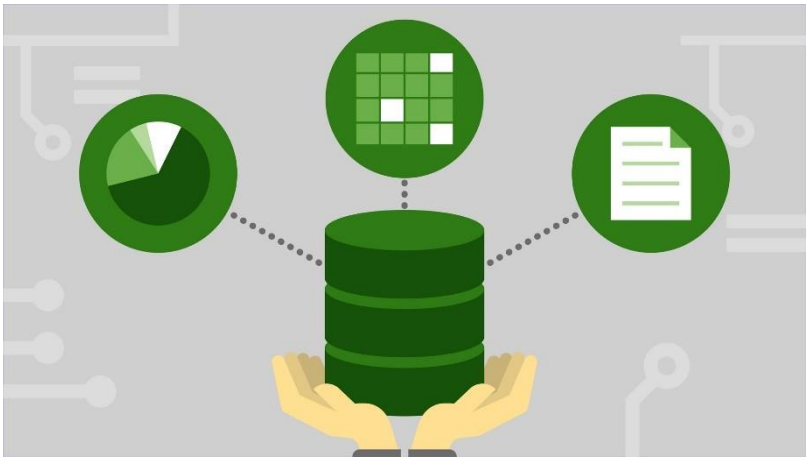
به نام خدا

دانشگاه تهران
پردیس دانشکده‌های فنی
دانشکده برق و کامپیوتر



آزمایشگاه پایگاه داده

دستور کار شماره ۵

نام و نام خانوادگی
معین شیردل ۸۱۰۱۹۷۵۳۵

آبان ماه ۱۴۰۰

توابع و تریگرها (از سایت severalnines):

```
ALTER TABLE person
    ADD CONSTRAINT PERSON_LOGIN_NAME_NON_NULL
    CHECK (LENGTH(login_name) > 0);

ALTER TABLE person
    ADD CONSTRAINT person_login_name_no_space
    CHECK (POSITION(' ' IN login_name) = 0);
```

پیش از این، جدول person با دو مقدار login_name و display_name ساخته شد و به کمک دستور بالا، دو محدودیت روی مقادیر ورودی برای login_name گذاشته شد که رشته خالی نباشد و در آن کاراکتر space به کار نرفته باشد.

```
CREATE OR REPLACE FUNCTION person_bit()
    RETURNS TRIGGER
    SET SCHEMA 'public'
    LANGUAGE plpgsql
    AS $$
    BEGIN
    IF LENGTH(NEW.login_name) = 0 THEN
        RAISE EXCEPTION 'Login name must not be empty.';
    END IF;

    IF POSITION(' ' IN NEW.login_name) > 0 THEN
        RAISE EXCEPTION 'Login name must not include white space.';
    END IF;
    RETURN NEW;
    END;
    $$;


CREATE TRIGGER person_bit
    BEFORE INSERT ON person
    FOR EACH ROW EXECUTE PROCEDURE person_bit();
```

به کمک دستورات بالا، ابتدا یک تابع به نام person_bit() نوشته شد و یک Trigger برای آن قرار داده شد که در هنگام وارد شدن هر رکورد به جدول person تابع person_bit() را با سطر جدید صدا کند و آن را بررسی کند. در این تابع، یک exception مناسب به ازای ورودی های خالی یا دارای space برای login_name داده می‌شود که نتیجه آن در تصویر زیر مشخص است (پس از ورود یک رکورد با login_name دارای space)

```
INSERT INTO person VALUES ('space man', 'Major Tom');
```

