



به نام خدا
دانشگاه تهران
دانشکده مهندسی برق و کامپیوتر



درس آزمایشگاه پایگاه داده دستورکار هفتم

گزارش کار دستورات سطح پیشرفته SQL

محمد شفیعیها و محمد عراقی

فهرست

1	قوانین
2	بخش ۱. مقدمه
2	۱-۱
2	۱-۲
3	بخش ۲ - دستورالعمل اجرایی
3	۱-۲. آماده سازی
4	بخش ۳ - دستورات
4	۱-۳. سوال اول
5	۲-۳. سوال دوم
6	۲-۳. سوال سوم
7	۴-۳. سوال چهارم
8	۵-۳. سوال پنجم
9	۶-۳. سوال ششم و آخر

قبل از پاسخ دادن به پرسش‌ها، موارد زیر را با دقت مطالعه نمایید:

- دستورکارهای حضوری به صورت دونفره انجام می‌شود و دستورکارهای غیرحضوری باید به صورت تک‌نفره انجام شود. توجه نمایید الزامی در یکسان ماندن اعضای گروه تا انتهای ترم وجود ندارد. (یعنی، می‌توانید تمرین اول را با شخص A و تمرین دوم را با شخص B و ... انجام دهید)
- در صورت داشتن هرگونه سوال با دستیاران آموزشی این تمرین از طریق رایانامه‌های زیر در ارتباط باشید:

m.shafiecha@ut.ac.ir

mamadiaraghi80@gmail.com

بخش ۱. مقدمه

هدف اصلی از این تمرین، آشنایی عملی با مفاهیم «تابع» و «تریگر» در پایگاه داده رابطه‌ای است.

۱-۱. معرفی توابع پنجره‌ای

در زبان SQL، توابع نقش مهمی در پردازش و تحلیل داده‌ها دارند. مشابه دیگر زبان‌های برنامه نویسی، توابع به شما امکان می‌دهند تا عملیات‌های مختلف را بر روی داده‌های خود انجام دهید و نتایج مورد نظر از جمله یک عدد یا یک جدول را به دست آورید.

توابع پنجره در SQL مکانیزم قدرتمندی برای انجام محاسبات در محدوده مشخصی از ردیف‌های کوئری هستند. برخلاف توابع تجمعی، توابع پنجره‌ای امکانی را فراهم می‌کنند تا به ازای هر رکورد، مقداری را بر اساس گروه بندی مرتبط با آن گروه ارائه کنیم (مثل رتبه دانشجو در یک کلاس خاص). ضرورت توابع پنجره‌ای هنگام پرداختن به کارهای تحلیلی معلوم می‌شود، چون به تحلیلگران این امکان را می‌دهند تا با در نظر گرفتن روابط بین ردیف‌ها در یک مجموعه خاص از نتایج، تحلیل‌های خاصی را ارائه کنند.

۱-۲. معرفی تریگرها

تریگرها در یک پایگاه داده مثل واکنش‌های خودکار به اتفاقات خاص (مثل اضافه کردن یا به روز رسانی داده‌ها) هستند. تریگرها مثل دستیارهای داخلی ای هستند که به اطمینان از سازماندهی پایگاه داده و رفتار طبق قوانین از پیش تعریف شده کمک می‌کنند. در اصل، تریگرها با انجام خودکار یکسری کار، احتمال خطا را کم می‌کنند.

مثلاً فرض کنید به ازای هر تراکنش مالی (برداشت یا واریز به حساب)، نیاز داریم به دلیل مسائل امنیتی، مقدار قبلی حساب را در یک جدول جداگانه به نام `transaction_history` به صورت خودکار ذخیره کنیم (عملیات Audit Log).

بخش ۲ - دستورالعمل اجرایی

۱-۲. آماده سازی

ابتدا به یک جدول برای کار کردن نیاز داریم، که در این تمرین داده‌های هویتی حساب کاربری را ذخیره می‌کنیم. پس قبل از شروع به حل تمرین با دستورات زیر جداول مورد نیاز برای حل سوالات را بسازید:

```
1 CREATE TABLE person (  
2     login_name varchar(9) not null primary key,  
3     display_name text  
4 );
```

شکل ۱. ساخت جدول

جدول حسابرسی معمولاً شامل همان ویژگی‌های جدول اصلی است که برای ثبت مقادیر تغییر یافته و ویژگی‌های اضافی یا ایجاد تغییر و یک مهر زمانی برای تراکنش استفاده می‌شود.

