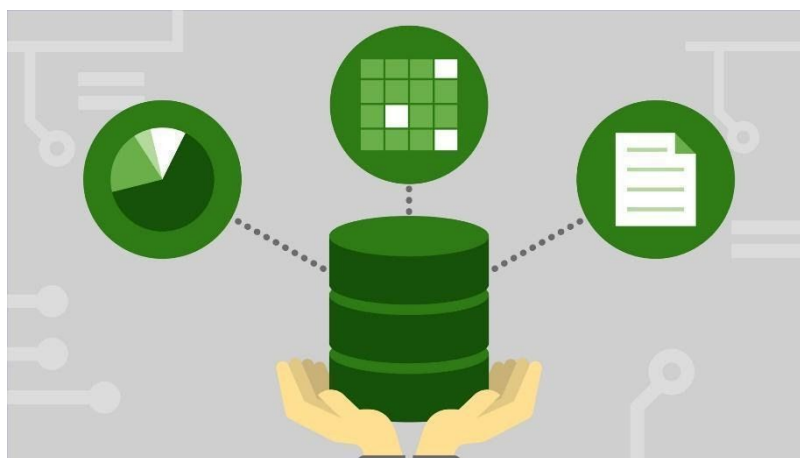


به نام خدا



دانشگاه تهران
پردیس دانشکده‌های فنی
دانشکده برق و کامپیوتر



آزمایشگاه پایگاه داده
دستور کار شماره ۵

شماره دانشجویی

۸۱۰۱۹۶۶۰۴

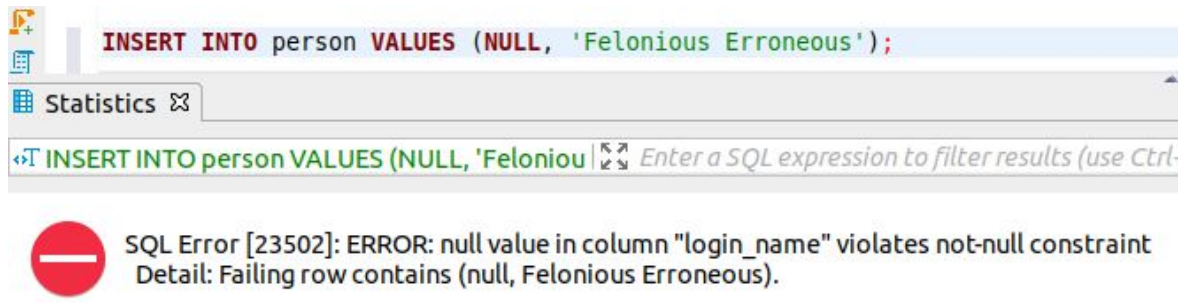
آبان ۹۹

نام و نام خانوادگی
سید پارسا حسینی نژاد

گزارش فعالیت‌های انجام شده

قسمت اول:

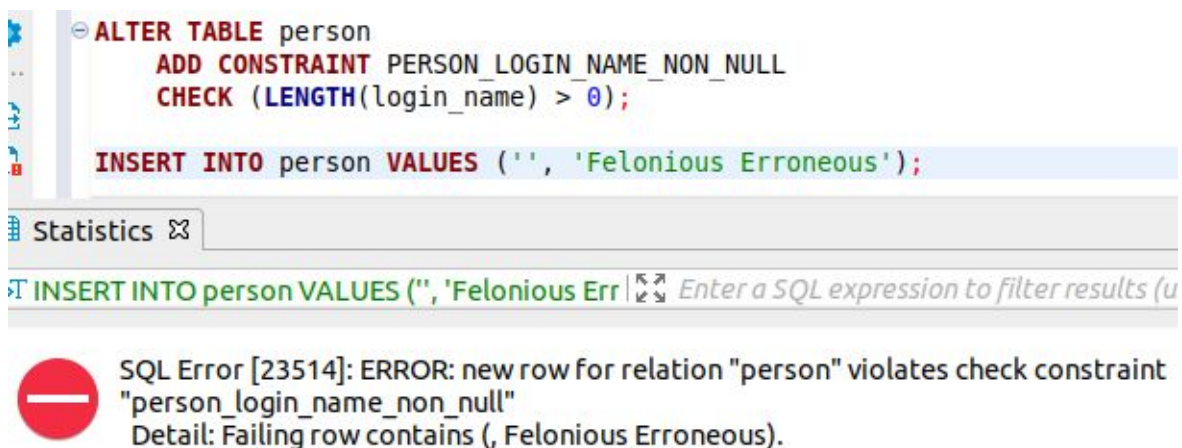
این عبارت با مشکل مواجه می‌شود چون در جدول person، فیلد login_name نمی‌تواند نال باشد.



