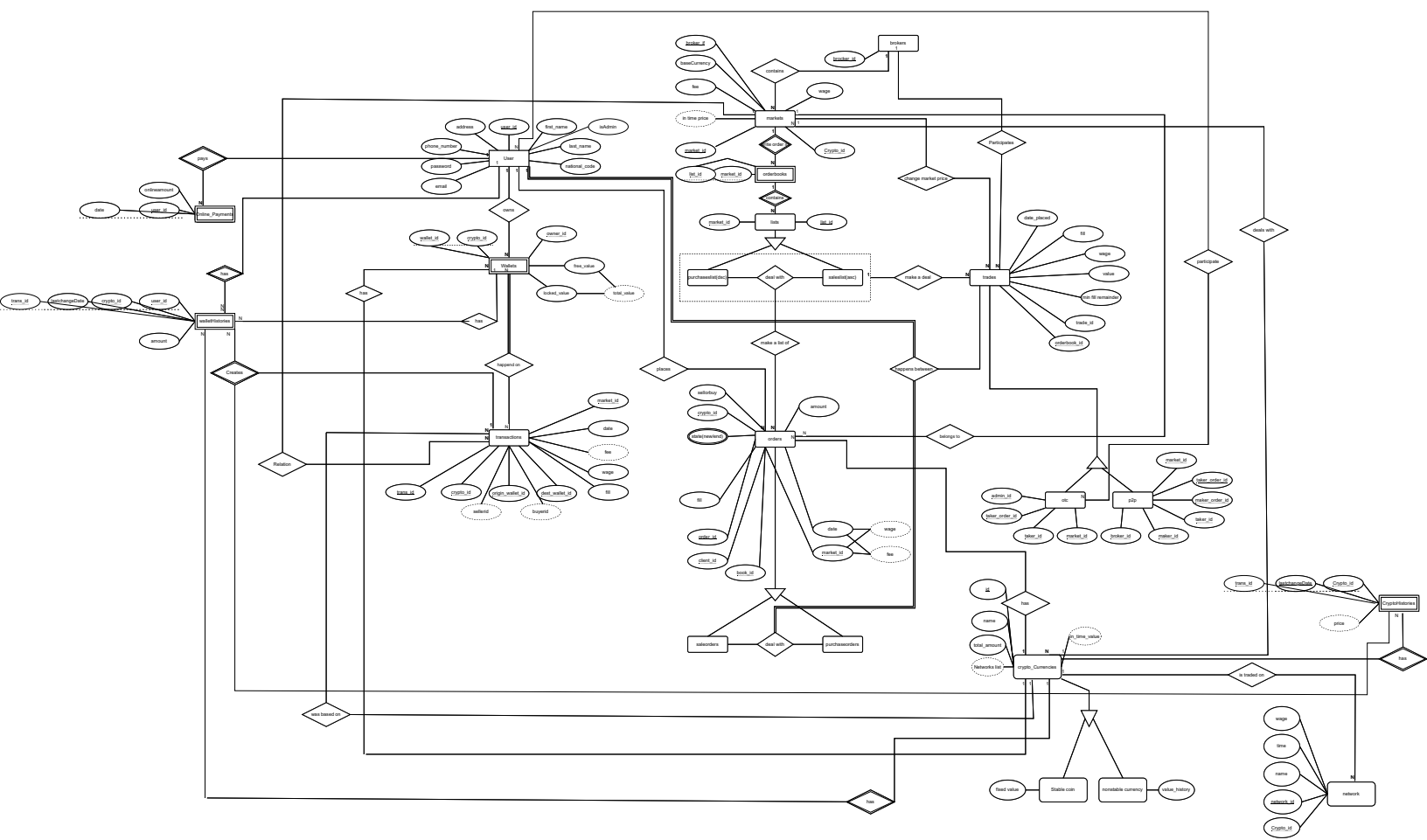
$$B = \rho_{cryptoid, price, totalSell} [cryptoid, price \mathcal{F}_{\text{Sum}(\text{amount})} ((Cryptocurrency \times prices) \bowtie_{Cryptocurrency.id=purchaseOrders.cryptoid} purchaseOrders)]$$

$$A \cup B$$

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 Chrome یا Edge باز نمایید.