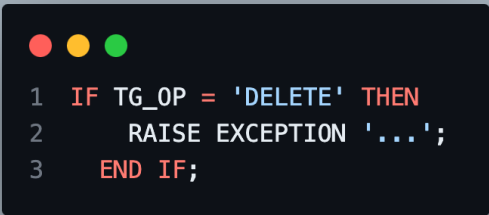
```
1 CREATE TABLE person_audit (  
2     login_name varchar(9) not null,  
3     display_name text,  
4     operation varchar,  
5     effective_at timestamp not null default now(),  
6     userid name not null default session_user  
7 );
```

شکل ۲. ساخت جدول حسابرسی

بخش ۳ – دستورات

۳-۱. سوال اول

تابعی بنویسید تا بررسی کند اسم خالی و دارای فاصله نباشد و سپس login name, display name و نوع عملیات (TG_OP) در جدول person_audit را ذخیره کند. همچنین برای هر عملیات یک تست ساده بنویسید و نتایج را به دستیار آموزشی نشان دهید.

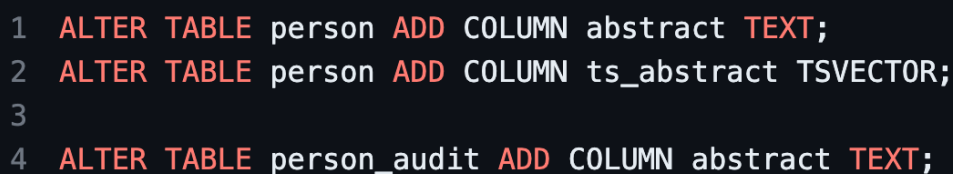


```
1 IF TG_OP = 'DELETE' THEN
2     RAISE EXCEPTION '...';
3 END IF;
```

شکل ۳. راهنمایی برای عمل حذف

۲-۳. سوال دوم

برای بهبود قابلیت جستجوی متنی، ما سه ستون جدید ("abstract"، "ts_abstract"، "abstract") را اضافه می‌کنیم. به طور خلاصه، این تغییرات جدول "person" را بزرگتر می‌کنند تا ذخیره اسناد و جستجوی متنی نیز وارد داستان شود. و همینطور جدول "person_audit" به‌روز می‌شود تا ستونی برای ذخیره سند مرتبط با هر audit داشته باشد.



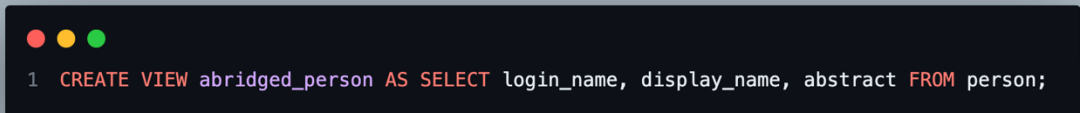
```
1 ALTER TABLE person ADD COLUMN abstract TEXT;  
2 ALTER TABLE person ADD COLUMN ts_abstract TSVECTOR;  
3  
4 ALTER TABLE person_audit ADD COLUMN abstract TEXT;
```

شکل ۴. پیش نیاز حل سوال دوم

تابع بخش اول را به گونه ای بنویسید که اطلاعات اضافی به دست آمده از متن abstract توسط تابع آماده to_tsvector را در ستون ts_abstract بریزد. و همینطور یک تست ساده بنویسید و نتایج را به دستیار آموزشی نشان دهید.

۲-۳. سوال سوم

در سناریویی که کاربر سعی کند مقداری را برای ستون `ts_abstract` درج کند، هر چیزی که وارد کند دور ریخته می‌شود و با مقداری که از داخل تابع `trigger` بدست آمده جایگزین می‌شود. هدف این کار حفاظت در برابر تغییرات غیر قابل قبول در جستجو است. برای مخفی کردن این ستون، می‌توانیم یک نمای خلاصه تعریف کنیم که شامل آن ویژگی نیست، اما همچنان از مزایای آن بهره می‌بریم.

A screenshot of a terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top left corner. The terminal displays a single line of SQL code: `1 CREATE VIEW abridged_person AS SELECT login_name, display_name, abstract FROM person;`

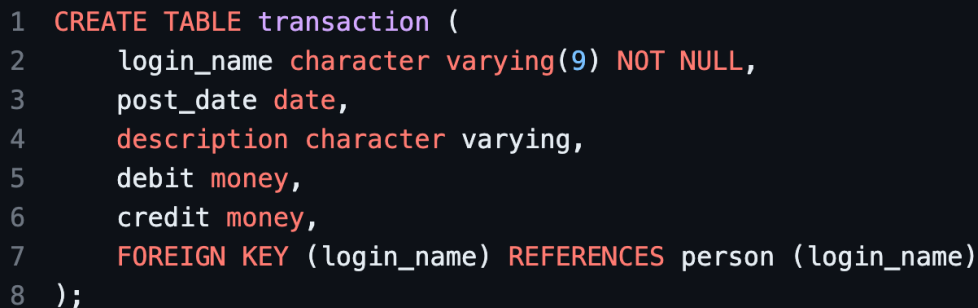
```
1 CREATE VIEW abridged_person AS SELECT login_name, display_name, abstract FROM person;
```

شکل ۵. پیش نیاز حل سوال سوم

در این شرایط آیا تریگرهای مان برای ویو هم کار میکنند یا نیاز به بازنویسی آن ها داریم؟

۳-۴. سوال چهارم

سناریویی را تصور کنید که در آن نوعی جدول تراکنش وجود دارد و شامل سابقه ای از ساعات کار، اضافه شدن موجودی و کاهش موجودی و مواردی از این دست برای هر فرد باشد.