این عبارت نیز با مشکل روبرو می‌شود چون حداکثر طول login_name برابر 9 است که در اینجا رعایت نشده است.



این عبارت با مشکل روبرو می‌شود چون یک constraint جدید وضع شده که طول login_name باید بیشتر از صفر باشد.




این عبارت با مشکل روبرو می‌شود چون یک constraint جدید وضع شده که نباید در login_name کاراکتر space وجود داشته باشد یا همان که position آن صفر باشد.

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

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

Statistics

Enter a SQL expression to filter results (use

 SQL Error [23514]: ERROR: new row for relation "person" violates check constraint "person_login_name_no_space"
Detail: Failing row contains (space man, Major Tom).

حال یک تابع تعریف می‌کنیم که شرط‌های دو عکس بالا را چک می‌کند و در صورت خطا در آن‌ها یک exception می‌دهد. برای مثال در این عکس، طول login_name باید از صفر بزرگتر 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;
$$;

INSERT INTO person VALUES ('', 'Felonious Erroneous');
```

Statistics

Enter a SQL expression to filter results (use

 SQL Error [P0001]: ERROR: Login name must not be empty.
Where: PL/pgSQL function person_bit() line 4 at RAISE

یا در این عکس، نباید در کاراکتر های login_name کاراکتر space وجود داشته باشد.

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

Statistics

INSERT INTO person VALUES ('space man', 'Mi' Enter a SQL expression to f



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

همانطور که مشاهده می‌شود، با اضافه کردن یک trigger که قبل insert تابع person_bit را انجام می‌دهد که آن تابع نیز در صورت وجود یک سری شرایط داده‌ها را به جدول person_audit نیز اضافه می‌کند که این امر قابل مشاهده است.

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

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

SELECT * FROM person;

SELECT * FROM person_audit;
```

person_audit

SELECT * FROM person_audit Enter a SQL expression to filter results (use Ctrl+Space)

	login_name	display_name	operation	effective_at	userid
1	dfunny	Doug Funny	INSERT	2020-11-18 19:29:41	postgres
2	pmayo	Patti Mayonnaise	INSERT	2020-11-18 19:30:01	postgres

این قسمت نیز مربوط به همان trigger قبل است که در صورت آپدیت نیز یک رکورد جدید به جدول person_audit اضافه می‌کند.

```
UPDATE person SET display_name = 'Doug Yancey Funny' WHERE login_name = 'dfunny';
SELECT * FROM person;
SELECT * FROM person_audit ORDER BY effective_at;
```

person_audit

SELECT * FROM person_audit ORDER BY effective_at

	login_name	display_name	operation	effective_at	userid
1	dfunny	Doug Funny	INSERT	2020-11-18 19:29:41	postgres
2	pmayo	Patti Mayonnaise	INSERT	2020-11-18 19:30:01	postgres
3	dfunny	Doug Yancey Funny	UPDATE	2020-11-18 19:50:51	postgres

در این قسمت نیز trigger مربوط به person_bdt فعال می‌شود که باعث insert شدن دوباره‌ی یک داده در جدول person_audit می‌شود.

```
DELETE FROM person WHERE login_name = 'pmayo';
SELECT * FROM person;
SELECT * FROM person_audit ORDER BY effective_at;
```

person_audit

SELECT * FROM person_audit ORDER BY effective_at

	login_name	display_name	operation	effective_at	userid
1	dfunny	Doug Funny	INSERT	2020-11-18 19:29:41	postgres
2	pmayo	Patti Mayonnaise	INSERT	2020-11-18 19:30:01	postgres
3	dfunny	Doug Yancey Funny	UPDATE	2020-11-18 19:50:51	postgres
4	pmayo	Patti Mayonnaise	DELETE	2020-11-18 19:54:45	postgres

```
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();
```

