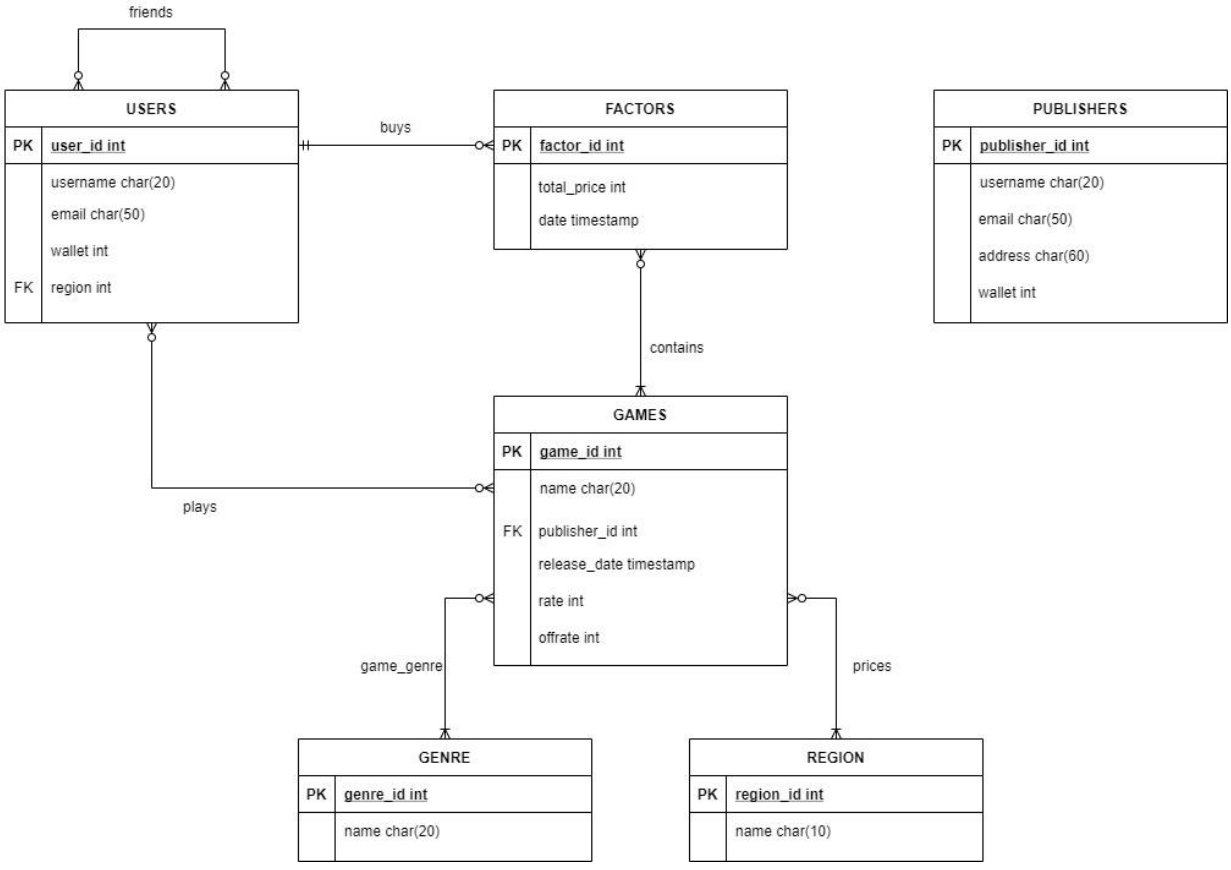
مقدمه

در این پروژه، هدف پیاده‌سازی سیستم ساده‌سازی شده سایت استیم است. در این سایت کاربران می‌توانند ثبت نام کنند و با خرید بازی، می‌توانند به بازی بپردازند. بازی‌ها توسط منتشرکننده‌ها روی سایت قرار می‌گیرد و آن‌ها می‌توانند برای بازی تخفیف بگذارند. علاوه بر این، استیم دارای شبکه‌ی اجتماعی نیز می‌باشد که در این پروژه بخش ساده‌ای از آن پیاده سازی می‌شود.