فصل ۵

کلیت فاز دوم پروژه

۵.۱ شرح

ما در این فاز از پروژه چهار تغییر در نمودار فاز قبلی خود ایجاد کردیم که به ترتیب عبارتند از بهینه کردن نمودار برای طراحی دیتابیس، ایجاد دیتابیس، نرمال سازی و ایندکس کردن دیتابیس و در نهایت انجام هشت جستجو در دیتابیس.

فصل ۶

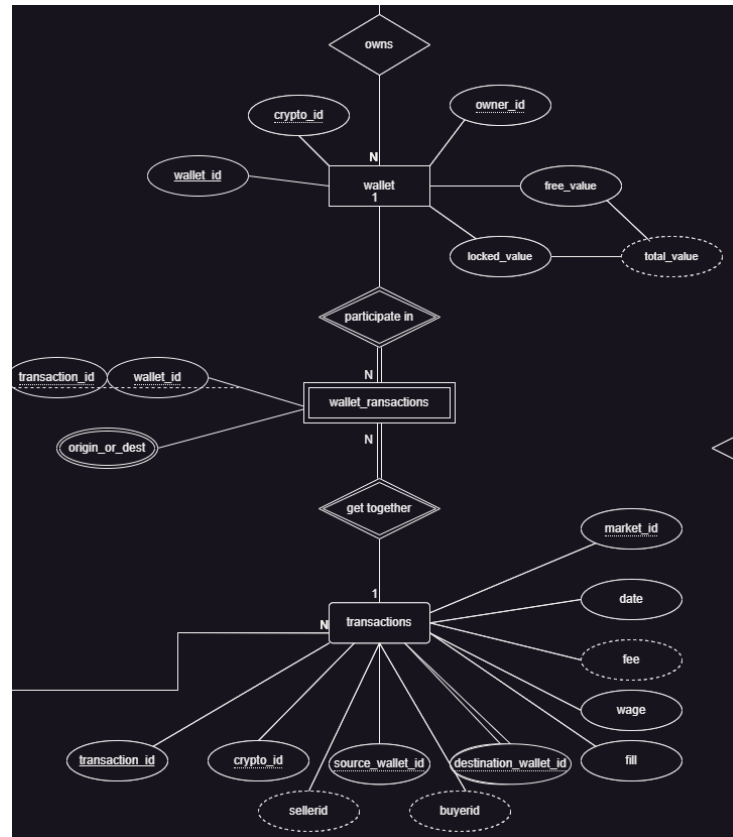
تبدیل نمودارهای فاز اول، به نمودارهای منطبق با SQL

۶.۱ چهار تغییر در نمودار برای بهینه سازی

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

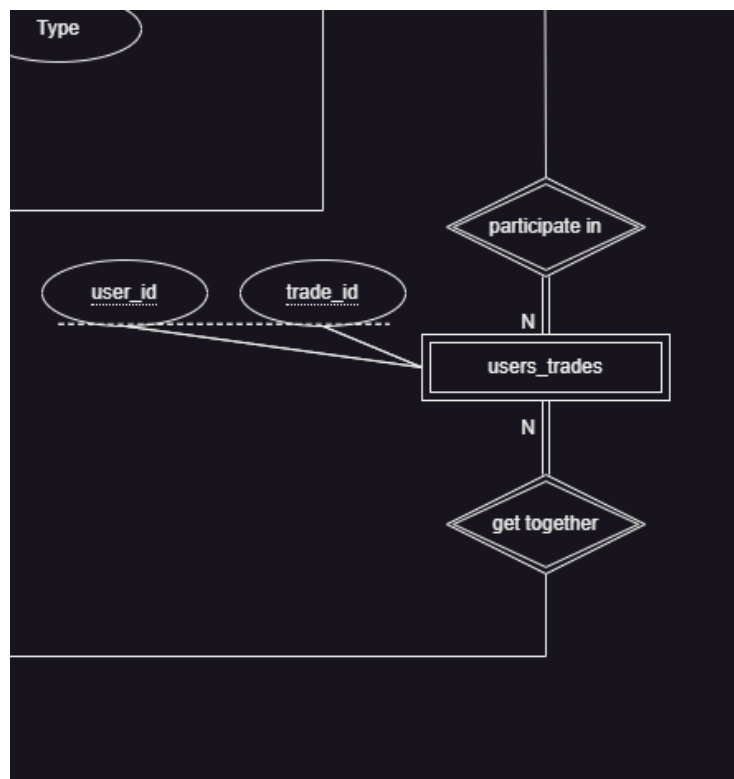
۶.۱.۱ رابطه ی چند به چند بین کیف پول و تراکنش ها

