فاز دوم پروژه پایگاه داده

استاد درس: دکتر شهریاری

پاییز 1404 – دانشکده مهندسی کامپیوتر، دانشگاه صنعتی امیرکبیر

نکاتی در مورد این تمرین نیاز به توجه و دقت دوستان دارد.

1 -هرگونه کپی کردن باعث عدم تعلق نمره به تمامی افراد مشارکت کننده در آن میشود.

2 -تمامی فایلهای خود (یک فایل pdf و فایل‌های sql) را به صورت یک فایل آرشیو zip که با قالب زیر نام گذاری شده است؛ آپلود بفرمایید.

phase2_Name_LastName_StudentNumber.zip

پاسخ به هر کدام از فازها را به صورت فایلی با فرمت و اسم مطرح شده در قسمت ارائه‌ی فاز در zip نهایی خود قرار دهید و آپلود کنید.
برای برخی از فازها گزارش لازم است. تمام گزارش‌ها را در قالب یک فایل pdf به نام زیر آپلود کنید و نام header هر فاز را به شکل خواسته شده قرار دهید.
phase2_Name_LastName_StudentNumber.pdf

3 -در صورت عدم رعایت موارد فوق تصحیح انجام نخواهد شد.

مقدمه

این پروژه به 10 فاز تقسیم شده است که در فاز صفر لازم است دانشجویان نمودار ER برای سیستم بانکداری آنلاین طراحی نمایند و در فازهای بعدی طراحی آن کامل‌تر شده و پیاده‌سازی می‌شود.

نصب نرم‌افزارهای مورد نیاز

در مرحله اول لازم است که DBMS بر روی سیستم خود نصب کنید. برای این پروژه استفاده از Postgresql اجباری می‌باشد.
پس از نصب Postgresql باید یک database جدید ایجاد کنید که در ادامه پروژه تمام جدول‌ها، اطلاعات و query ها در آن ذخیره و اجرا خواهند شد.
در صورت بروز مشکل در نصب یا تنظیمات، می‌توانید از مستندات رسمی Postgresql ،آموزش‌های آنلاین یا تدریس یاران کمک بگیرید.

نوشتن کد SQL

در اکثر فازهای این پروژه لازم است که کدهای SQL را داخل فایل‌هایی با فرمت sql ذخیره کنید.
در این پروژه شما نباید تنها روی محیط خط فرمان (REPL) تکیه کنید، بلکه باید فایل SQL ایجاد کنید، query ها و دستورات را داخل آن بنویسید، و سپس آن فایل را توسط DBMS اجرا کنید. این روش بهتر این است که کد را در پنجره اجرای SQL داخل DBMS کپی کرده و اجرا کنید. زیرا کدها قابل ذخیره و مستندسازی هستند، می‌توانید در آینده آنها را مجدد اجرا یا اصلاح کنید و روند کار شما قابل بررسی خواهد بود.

فاز صفر - طراحی ER Diagram

در این فاز ابتدا توصیفی از نرم افزار ارائه میشود که شما باید برای آن یک Diagram-ER طراحی کنید.

این سامانه یک سیستم بانکداری آنلاین برای مشتریان و بانک‌ها فراهم می‌کند که به آن‌ها امکان مدیریت حساب‌ها، تراکنش‌ها، کارت‌ها، وام‌ها و پرداخت‌ها را می‌دهد. مشتریان می‌توانند از طریق اینترنت، عملیات واریز، برداشت، انتقال وجه و پرداخت‌های خود را انجام دهند. بانک‌ها نیز می‌توانند کارمندان و شعب خود را مدیریت کنند و گزارش تراکنش‌ها و وام‌ها را کنترل کنند.