همچنین استیم، برای افراد در نواحی مختلف جهان، قیمت‌های مختلفی دارد که در خرید تاثیر می‌گذارد.

کاربران بعد از خرید بازی، می‌توانند مجموع ساعت بازی شده را برای هر بازی ببینند. علاوه بر این، صاحبان بازی قادرند به بازی رای مثبت یا منفی بدهند.

ERD اولیه در زیر قابل مشاهده است. لازم به ذکر است، بخشی از این ERD در پیاده‌سازی نهایی تغییر کرده است.



جداول ساخته شده (src/tables.sql)

- کاربران - **users**: این جدول شامل اطلاعات کاربران اصلی سیستم می‌شود که قادر به خرید بازی، امتیاز دادن به بازی و بازی کردن هستند. علاوه بر این کاربران قادر به اضافه کردن دوست نیز می-باشند.

user-id	username	email	wallet	region_id
1	amirbin	amirbin@gmail.com	42.69	1

ستون wallet نشان می‌دهد که این کاربر با واحد پولی ناحیه‌ی 1 مقدار 42.69 پول دارد.

- ژانرها - **genres**: این جدول حاوی ژانرهایی است که بازی‌های می‌توانند به آن‌ها نسبت داده شوند.

genre_id	genre_name
1	action

- ناحیه‌ها – **regions**: این جدول شامل ناحیه‌هایی است که هر کاربر باید به یکی از این ناحیه‌ها نسبت داده شود. کاربران برای خرید بازی، مقدار پول مربوط به آن ناحیه را پرداخت می‌کنند و طبیعتا واحد پولی نواحی مختلف، فرق دارد.

region_id	region_name	region_currency
1	JAPAN	¥

- منتشرکننده‌ها - **publishers**: این جدول شامل منتشرکنندگانی است که می‌توانند به لیست بازی-ها، بازی‌های ساخته شده را اضافه کنند. پول پرداختی از خرید هر بازی، به کیف پول منتشرکننده مربوطه اضافه خواهد شد.

publisher_id	publisher_name	email	address	wallet
1	Rockstar games	info@rs.com	Scottland	302.33

واحد پولی این جدول دلار است و هنگام اجرای transaction از کاربر به منتشر کننده، تبدیل واحد باید انجام شود. برای سادگی در این پروژه فرض می‌کنیم نسبت واحدهای پولی یک به یک است.

- بازی‌ها - **games**: این جدول، لیست تمام بازی‌ها را به علاوه اطلاعات اضافی از قبیل امتیاز این بازی و درصد تخفیف شامل می‌شود. امتیاز بازی‌ها که با دو ستون rate و voted نشان داده می‌شود با فرمول rate/voted نمایش داده می‌شود.

game_id	game_name	publisher_id	release_date	voted	rate	offrate
1	LA Noire	1	2011-10-10	5100	4500	33

به این بازی 5100 نفر رای داده‌اند که 4500 تای آنها رای مثبت بوده است. این بازی الان 33 درصد تخفیف دارد.

- دوستان - **friends**: این جدول شامل جفتی از آی‌دی کاربران است که با هم دوست هستند. برای سادگی ستون مربوط به پذیرفته شدن درخواست دوستی، حذف شده است.

sender_id	receiver_id
1	5

- ژانر بازی‌ها - **game_genres**: این جدول شامل جفت آی‌دی بازی و آی‌دی ژانر است که نشان می-دهد هر بازی چه ژانری دارد. همانطور که در ERD دیده شد، هر بازی می‌تواند چند ژانر داشته باشد یا ژانر نداشته باشد.

game_id	genre_id
1	2

- قیمت بازی‌ها – **game_prices**: این جدول شامل سه‌تایی های آی‌دی بازی، آی‌دی ناحیه و قیمت آن بازی در آن ناحیه است. لزومی ندارد همه‌ی بازی ها برای همه‌ی ناحیه‌ها قیمت داشته باشند و در این صورت به این معناست که آن بازی در آن ناحیه قابل خرید نیست.

game_id	region_id	price
1	1	500

بازی ۱ در ناحیه‌ی ۱، بدون تخفیف ۵۰۰ واحد پولی ناحیه‌ی یک ارزش دارد. تخفیف هنگام پرداخت اعمال می‌شود و این جدول قیمت پایه را نشان می‌دهد.