Statistics 1 ×

INSERT INTO person VALUES ('space man', 'Major Tom') Enter a SQL expression to

SQL Error [P0001]: ERROR: Login name must not include white space.
Where: PL/pgSQL function person_bit() line 8 at RAISE

```
CREATE TABLE person_audit (
    login_name varchar(9) not null,
    display_name text,
    operation varchar,
    effective_at timestamp not null default now(),
    userid name not null default session_user
);

CREATE OR REPLACE FUNCTION person_bit()
    RETURNS TRIGGER
    SET SCHEMA 'public'
    LANGUAGE plpgsql
    AS $$
    BEGIN
    IF LENGTH(NEW.login_name) = 0 THEN
        RAISE EXCEPTION 'Login name must not be empty.';
    END IF;

    IF POSITION(' ' IN NEW.login_name) > 0 THEN
        RAISE EXCEPTION 'Login name must not include white space.';
    END IF;

    -- New code to record audits

    INSERT INTO person_audit (login_name, display_name, operation)
        VALUES (NEW.login_name, NEW.display_name, TG_OP);

    RETURN NEW;
    END;
    $$;


DROP TRIGGER person_bit ON person;

CREATE TRIGGER person_biut
    BEFORE INSERT OR UPDATE ON person
    FOR EACH ROW EXECUTE PROCEDURE person_bit();
```

```
CREATE OR REPLACE FUNCTION person_bdt()
    RETURNS TRIGGER
    SET SCHEMA 'public'
    LANGUAGE plpgsql
    AS $$
    BEGIN

    -- Record deletion in audit table

    INSERT INTO person_audit (login_name, display_name, operation)
      VALUES (OLD.login_name, OLD.display_name, TG_OP);

    RETURN OLD;
    END;
    $$;

CREATE TRIGGER person_bdt
    BEFORE DELETE ON person
    FOR EACH ROW EXECUTE PROCEDURE person_bdt();

INSERT INTO person VALUES ('dfunny', 'Doug Funny');
INSERT INTO person VALUES ('pmayo', 'Patti Mayonnaise');

SELECT * FROM person;
SELECT * FROM person audit;
```

به کمک تکه کد های بالا، یک تریگر برای زمان آپدیت یا insert در جدول person ایجاد شد که هر اقدامی برای insert یا update کردن فیلدهای person را در جدول person_audit ثبت می‌کند. تریگر person_bdt نیز برای اقدام به دیلیت کردن رکورد های جدول person است. در هر حالت، مقدار رکورد های دیلیت یا insert شده به همراه زمان آن ها و نوع عملیات صورت گرفته و کاربر انجام دهنده ثبت می‌شود. در نهایت با insert کردن دو رکورد به عنوان مثال در جدول person، و آپدیت کردن یکی به یک مقدار جدید و پاک کردن دیگری، رکوردهای زیر در جدول person_audit ثبت می‌شوند.

	login_name	display_name	operation	effective_at	userid
1	dfunny	Doug Funny	INSERT	2021-12-26 22:06:03.944	postgres
2	pmayo	Patti Mayonnaise	INSERT	2021-12-26 22:06:07.118	postgres
3	dfunny	Doug Yancey Funny	UPDATE	2021-12-26 22:19:38.930	postgres
4	pmayo	Patti Mayonnaise	DELETE	2021-12-26 22:19:41.533	postgres

---

در قسمت بعدی، یک تابع به عنوان تریگر پیش از insert طراحی شد تا متن های طولانی را به text-search vector هایی تبدیل کند و کار جستجوی متنی را ساده‌تر کند.

```
CREATE OR REPLACE FUNCTION person_bit()
    RETURNS TRIGGER
    LANGUAGE plpgsql
    SET SCHEMA 'public'
    AS $$
    BEGIN
    IF LENGTH(NEW.login_name) = 0 THEN
        RAISE EXCEPTION 'Login name must not be empty.';
    END IF;

    IF POSITION(' ' IN NEW.login_name) > 0 THEN
        RAISE EXCEPTION 'Login name must not include white space.';
    END IF;

    -- Modified audit code to include text abstract

    INSERT INTO person_audit (login_name, display_name, operation, abstract)
        VALUES (NEW.login_name, NEW.display_name, TG_OP, NEW.abstract);

    -- New code to reduce text to text-search vector

    SELECT to_tsvector(NEW.abstract) INTO NEW.ts_abstract;

    RETURN NEW;
    END;
    $$;

UPDATE person SET abstract = 'Doug is depicted as an introverted, quiet, insecure and gullible 11 (later 12) ;
```

Statistics 1 | person... ×

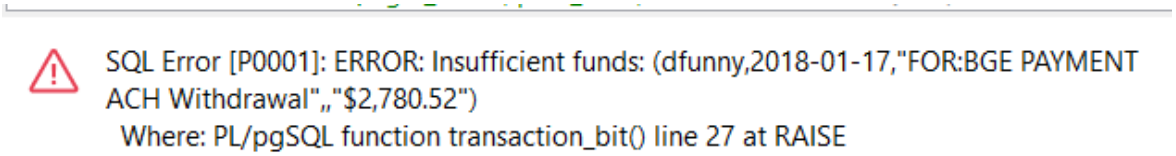
SELECT p.login_name, p.ts_abstract from person p | Enter a SQL expression to filter results (use Ctrl+Space)

	login_name	ts_abstract
1	dfunny	'11':11 '12':13 'boy':16 'crowd':24 'depict':3 'doug':1 'fit':20 'gullibl':10 'insecur':8 'introvert':6 'later':12 'old':15 'quiet':7 'want':18 'yea