ویژگی‌های سامانه:

- **مشتریان:** هر مشتری اطلاعات شخصی شامل نام، نام خانوادگی، شماره ملی، تاریخ تولد، شماره تماس، آدرس و ایمیل دارد.
- **حساب‌های بانکی:** هر مشتری می‌تواند چندین حساب داشته باشد. هر حساب شامل نوع، موجودی، تاریخ افتتاح و شعبه متعلق به آن است.
- **تراکنش‌ها:** تمام تراکنش‌ها شامل واریز، برداشت و انتقال وجه ثبت می‌شوند و هر تراکنش به یک یا دو حساب مرتبط است.
- **کارت‌های بانکی:** هر حساب می‌تواند چند کارت بانکی داشته باشد که شامل شماره کارت، تاریخ انقضا، CVV و نوع کارت است.
- **وام‌ها:** مشتریان می‌توانند وام دریافت کنند. هر وام شامل مبلغ، نوع، نرخ بهره، تاریخ شروع و پایان و حساب مرتبط است.
- **پرداخت‌ها:** پرداخت قبض و خدمات مختلف از طریق حساب مشتری انجام می‌شود.
- **بانک و شعب:** بانک‌ها شامل اطلاعات نام، شعبه مرکزی و شماره تماس هستند. هر شعبه دارای آدرس و ساعات کاری است و کارمندان به آن شعبه تعلق دارند.
- **کارمندان:** کارمندان بانک با اطلاعات شخصی و شغلی خود، به مدیریت حساب‌ها، تراکنش‌ها و مشتریان کمک می‌کنند

روابط اصلی بین موجودیت‌ها:

- هر مشتری می‌تواند چند حساب بانکی داشته باشد.
- هر حساب متعلق به یک مشتری و یک شعبه است.

- هر تراکنش به یک یا دو حساب مرتبط است.
- هر حساب می‌تواند چند کارت بانکی داشته باشد.
- هر مشتری می‌تواند چند وام و چند پرداخت داشته باشد.
- هر کارمند به یک شعبه تعلق دارد.

نکات مهم:

1. قبل از شروع طراحی، کل این سند را کامل مطالعه کنید. درک نادرست از نیازمندی‌ها باعث بازطراحی، اصلاح‌های سنگین و اتلاف زمان خواهد شد.
هدف این فاز این است که شما تصویر کامل از سیستم داشته باشید.

2. فیلدهای ذکر شده برای موجودیت‌ها فقط اطلاعات خام اولیه هستند. شما **باید** فیلدهای عملیاتی و فنی لازم برای تبدیل موجودیت‌ها به جدول‌های واقعی دیتابیس را اضافه کنید.

تحویل فاز صفر

همانگونه که در ابتدا نیز ذکر شده برای این فاز صرفا یک Diagram ER باید تحویل دهید که این ER Diagram را میتوانید به صورت دستی (خوانا و بدون خط خوردگی) یا با استفاده از نرم افزار هایی همچون creately یا draw.io طراحی کنید و بعد به صورت عکس در zip نهایی خود با اسم phase0-ER قرار دهید.

فاز اول – نرمال‌سازی پایگاه داده تا BCNF

پس از طراحی ER Diagram سامانه بانکداری آنلاین، پایگاه داده را تا سطح BCNF نرمال‌سازی کنید.

هدف این فاز این است که پایگاه داده بهینه و استاندارد باشد. یعنی جداول به گونه‌ای طراحی شوند که از افزونگی داده جلوگیری شود و اطلاعات تکراری به حداقل برسد.

همچنین تحلیل و گزارش‌گیری آسان شود یعنی تیم تحلیل داده بتواند بدون پیچیدگی به اطلاعات دسترسی داشته باشد و داده‌ها را به راحتی پردازش کند.

یکپارچگی داده‌ها نیز حفظ می‌شود.

تحویل فاز اول

در این قسمت شما باید با استفاده از دانشی که در طول ترم در مورد BCNF و روشهای normalization بدست آورده اید، تمام جدول های خود را تا حد BCNF نرمالایز کرده و گزارشی از روند normalization خود تهیه کنید و با عنوان phase1-normalization در فایل گزارش خود قرار دهید.

نکته مهم: بعد از نرمال‌سازی، لازم است که تمام وابستگی‌های تابعی (Functional Dependencies) که در طول فرآیند شناسایی شده‌اند، نیز ارسال شوند.

فاز دوم – پیاده‌سازی جداول پایگاه داده در سامانه بانکداری آنلاین

در فاز های قبلی شما پایگاه داده سامانه بانکداری آنلاین را طراحی کرده و ER Diagram اولیه را ایجاد کردید.
در این مرحله باید ER Diagram خود را به جداول رابطه‌ای تبدیل کنید و آن‌ها را در محیط پایگاه داده بسازید.