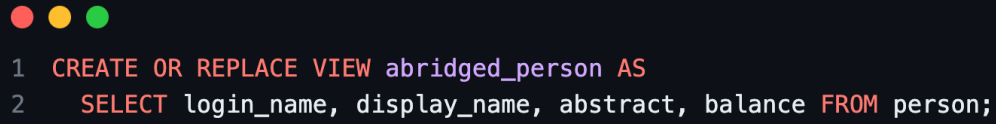
```
1 CREATE TABLE transaction (  
2     login_name character varying(9) NOT NULL,  
3     post_date date,  
4     description character varying,  
5     debit money,  
6     credit money,  
7     FOREIGN KEY (login_name) REFERENCES person (login_name)  
8 );
```

شکل ۶. پیش نیاز حل سوال چهارم

تابعی بنویسید تا قبل برای هر عملیات مالی بررسی کند که مقدار credit و debit منفی نباشد. همچنین در نظر گرفته شود که کاربر پول کافی برای این عملیات را داشته باشد اگر نه با خطا مواجه شود. (هزینه عملیات برای کاربر: debit – credit). همینطور یک تست ساده بنویسید و نتایج را به دستیار آموزشی نشان دهید.

۳-۵. سوال پنجم

برای حل این سوال امکان دسترسی خواندن به موجودی را فراهم کنید:

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top left corner. It contains two lines of SQL code.

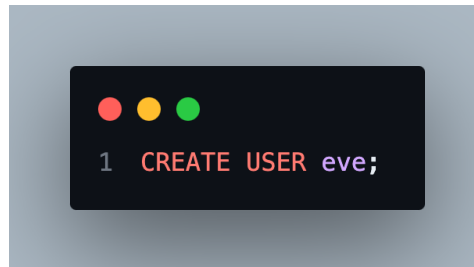
```
1 CREATE OR REPLACE VIEW abridged_person AS
2   SELECT login_name, display_name, abstract, balance FROM person;
```

شکل ۷. پیش نیاز حل سوال پنجم

تابعی بنویسید که امکان تغییر مستقیم پول کاربر را بدون transactin (عملیات مالی) ندهد. یک تست ساده بنویسید و نتایج را به دستیار آموزشی نشان دهید.

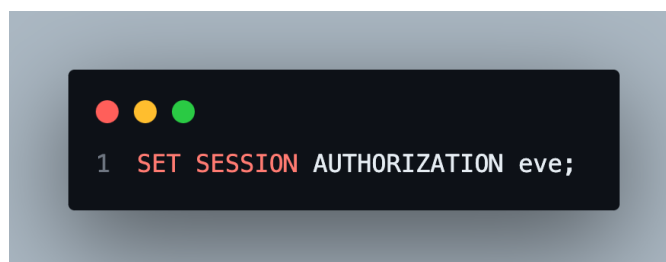
۳-۶. سوال ششم و آخر

این مثال به ما نشان می‌دهد که چگونه می‌توان از تریگرها و توابع برای اجازه اجرای کد توسط یک کاربر غیرمجاز با امتیاز بالاتری نسبت به کاربر وارد شده استفاده کرد. پس یک نقش ورود غیرمجاز تعریف می‌کنیم. با استفاده از Grant امکان های روبرو را به کاربرمان بدهید: امکان خواندن و نوشتن و افزودن به ویوی abridged_person + امکان خواندن و افزودن عملیات مالی



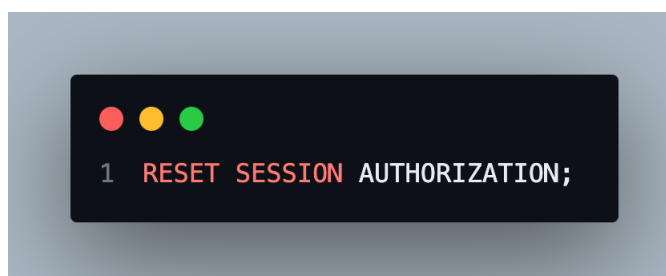
شکل ۸. راهنمایی اول حل سوال ششم

با نوشتن یک تست موفق و یک تست ناموفق نشان دهید کاربر امکان انجام بقیه عملیات ها را ندارد.



شکل ۹. راهنمایی دوم حل سوال ششم

چرا کاربر امکان افزودن عملیات مالی جدید را ندارد؟ (با استفاده از SECURITY DEFINER این مشکل را حل کنید).



شکل ۱۰. راهنمایی سوم حل سوال ششم