- کاربران بازی‌ها – **user_games**: این جدول اطلاعات بازی‌هایی که توسط کاربران خریداری شده است را نگهداری می‌کند. علاوه بر این مقدار ساعتی که هر کاربر آن بازی را اجرا کرده در این جدول نگهداری می‌شود. ستون آخر این جدول مربوط به این است که آیا کاربر آن بازی را پیشنهاد می‌کند یا نه که در امتیاز جدول بازی تاثیر می‌گذارد.

user_id	game_id	hours_played	recommended
1	1	25.1	true

کاربر ۱، بازی ۱ را ۲۵.۱ ساعت بازی کرده است و به آن رای مثبت داده است. رای‌ها به صورت پیشفرض خالی هستند و در امتیاز بازی اثر ندارند و بعد از رای اثر خود را می‌گذارند. پس از رای دادن، فقط می‌توان رای خود را برگرداند ولی امکان خالی گذاشتن آن دیگر وجود ندارد.

- لاگ سرپرست – **curator_logs**: این جدول تغییرات جدول ژانر بازی‌ها را ثبت می‌کند.

Action_id	Game_id_old	Game_id_new	Genre_id_old	Genre_id_new	updated
1	null	1	null	5	2020-12-31

خالی بودن old ها نشان می‌دهد که اکشن 1، یک سطر جدید به game_genres اضافه کرده است.

کاربران (src/roles/)

- ادمین استیم – steam_admin: این کاربر به همه‌ی جداول دسترسی دارد (سوپرکاربر).
- بازدیدکننده استیم – steam_visitor: این کاربر قابلیت select روی جدول games, game_genres, genres, game_prices را دارد.
- سرپرست استیم – steam_curator: این کاربر علاوه بر قابلیت کاربر بازدیدکننده، می‌تواند محتویات جدول game_genres را تغییر دهد و به genres سطرهای جدید اضافه کند.
- کاربر استیم – steam_user: این کاربر می‌تواند به جدول friends دوستان خود را اضافه کند(rls). همچنین هر کاربر قابلیت تغییر ساعت بازی شده و رای بازی‌های خودش را دارد(rls)
- منتشرکننده استیم – steam_publisher: این کاربر می‌تواند به جدول games بازی‌های خودش را اضافه کند. همین‌طور می‌تواند offrate بازی‌های خودش را تغییر دهد. علاوه بر این هر منتشرکننده می‌تواند در جدول game_prices تغییرات ایجاد کند به شرطی که خودش منتشرکننده‌ی آن بازی باشد(rls)

* توجه شود rls ها با کمک my.publisher_id و my.user_id تعیین می‌شوند. این متغیرها با دستورعمل set ست می‌شوند. در کاربرد، پس از ورود منتشرکننده یا کاربر به سیستم (با استفاده از دیتابیس شامل username و password یا روش‌های دیگر مثل jwt یا digest hash و ...) این متغیر ست می‌شود و قابلیت تغییر ندارد.

کوئری‌ها (src/queries/)

توضیحات اجرای کوئری‌ها در کامنت کد داده شده است.