در این فاز لازم است:

- موجودیت‌های طراحی شده در ER را به جداول تبدیل کنید.
- کلیدهای اصلی (Primary Keys) هر جدول را تعیین و تعریف کنید.
- کلیدهای خارجی (Foreign Keys) را براساس روابط بین موجودیت‌ها مشخص کرده و در SQL پیاده‌سازی کنید.
- انواع داده هر ستون را مناسب با اطلاعات موجود انتخاب کنید (مثل INT، VARCHAR، DATE و ...).
- محدودیت‌های لازم را اعمال کنید (NOT NULL، UNIQUE، CHECK و ...).
- کلیدهای خارجی به گونه‌ای تعریف شوند که یکپارچگی داده تضمین شود (در صورت نیاز با ON DELETE / ON UPDATE).

تحویل فاز دوم

در این مرحله باید تمام کوئری‌های مربوط به ایجاد جدول‌های پایگاه داده (CREATE TABLE) را در یک فایل SQL قرار دهید. کلیدهای اصلی و خارجی باید در داخل همین کوئری‌ها تعریف شده باشند.
فایل خروجی با نام phase2-create.sql در فایل zip نهایی پروژه قرار داده شود.

فاز سوم – درج داده‌ها و تست پایگاه داده

در فازهای قبل جداول سامانه بانکداری آنلاین طراحی و ساخته شدند.
در این مرحله، هدف شما پر کردن جداول با داده‌های نمونه و اجرای چند پرس‌وجوی تستی است تا مطمئن شوید ساختار پایگاه داده به‌درستی کار می‌کند.

درج داده در تمام جداول

تمام جداول شما باید حداقل ۱۰ ردیف داده داشته باشند. هدف ایجاد یک پایگاه داده است که بتوان روی آن عملیات عادی بانکی را تست کرد. داده‌ها می‌توانند توسط خودتان وارد شوند یا با استفاده از Fake Data Generators مثل:

Mockaroo

GenerateData

Online Random CSV tools

Power Query

یا هر ابزار مشابه دیگر

مهم است که داده‌ها منطقی باشند (تاریخ افتتاح حساب در گذشته باشد، موجودی حساب منفی نباشد و ...)

پرس‌وجوهای تست

- نمایش اطلاعات تمام حساب‌های متعلق به یک مشتری مشخص
- نمایش ۵ تراکنش اخیر مربوط به یک حساب مشخص بر اساس تاریخ تراکنش.
- نمایش مشتریانی که بیش از یک حساب فعال دارند.
- نمایش لیست تمام حساب‌هایی که موجودی آن‌ها کمتر از یک مقدار مشخص (مثلاً ۵۰۰,۰۰۰ تومان) است.
- نمایش تعداد تراکنش‌های انجام شده در یک بازه زمانی مشخص (مثلاً یک هفته گذشته یا یک ماه گذشته).

تحویل فاز سوم

برای این مرحله باید تمام دستورهای INSERT خود برای درج داده‌ها را در یک فایل SQL قرار دهید و پرس‌وجوهای تست را نیز در انتهای همان فایل SQL قرار دهید. فایل خروجی با نام phase3-insert.sql در فایل zip نهایی پروژه قرار داده شود.

فاز چهارم - طراحی و پیاده سازی index

در این فاز قرار است کیفیت عملکرد دیتابیس را با طراحی ایندکس‌های مناسب بهبود دهید. اگرچه ایندکس‌ها ابزار قدرتمندی برای افزایش سرعت جست‌وجو و واکشی داده هستند، اما استفاده‌ی غلط از آن‌ها باعث افت کارایی، افزایش سایز دیتابیس، و هزینه‌های اضافه در عملیات درج/به‌روزرسانی می‌شود. بنابراین این مرحله قرار نیست فقط حدسی باشد، بلکه باید تحلیلی و مبتنی بر تست عملکرد واقعی باشد.

در این فاز هدف شما این است که بر اساس Query هایی که تست کرده‌اید مشخص کنید کدام ستون‌ها نیاز به ایندکس دارند.