از انجایی 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.



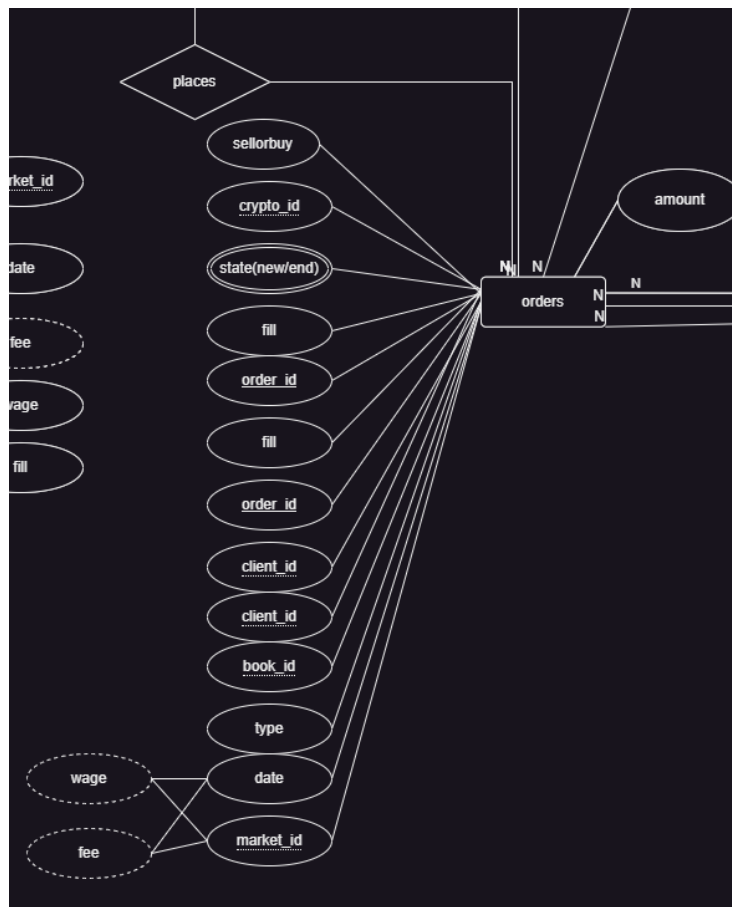
۶.۱.۲ رابطه ی چند به چند بین کاربرها و تبادلاتها