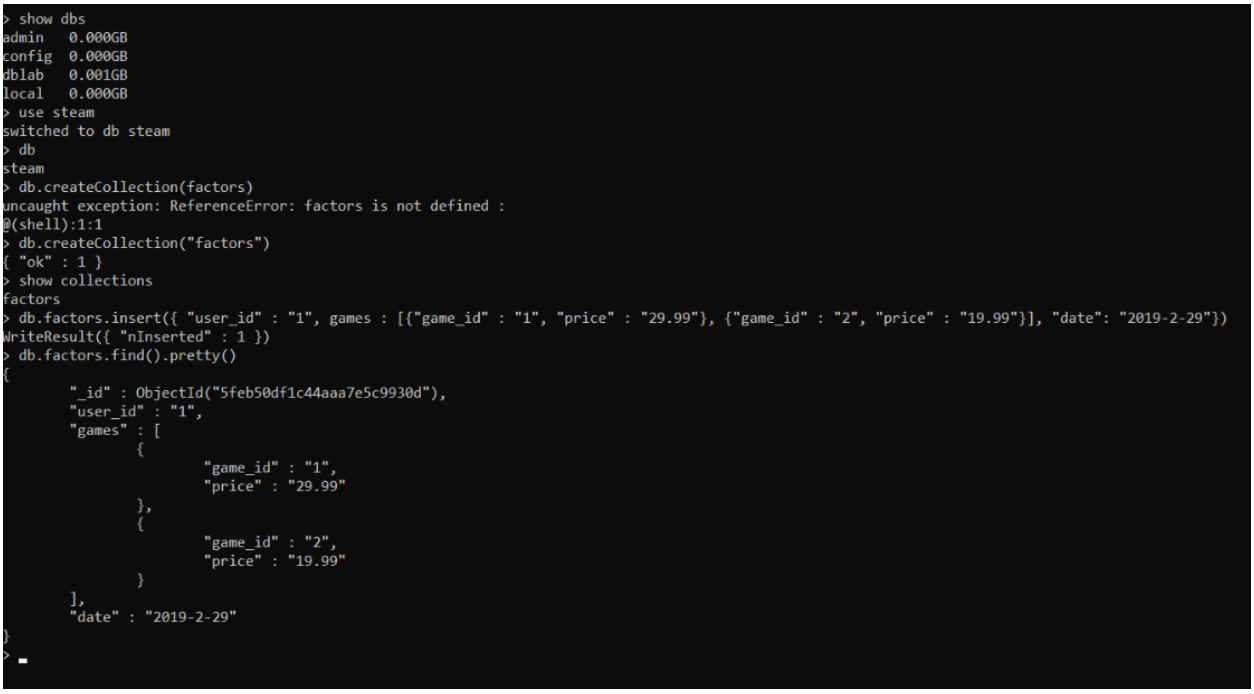
- میانگین قیمت‌ها براساس ناحیه، بازی و یا منتشرکننده(avg-price.sql)
- تعداد بازی‌های هر ژانر بر اساس کاربران و برعکس(cube.sql)
- ژانرهای یک بازی(get-genres.sql)
- میانگین ساعت‌های بازی شده هر بازی(game-avg-hours.sql)
- مجموع های بازی هر کاربر (user-sum-hours.sql)
- تعداد فروش بازی‌ها(game-sales.sql)
- تعداد بازی‌های خریده شده توسط هر کاربر(sum-user-games.sql)
- مرتب سازی ژانرهای براساس تعداد بازی‌های آن ژانر(popular-genres.sql)
- جدول فروش هر بازی برای هر منتشر کننده و کل بازی‌های آن منتشر کننده(rollup.sql)
- رنک کردن پرفروش‌ترین بازی‌ها(top-selling.sql)
- مرتب‌سازی بازی‌ها براساس امتیاز(top-rated.sql)

توابع، رویه‌ها و آغازگرها (src/funcs/)