از ایجاد ایندکس صرفاً روی primary key و foreign key و unique key constraints پرهیز کنید — آن‌ها به‌صورت پیش‌فرض ایندکس می‌شوند و هدف این فاز تحلیل فراتر از موارد بدیهی است.

برای اینکه ایندکس های خوبی بسازید، سوال های زیر را باید تحلیل کنید:

- کدام ستون‌ها بیشترین تکرار در شرط‌های WHERE را دارند؟
- کدام ستون‌ها برای فیلتر، جست‌وجو یا sort استفاده می‌شوند؟
- کدام Queryها روی داده‌های زیاد اجرا می‌شوند و کند هستند؟
- آیا آن ستون تنوع داده‌ای مناسب دارد؟

(ستونی با مقدار تکراری زیاد، مثلا جنسیت، ایندکس خوبی نیست)

نکته مهم: ایندکس باید بر اساس کاربرد باشد، نه صرفاً تعداد ردیف‌های جدول.

تحویل فاز چهارم

در این مرحله باید گزارش لیست ستون‌هایی که ایندکس شده‌اند و دلیل دقیق انتخاب آن‌ها را به گزارش خود با عنوان phase4-index اضافه کنید. فایل خروجی با نام phase4-index.sql شامل تمام دستورات ایجاد ایندکس نیز در فایل zip نهایی پروژه قرار داده شود.

حداقل ۳ ایندکس باید طراحی و تحلیل شوند.

فاز پنجم - طراحی و پیاده سازی view ها

قرار است به یکی از تیم‌های backend برای اضافه کردن چند feature جدید به اپلیکیشن مدیریت بانک به داده دسترسی بدهیم. از آن‌جایی که نمیخواهیم در مراحل develop به تمام اطلاعات مالی دسترسی داشته باشند, میخواهیم view های زیر را طراحی کنیم تا تنها با استفاده از این موارد feature های جدید را طراحی کنند. دقت کنید که این تیم فقط به داده‌های ویوهای زیر دسترسی دارد و باید بتوانند تمام اطلاعات لازم مشتری و موارد دیگر را در اپلیکیشن نشان دهند پس تشخیص ستون‌های لازم به عهده‌ی شماست. میتوانید هوشمندانه عمل کنید و برای کم کردن ستون‌های هر view از view چهارم استفاده کنید.

۱- خلاصه‌ی حساب هر مشتری

نام این view را vw_customer_accounts_summary بگذارید. مجموع و تعداد حساب‌ها مهم هستند.

۲- وام‌های فعال مشتریان

نام این view را vw_active_loans بگذارید.

۳- تراکنش های ماه اخیر هر مشتری

نام این view را vw_recent_transactions_30days بذارید. دقت کنید که در این‌جا ریز تراکنش‌ها مهم هستند نه مجموع آن ها.

۴- طراحی این view به انتخاب شماست. از این view برای ستون‌های پر تکرار استفاده کنید.

فایل خروجی با نام phase5-view.sql در فایل zip نهایی پروژه قرار داده شود. شمایی از viewها را با هدر phase5-view در فایل pdf قرار دهید.

فاز ششم - طراحی و پیاده سازی procedure ها

در این فاز میخواهیم چند Stored Procedure طراحی و پیاده‌سازی کنیم تا عملیات متداول بانکی به صورت ایمن و قابل استفاده‌ی مجدد باشد.
برای هر procedure چند خطای ممکن را در نظر بگیرید و رفتار مربوطه (مانند RAISE EXCEPTION) را برای آن در نظر بگیرید. نام هر یک را دقیقا چیزی که تعریف شده قرار دهید. procedure ها را به گونه‌ای تعریف کنید که در فاز بعدی از آن‌ها به راحتی استفاده کنید.

۱- انتقال وجه بین دو حساب - pr_transfer_funds
انجام عملیات انتقال وجه بین دو حساب بانکی به همراه ثبت تراکنش مربوطه به شکلی که ۱) موجودی حساب مبدا کافی باشد ۲) هر دو حساب معتبر و فعال باشد ۳) دو رکورد تراکنش در جدول تراکنش ثبت شود.

۲- پرداخت قبض - pr_pay_bill
انجام عملیات پرداخت از حساب مشتری با ثبت اطلاعات لازم در جداول پرداخت‌ها (با ذکر نوع پرداخت) و تراکنش‌ها.