همانطور که مشاهده می‌شود با update شدن trigger مربوط به آپدیت اجرا شده و تابع person_bit نیز اجرا می‌شود. حال این تابع به جدول person مقدار ts_abstract را نیز اضافه می‌کند که از جنس TSVECTOR است که برای سرچ در متن ها به کار میرود و مانند یک آرایه است.

```
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) year old
SELECT login_name, ts_abstract FROM person;
```

person

SELECT login_name, ts_abstract FROM person

	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

فعال شدن این trigger به اضافه شدن یک داده به جدول person_audit نیز منجر می شود (مطابق عکس قسمت پیش و تابع person_bit)

```
SELECT login_name, display_name, operation, userid FROM person_audit ORDER BY effective_at;
```

person_audit

SELECT login_name, display_name, operation, | Enter a SQL expression to filter results (use Ctrl+Space)

	login_name	display_name	operation	userid
1	dfunny	Doug Funny	INSERT	postgres
2	pmayo	Patti Mayonnaise	INSERT	postgres
3	dfunny	Doug Yancey Funny	UPDATE	postgres
4	pmayo	Patti Mayonnaise	DELETE	postgres
5	dfunny	Doug Yancey Funny	UPDATE	postgres
6	skeeter	Mosquito Valentine	INSERT	postgres

بعد اضافه کردن ستون balance و ست کردن مقدار دیفالت آن به صفر می بینیم که سطرهای جدول نیز مقدار balance برابر صفر دارند.

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

person

SELECT login_name, balance FROM person WHERE login_name = 'dfunny';

	login_name	balance
1	dfunny	0 ریال

ابتدا برای رفع ارور نشناختن \$ نوع پول را به دلار عوض می‌کنیم (خط اول). حال با insert کردن به جدول transaction، تریگر مربوطه اجرا شده و باعث اجرا شدن تابع transaction_bit می‌شود. این تابع پول credit و debit را چک می‌کند تا مثبت باشد. همچنین اگر تقریب این دو منفی شد، ارور می‌دهد. وگرنه، مقدار مقدار balance را با توجه به این دو عوض می‌کند.

برای این مثال نیز 2.000.000 دلار به حساب dfunny واریز شده است.

```
set lc_monetary to 'C';
INSERT INTO transaction (login_name, post_date, description, credit, debit) VALUES ('dfunny', '2018-01-11',
SELECT login_name, balance FROM person WHERE login_name = 'dfunny';
```

person

SELECT login_name, balance FROM person WHERE login_name = 'dfunny'

	login_name	balance
1	dfunny	\$2,000.00

```
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;
$;
```


اما در اینجا، چون میزان debit از میزان balance بیشتر است، این اتفاق رخ نداده و ارور رخ می‌دهد.

```
INSERT INTO transaction (login_name, post_date, description, credit, debit) VALUES ('dfunny',
```

person

```
INSERT INTO transaction (login_name, post_d
```

SQL Error [P0001]: ERROR: Insufficient funds: (dfunny,2018-01-17,"FOR:BGE PAYMENT ACH Withdrawal",,"\$2,780.52")
Where: PL/pgSQL function transaction_bit() line 27 at RAISE

در اینجا نیز استفاده‌ی این تابع نشان داده شده است که می‌تواند transaction های مالی را برای ما انجام دهد. برای مثال، ابتدا پول حساب برابر 2.000.000 بوده که دو باز از آن برداشت شده است.