- اضافه کردن دوست(add-friend-trigger.sql): این آغازگر قبل از اضافه کردن دوست چک می‌کند که کسی با خودش دوست نباشد. استاندارد جدول دوستان اینگونه است که sender_id باید از receiver_id کوچکتر باشد؛ با این کار که این آغازگر آن را انجام می‌دهد، دوستی‌ها دوطرفه می‌شود.
- اضافه کردن قیمت پیشفرض(add-price-trigger.sql): بعد از اضافه شدن یک بازی به جدول games، در جدول game_prices قیمت پیشفرض 19.99 برای تمام نواحی اضافه می‌شود.
- اضافه کردن بازی(add-game-trigger.sql): این آغازگر قبل از اضافه شدن بازی، مطمئن می‌شود rate و voted برابر صفر هستند و همینطور offrate بین صفر و 100 هستند.
- لاگ سرپرست(curator-log-trigger.sql): این آغازگر بعد از تغییرات در جدول game_genres، لاگ آن را ثبت می‌کند که در صورت اشتباه، لاگی از تغییرات داشته باشیم.
- حذف همه‌ی جداول(drop-all-procedure.sql): این رویه تمام جداول ساخته شده را حذف می‌کند.
- گرفتن قیمت بازی در ناحیه(get-game-price.sql): این تابع با ورودی آی‌دی بازی و ناحیه، قیمت آن بازی در آن ناحیه را برمیگرداند.
- گرفتن منتشرکننده(get-game-publisher.sql): این تابع با ورودی آی‌دی بازی، منتشرکننده‌ی آن را برمیگرداند.
- گرفتن بازی‌های منتشرکننده(get-publishers-games.sql): این تابع با ورودی آی‌دی منتشرکننده، بازی‌های آن منتشر کننده را برمیگرداند.
- اضافه کردن داده‌های اولیه(insert-data-procedure.sql): این رویه، داده‌های اولیه را اضافه می‌کند
- آپدیت امتیازات(update-scores-procedure.sql): این رویه، امتیاز بازی‌ها را با توجه به رای کاربران آپدیت می‌کند. در کاربرد این کار آخر هر روز انجام می‌شود.
- خرید بازی(buy-game-transaction.sql): این تراکنش که توسط سرور صدا زده می‌شود، مقدار پول را از کاربر کم می‌کند و به منتشرکننده اضافه می‌کند و در نهایت بازی خواسته شده را به user_games اضافه می‌کند. توجه شود که commit کردن این تراکنش، توسط سرور(پایتون) انجام می‌شود.
- تغییر ساعت‌بازی یا رای به بازی(update-user-games-trigger.sql): این آغازگر مطمئن می‌شود کاربر نتواند رای داده شده را به خالی برگرداند. همینطور مطمئن می‌شود هنگام آپدیت کردن user_id و game_id را تغییر ندهد و ساعت بازی شده را کم نکند.

اتصال به مونگو

در این پروژه، برای ذخیره‌سازی فاکتورها از mongodb استفاده کرده‌ام. نحوه‌ی ساخت و نمونه‌ی داده در تصویر زیر نشان داده شده است.



برای ساخت جدول از کد زیر استفاده شده است:

```
> use steam
> db.createCollection(factors)
```

پروتوتایپ سرور و اتصال به پایتون (src/server/)

در این قسمت، پروتوتایپی از سرور پیاده‌سازی شده که می‌تواند با هر دو دیتابیس ارتباط برقرار کند. در db.py کامنت‌های لازم وجود دارد. این سرور دو قابلیت اصلی دارد که شامل کوئری زدن به دیتابیس‌ها و تغییر دیتابیس می‌باشد.

تابع add_wallet با دو ورودی آی‌دی کاربر و مقدار پول، به آن کاربر مقدار ورودی پول اضافه می‌کند. این تابع در کاربرد توسط اینترفیس‌های درگاه پرداخت صدا زده می‌شود.

تابع buy_request که توسط کلاینت کاربر صدا زده می‌شود، آی‌دی کاربر و لیستی از بازی‌های مورد خرید می‌گیرد و خرید را انجام می‌دهد. محاسبه‌ی تخفیفات و هندل کردن خطاها در سرور انجام می‌شود و در صورتی که تراکنش‌های پرداخت به درستی انجام شوند، فاکتور خرید به دیتابیس مونگو اضافه می‌شود.