۳- ثبت وام جدید - pr_register_loan
ایجاد یک وام جدید برای مشتری، اتصال آن به یک حساب بانکی مشخص، و واریز مبلغ وام به حساب مشتری.
فایل خروجی با نام phase6-procedures.sql در فایل zip نهایی پروژه قرار داده شود و قبل از هر procedure نام آن را در کامنت بنویسید. توضیحی خلاصه از منطق و مراحل لازم هر procedure را با هدر phase6-procedure در فایل pdf قرار دهید. دقت کنید که نوشتن مراحل هر procedure در این فاز مهم است چون در فاز بعدی باید از این مراحل استفاده کنید.

فاز هفتم - طراحی و پیاده سازی transaction

در این فاز می‌خواهیم روی همان سامانه‌ی بانکداری آنلاین، چند تراکنش طراحی کنیم که با استفاده از procedure های فاز ششم (pr_register_loan، pr_pay_bill، pr_transfer_funds) به‌صورت اتمیک اجرا شوند؛ یعنی اگر هر کدام از مراحل داخلی آن‌ها با مشکلی روبه‌رو شد، کل تغییرات به وضعیت قبل از شروع تراکنش برگردد. نام هر transaction را در قالب کامنت قبل از آن بنویسید.

تراکنش‌ها را به صورت زیر تعریف کنید:

۱- تراکنش انتقال وجه بین دو حساب - tr_transfer_funds

transaction ای بنویسید که برای یک مشتری، با استفاده از pr_transfer_funds مبلغی را از یک حساب به حساب دیگر منتقل کند.
اگر در هر مرحله خطایی رخ داد، تمام تغییرات باید به وضعیت قبل از اجرای تراکنش بازگردانی شود.

۲- تراکنش ثبت وام جدید برای مشتری - tr_register_loan

transaction ای بنویسید که با استفاده از pr_register_loan برای یک مشتری، یک وام جدید ثبت کند. اگر در هر کدام از این مراحل مشکلی پیش بیاید، وضعیت تمام جداول باید به حالت قبل از شروع تراکنش برگردد.

۳- تراکنش پرداخت قبض از حساب مشتری - tr_pay_bill

transaction ای بنویسید که با استفاده از pr_pay_bill، برای یک مشتری یک قبض را از یک حساب مشخص پرداخت کند. اگر به هر دلیل هر یک از این مراحل با خطا مواجه شد، کل تراکنش باید لغو گردد و تمامی داده‌ها به وضعیت قبل از اجرای تراکنش بازگردانی شود.

فایل خروجی را با نام phase7-transaction.sql در فایل zip نهایی پروژه قرار دهید و قبل از هر transaction نام آن را در کامنت بنویسید. دقت کنید که باید برای هر transaction باید ۳ حالت بنویسید که شامل یک حالت موفق و ۲ حالت ناموفق باشد. برای هر یک اسکرین شات خروجی‌ها را با هدر phase7-transaction در فایل pdf قرار دهید.

فاز هشتم - پیاده سازی Roles و Security

در این فاز هدف این است که با استفاده از Roleها و مجوزهای دسترسی (Privileges)، سطح دسترسی کاربران مختلف سامانه‌ی بانکداری آنلاین را کنترل کنید. این فاز باید نشان دهد که هر کاربر فقط به داده‌ها و عملیات متناسب با نقش خود دسترسی دارد و از دسترسی‌های غیرمجاز جلوگیری می‌شود. برای هر نقش یک role و کاربر نمونه بسازید و دسترسی‌های لازم را به او بدهید.

۱) نقش مدیر سیستم بانک (role_bank_admin)

این نقش بالاترین سطح دسترسی را دارد و معمولاً برای مدیر سیستم (DBA یا مدیر ارشد بانک) در نظر گرفته می‌شود. نیاز است که دسترسی کامل به همه‌ی جداول اصلی سامانه را داشته باشد

۲) نقش کارمند شعبه (role_branch_employee)

این نقش برای کارمندانی است که در شعبه‌ها با مشتریان و حساب‌ها کار می‌کنند.