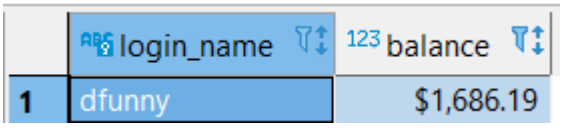
در اینجا، مقادیر وارد شده در ستون abstract پیش از وارد شدن، در تابع person_bit به کمک to_tsvector تبدیل به TSVector می‌شوند که کلمات ستون abstract در اصل split شده و سورت شده‌اند و در ستون اضافه شده به نام ts_abstract قرار داده می‌شوند. این نوع دیگری از کاربردهای توابع و تریگرهاست که برای تبدیل انواع داده استفاده می‌شود.

```
CREATE FUNCTION transaction_bit() RETURNS trigger
    LANGUAGE plpgsql
    SET SCHEMA 'public'
    AS $$
    DECLARE
    newbalance money;
    BEGIN

    -- Update person account balance
    UPDATE person
        SET balance =
            balance +
            COALESCE(NEW.debit, 0::money) -
            COALESCE(NEW.credit, 0::money)
        WHERE login_name = NEW.login_name
                RETURNING balance INTO newbalance;

    -- Data validation
    IF COALESCE(NEW.debit, 0::money) < 0::money THEN
        RAISE EXCEPTION 'Debit value must be non-negative';
    END IF;

    IF COALESCE(NEW.credit, 0::money) < 0::money THEN
        RAISE EXCEPTION 'Credit value must be non-negative';
    END IF;

    IF newbalance < 0::money THEN
        RAISE EXCEPTION 'Insufficient funds: %', NEW;
    END IF;

    RETURN NEW;
    END;
    $$;
```

به کمک کد بالا، یک محدودیت روی مقادیر جدید وارد شده در جدول transaction اعمال می‌کنیم. این جدول، اطلاعات تراز مالی افراد را دارد و در خود، یک primary key از رکوردهای جدول person دارد. به کمک محدودیت، اجازه نمیدهیم تراز مالی افراد در اثر ورودی یک رکورد جدید منفی شود و اگر یک عملیات ناموفق بود و به منفی شدن تراز فردی منجر می‌شد، آن را roll back میکنیم و مراحلش را undo میکنیم. به طور مثال پس از شارژ کردن حساب به اندازه ۲۰۰۰ دلار و بعد اقدام به برداشت ۲۷۸۰ دلار، تراز منفی خواهد شد و در اثر این اقدام، ارور پایین را میگیریم و اقدامی انجام نمیشود و تراز به همان مقدار +۲۰۰۰ دلار بازمیگردد:



پس از این مرحله، میبینیم که پس از دو برداشت پول (کسر اعتبار) به مقادیر ۲۷۸ و ۳۵ دلار، مقدار اعتبار یا تراز مالی به اندازه‌ای که در تصویر مشخص است در می‌آید.



	login_name	balance
1	dfunny	$1,686.19

در قسمت بعدی، یک محدودیت روی فیلد balance گذاشته شد که نتوان آن را به صورت دستی تغییر داد و حتما با یک transaction تغییر کند. بدین صورت میبینیم که با اجرای دستور update در کد پایین، مقدار balance تغییری نکرده و همان مقدار قبلی را خواهد داشت.

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


 CREATE FUNCTION abridged_person_iut() RETURNS TRIGGER
    LANGUAGE plpgsql
    SET search_path TO public
    AS $$
    BEGIN

    -- Disallow non-transactional changes to balance

      NEW.balance = OLD.balance;
    RETURN NEW;
    END;
    $$;

CREATE TRIGGER abridged_person_iut
    INSTEAD OF UPDATE ON abridged_person
    FOR EACH ROW EXECUTE PROCEDURE abridged_person_iut();

UPDATE abridged_person SET balance = '1000000000.00';

SELECT login_name, balance FROM abridged_person WHERE login_name = 'dfunny';
```

abridged_person 1 ×

SELECT login_name, balance FROM abridged_person W | Enter a SQL expression to filter results (use Ctrl+Sp

	login_name	balance
1	dfunny	$1,686.19

به کمک دستور \dp میتوان لیستی از دسترسی ها و اختیارات در دیتابیس مشاهده نمود.

به کمک دستورات زیر، یک کاربر به نام eve ساخته شده که دسترسی های مشخصی به آن داده شده است. هنگام تغییر کاربر از postgres (که تمامی دسترسی ها را دارد) به eve، امکان دیدن جداول person و person_audit از ما گرفته می‌شود و با ارور permission denied مواجه می‌شویم. البته این محدودیت روی جداول دیگر مثل transaction وجود ندارد چون روی آن ها تعریف نشده است.