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



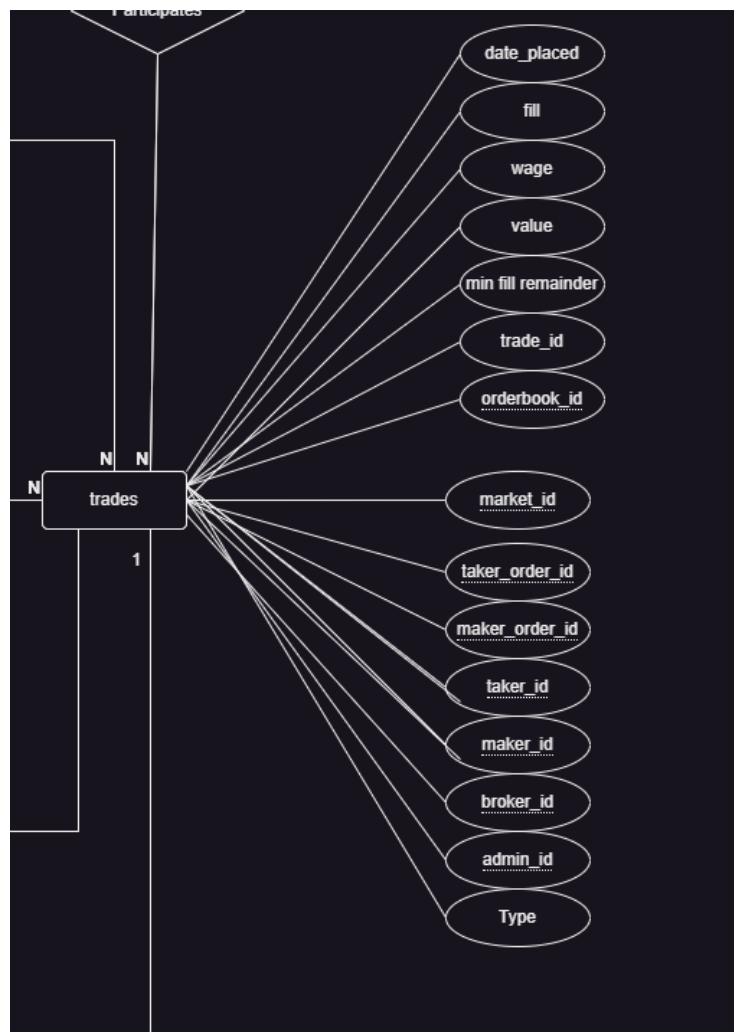
۶.۱.۳ رابطه‌ی specification در سفارشات

در فاز قبلی سفارشات به دو دسته‌ی سفارشات 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 تبادلهای

در فاز قبلی تبادلهای شامل دو نوع p2p و otc می شدند 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 جدول ها 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 نرمال‌سازی

در این بخش ما تمام جداول پایگاه داده‌ی خود را به فرم نرمال در آورده‌ایم و تغییرات نمودار و دستورات ایجاد پایگاه داده را در دو فایل `normalized_sqlform_Integrated.drawio` و `Normalized` 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 آنجایی که در $market_id \rightarrow crypto_id$ یک `non prime attribute` به یک `non prime` از `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 مجموعه‌ای 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 تغییر دادن و یا حذف کرد یک رمزارز از یک فروشگاه مجبور بودیم که دو جدول را تغییر دهیم و ممکن بود اطلاعات اشتباه تغییر دهیم ولی در حالت جدید این رابطه فقط در یک جدول وجود دارد .

۸.۱.۲ حذف $orders$ از $sales_lists_id$, $purchase_lists_id$

$orders(\underline{order_id}, sales_lists_id, purchase_lists_id, is_sell, state, fill, client_id,$
 $date, market_id, amount)$

$F.D = \{order_id \rightarrow all\ attributes, (market_id, is_sell) \rightarrow (sales_lists_id, purchase_lists_id)\}$

از انجایی که در $(market_id, is_sell) \rightarrow (sales_lists_id, purchase_lists_id)$

دو non prime attribute به دو non prime attribute دیگر اشاره می‌کند، این

دیندگی باعث می‌شود که جدول ما سومین نرمال نباشد و ما باید آن را به یک

جدول دیگر انتقال بدهیم.

ما می‌توانیم $(market_id, is_sell) \rightarrow (sales_lists_id, purchase_lists_id)$

به دو رابطه $(market_id, is_sell) \rightarrow sales_lists_id$ و $(market_id, is_sell) \rightarrow purchase_lists_id$

بشکنیم و با فرض اینکه is_sell یک boolean با دو مقدار می

باشد، دو جدول برای روابط بالا بسازیم

$orders(\underline{order_id}, is_sell, state, fill, client_id, date, market_id, amount)$