وظایف و دسترسی‌ها:

- مشاهده اطلاعات مشتریان و حساب‌ها برای انجام کارهای روزمره مانند SELECT روی جداول مشتری، حساب، کارت، وام، تراکنش، پرداخت.
- ثبت عملیات مالی روزمره مانند INSERT روی جدول تراکنش و روی جدول پرداخت.
- امکان ویرایش محدود در صورت نیاز روی برخی فیلدهای غیرحساس.

فایل خروجی را با نام phase8-roles.sql در فایل zip نهایی پروژه قرار دهید.

فاز نهم (امتیازی) - پیاده سازی سیستم Audit و Log

در این فاز امتیازی، هدف این است که برای سامانه‌ی بانکداری آنلاین، یک سیستم ثبت رخداد (/ Audit Log) طراحی کنید تا تغییرات مهم داده‌ها و عملیات حساس در پایگاه داده ثبت و قابل رهگیری باشد. این سیستم باید کمک کند که بتوانیم بعداً بررسی کنیم چه کسی، چه تغییری، در چه زمانی و روی کدام رکورد انجام داده است.

۱) طراحی جداول Audit / Log

برای ثبت رخدادها، حداقل یک جدول مخصوص Audit طراحی و ایجاد کنید. به عنوان مثال:
جدول audit_log برای ثبت رخدادهای کلی (تغییر روی مشتری، حساب، وام، تراکنش و ...)
پیشنهاد می‌شود در جدول audit_log ستون‌های زیر (یا معادل آن‌ها) وجود داشته باشد:

- شناسه رخداد (کلید اصلی)
- نام جدول (مثلاً: مشتری، حساب، تراکنش، وام و ...)
- نوع عملیات (DELETE، UPDATE، INSERT)
- شناسه رکورد تحت تأثیر (مثلاً شناسه_مشتری یا شناسه_حساب)
- مقدار قدیمی داده‌ها (در صورت نیاز، مثلاً به صورت JSON یا متن)
- مقدار جدید داده‌ها (در صورت نیاز)
- نام کاربر پایگاه داده‌ای که تغییر را انجام داده (با استفاده از CURRENT_USER)
- تاریخ و زمان انجام عملیات (TIMESTAMP)

در صورت تمایل می‌توانید جداول جداگانه برای انواع رخدادهای خاص (مثل تغییرات حساب، وام، تراکنش) تعریف کنید، اما داشتن یک جدول کلی audit_log کافی است.

۲) تعریف Trigger برای ثبت رخدادها

برای حداقل دو جدول اصلی سامانه باید Trigger هایی تعریف کنید که در زمان انجام عملیات زیر اجرا شوند:

- بعد از INSERT
- بعد از UPDATE
- بعد از DELETE

و هر بار که این عملیات انجام شد، یک رکورد مناسب در جدول audit_log ثبت شود.

Trigger باید به یک تابع (Function) متصل باشد که عمل ثبت اطلاعات را در جدول audit_log انجام می‌دهد.

۳) ثبت اطلاعات کاربر و زمان انجام عملیات

در پیاده‌سازی Trigger ها و Function های مربوط به Audit، لازم است حداقل موارد زیر ثبت شوند:

- کاربری که عملیات را انجام داده
- تاریخ و زمان دقیق انجام عملیات
- این اطلاعات برای پیگیری امنیتی و بررسی‌های بعدی ضروری هستند.

۴) طراحی View یا گزارش برای Audit

برای راحت‌تر دیدن و تحلیل لاگ‌ها، حداقل یک view تعریف کنید، مثلاً:

- View برای نمایش آخرین تغییرات روی حساب‌ها
- View برای نمایش تاریخچه‌ی تغییر وضعیت وام‌ها

تمام دستورات SQL مربوط به این فاز را در فایل phase9_audit.sql قرار دهید. این فایل sql باید شامل تست‌های شما هم باشد. به طور مختصر فرایند را توضیح دهید و اسکرین‌شات‌های لاگ‌ها را با هدر phase9_audit در فایل pdf قرار دهید.

این فاز امتیازی است و پیاده‌سازی تمیز و مستند آن می‌تواند در نمره‌ی پروژه تأثیر مثبت داشته باشد.