```
CREATE USER eve;
GRANT SELECT,INSERT, UPDATE ON abridged_person TO eve;
GRANT SELECT,INSERT ON transaction TO eve;

\dp access privileges


SET SESSION AUTHORIZATION eve;
select * from person;
select * from person_audit;
```

Statistics 1 | Statistics... ×

select * from person_audit | Enter a SQL expression to filter results (use Ctrl+S

SQL Error [42501]: ERROR: permission denied for table person_audit

توابع پنجره‌ای (Window Functions):

سوال اول) رتبه‌ی هر مشتری در کشور خودش از لحاظ تعداد سفارش

```
select customer_order_count.*,
    RANK() over (partition by customer_order_count.country order by customer_order_count.order_count desc) rank_in_country
from (select c.customer_id, c.country, COUNT(o.order_id) as order_count
    from customers c join orders o on o.customer_id = c.customer_id
    group by c.customer_id, c.country
    ) as customer_order_count
order by country;
```

به کمک تکه کد بالا و استفاده از تابع پنجره ای rank روی هر شهر، مورد خواسته شده محاسبه شد و نتیجه آن قابل مشاهده است:

	customer_id	country	order_count	rank_in_country
1	CACTU	Argentina	6	1
2	OCEAN	Argentina	5	2
3	RANCH	Argentina	5	2
4	ERNSH	Austria	30	1
5	PICCO	Austria	10	2
6	SUPRD	Belgium	12	1
7	MAISD	Belgium	7	2
8	HANAR	Brazil	14	1
9	QUEEN	Brazil	13	2
10	RICAR	Brazil	11	3
11	GOURL	Brazil	9	4
12	QUEDE	Brazil	9	4
13	WELLI	Brazil	9	4
14	FAMIA	Brazil	7	7
15	TRADH	Brazil	6	8
16	COMMI	Brazil	5	9

سوال دوم) تعداد اقلام باقی مانده از گرانترین کالای فروشگاه در هر دسته بندی

```
select cp.product_name, cp.category_id, cp.units_in_stock
from (
    select p.*,
        max(p.unit_price) over (partition by p.category_id) as highest_price
    from products p
    ) as cp
where cp.unit_price = cp.highest_price;
```

به کمک تکه کد بالا از تابع پنجره ای max برای یافتن گرانترین کالای هر کتگوری استفاده شد و نتایج زیر حاصل شد:

	product_name	category_id	units_in_stock
1	Côte de Blaye	1	17
2	Vegie-spread	2	24
3	Sir Rodney's Marmalade	3	40
4	Raclette Courdavault	4	79
5	Gnocchi di nonna Alice	5	21
6	Thüringer Rostbratwurst	6	0
7	Manjimup Dried Apples	7	20
8	Carnarvon Tigers	8	42

سوال سوم) اسم و فامیل جوان‌ترین کارمند فروشگاه

```
select ELB.first_name, ELB.last_name, ELB.birth_date
from (
    select e.*,
        RANK() over (order by e.birth_date desc) as birthday_rank
    from employees e
    ) as ELB
where ELB.birthday_rank = 1;
```

در تکه کد بالا، به کمک تابع پنجره ای rank، فردی که رتبه اول در ترتیب نزولی تاریخ تولد را دارد (جوان ترین) انتخاب می‌شود و نامش استخراج می‌شود.

	first_name	last_name	birth_date
1	Anne	Dodsworth	1966-01-27

سوال امتیازی اول:

در این سوال، تعدادی جداول کمکی در ابتدا ساخته شده اند. یکی از آنها PurchasingUsers است که شامل کاربرانی است که حداقل یک خرید (یک رکورد با action_type = buy) داشته اند.

جدول MovingUsers که شامل کاربرانی است که از بیش از یک مکان (zipcode متفاوت) خرید داشته اند.

و جدول UserSessionMetrics که بیانگر تعداد کلیک، خرید و لاگین های کاربران در هر سشن شان است.

در نهایت جدول سوم با دو جدول اول inner join میخورد و خروجی حاصل، شامل کاربرانی است که از بیش از یک مکان مختلف به سرور وصل شده اند و حداقل یک خرید نیز داشته‌اند. برای این کاربران، تمام سشن هایشان به همراه تعداد خریدها و لاگین ها و کلیک های آنها در این سشن ها آمده است.

در ساختن این جداول، دستور case به نوعی مانند if عمل می‌کند و برای شمارش اکشن ها استفاده شده است که بتوانیم به راحتی تعداد اکشن های کلیک، خرید و لاگین را محاسبه کنیم.