$F.D = \{order_id \rightarrow all\ attributes\}$

$is_sell == TRUE : sale_table(market_id, sales_lists_id)$

$F.D = \{market_id \rightarrow sales_lists_id\}$

$is_sell == FALSE : purchase_table(market_id, purchase_lists_id)$

$F.D = \{market_id \rightarrow purchase_lists_id\}$

همانطور که می‌دانیم 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 $market_id \rightarrow *_list_id$ در جدول دفتر سفارشات هم وجود داشت، برای تغییر دادن و یا حذف کرد یک دفتر سفارش از یک سفارش مجبور بودیم که دو جدول را تغییر دهیم و ممکن بود اطلاعات اشتباه تغییر دهیم ولی در حالت جدید این رابطه فقط در یک جدول وجود دارد .

۸.۱.۳ نرمال کردن جدول trades

$trades(trade_id, date_placed, fill, wage, value, min_fill_remainder, type, market_id, taker_order_id, maker_order_id, taker_id, maker_id, broker_id, admin_id)$

$F.D = \{trade_id \rightarrow all\ attribute, taker_order_id \rightarrow taker_id, maker_order_id \rightarrow maker_id,$

$type \rightarrow (maker_order_id, admin_id, min_fill_remainder)\}$

در مرحله‌ی اول ما دو دیپندسی $taker_order_id \rightarrow taker_id$ و $maker_order_id \rightarrow maker_id$

باعث می‌شوند جدول ما در سومین فرم نرمال قرار نگیرد به دلیل مثال

قبلی باید در جداولی دیگر قرار می‌دهیم که این دو جدول هر دو زیر مجموعه از جدول

سفارشات هستند که ما نیازی به ساختن آن‌ها نداریم.

$trades(trade_id, date_placed, fill, wage, value, min_fill_remainder, type, market_id,$

$taker_order_id, maker_order_id, broker_id, admin_id)$

$F.D = \{trade_id \rightarrow all\ attribute, type \rightarrow (maker_order_id, admin_id, min_fill_remainder)\}$

رابطه‌ی $type \rightarrow (maker_order_id, admin_id)$ یک دیپندسی کامل نیست

و به طور مثال ما اگر بدانیم که $type == 'otc'$ هست می‌توانیم نتیجه بگیریم که

$maker_order_id == NULL$ می‌باشد و ما پس از مشورت با هلد پروژه تصمیم

گرفتیم که جدول را بر اساس $type$ به دو جدول $p2p$ و otc تقسیم بکنیم.

$p2p(p2p_id, date_placed, fill, wage, value, min_fill_remainder, market_id,$

$taker_order_id, maker_order_id)$

$F.D = \{p2p_id \rightarrow all\ attribute\}$

$otc(otc_id, date_placed, fill, wage, value, taker_id, broker_id, admin_id)$

$F.D = \{otc_id \rightarrow all\ attribute\}$

Anomaly: در مرحله‌ی اول نرمال سازی همان مشکلات دو تا نرمال سازی قبلی

(تغییر دادن و حذف کردن در دو جدول) را برطرف کردیم و در مرحله‌ی دوم اگر جدول

تبادلات را به دو جدول کوچکتر تبدیل نمی‌کردیم آن وقت یک جدول بزرگ با حجم زیاد

داشتیم که مقدار بعضی از ستون‌ها در بعضی از سطرها NULL می‌شد که هم حافظه‌ی

زیادی مصرف میکرد و هم کوئری زدن رو کندتر می‌کرد.

۸.۲ index ها

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

۸.۲.۱ ایندکس تاریخ بر روی p2p

از انتحایی که ما نیاز داریم تا قیمت لحظه‌ای هر بازار را بر اساس آخرین 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 انتحایی که ما نیازمند محاسبه‌ی موجودی کیف پول هر کاربر هستیم پس بهتر است تراکنش‌های 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 انتحایی که در هر 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;

```

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;
;

```

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
        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;

```

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);
```

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);
```

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;

<<<<<< HEAD
-- 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(100, 1, 1.5);
=====

-- 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(320, 1, 1.5);
>>>>>> 509580b4540bd21de5f4979b52bb8e9d699ba3a9

```