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

person

```
SELECT login_name, balance FROM person WH
```

	login_name	balance
1	dfunny	\$2,000.00

```
INSERT INTO transaction (login name, post date, description, credit, debit) VALUES ('dfunny', '2018-01-17', 'FOR:BGE PAYMENT ACH Withdrawal', '$278.52', NULL);
SELECT login_name, balance FROM person WHERE login_name = 'dfunny';
```

person

```
SELECT login_name, balance FROM person WH
```

	login_name	balance
1	dfunny	\$1,721.48

```
INSERT INTO transaction (login name, post date, description, credit, debit) VALUES ('dfunny', '2018-01-23', 'FOR: ANNE ARUNDEL ONLINE PMT ACH Withdrawal', '$35.
SELECT login_name, balance FROM person WHERE login_name = 'dfunny';
```

person

```
SELECT login_name, balance FROM person WH
```

	login_name	balance
1	dfunny	\$1,686.19

حال می‌خواهیم امنیت جدول را به وجود آوریم تا کسی نتواند دستی آپدیت کند. برای این کار از rollback استفاده می‌کنیم. همانطور که مشاهده می‌شود بعد تغییر balance و rollback کردن آن مقدار balance تغییر نکرده است.

```
BEGIN;
UPDATE person SET balance = '1000000000.00';
SELECT login_name, balance FROM person WHERE login_name = 'dfunny';
```

person

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

	login_name	balance
1	dfunny	\$1,000,000,000.00

```
SELECT login_name, balance FROM person WHERE login_name = 'dfunny';
ROLLBACK;
```

person

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

	login_name	balance
1	dfunny	\$1,686.19

حال مشاهده می‌کنیم که با استفاده از view به تنهایی نیز این مهم قابل دسترسی نیست چون postgres امکان write کردن را به view ها می‌دهد.

```
BEGIN;
UPDATE abridged_person SET balance = '1000000000.00';
SELECT login_name, balance FROM abridged_person WHERE login_name = 'dfunny';
```

abridged_person

```
SELECT login_name, balance FROM abridged_person
```

	login_name	balance
1	dfunny	\$1,000,000,000.00

حال با استفاده از view، این کار را انجام می‌دهیم. یک trigger می‌سازیم که به جای آپدیت روی جدول، تابع abridged_person_iut را اجرا کند. این تابع مقدار سطر را بعد آپدیت تغییر نداده و برابر مقدار قبلی می‌گذارد. به این شیوه، مقدار جدید balance را برابر مقدار قبلی گذاشته و از update جلوگیری می‌کنیم.

```
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

```
SELECT login_name, balance FROM abridged_person
```

	login_name	balance
1	dfunny	\$1,686.19

قسمت دوم:

1. با این query تعداد order به ازای هر shipper را محاسبه می‌کنیم. از تابع پنجره‌ای نیز برای حساب کردن تعداد order های هر shipper استفاده می‌کنیم.

```
select distinct s.company_name, count(*)
over (partition by s.company_name)
from shippers s, orders o
where o.ship_via = s.shipper_id;
```

shippers

```
select distinct s.company_name, count(*) over
```

	ABC company_name	123 count
1	United Package	326
2	Speedy Express	249
3	Federal Shipping	255

2. در این قسمت با استفاده از تابع rank، رتبه‌ی هر غذا در category خودش بر اساس قیمتش را محاسبه می‌کنیم.

```
select distinct c.category_name, p.product_name, p.unit_price,
rank() over(partition by c.category_name order by p.unit_price desc)
from products p, categories c
where c.category_id = p.category_id and c.category_name = 'Seafood'
order by rank;
```

categories(+)

```
select distinct c.category_name, p.product_name
```

	ABC category_name	ABC product_name	123 unit_price	123 rank
1	Seafood	Carnarvon Tigers	62.5	1
2	Seafood	Ikura	31	2
3	Seafood	Gravad lax	26	3
4	Seafood	Nord-Ost Matjesherin	25.88999939	4
5	Seafood	Inlagd Sill	19	5
6	Seafood	Boston Crab Meat	18.39999962	6
7	Seafood	Röd Kaviar	15	7
8	Seafood	Escargots de Bourgogi	13.25	8
9	Seafood	Spegesild	12	9
10	Seafood	Jack's New England Cl	9.64999962	10
11	Seafood	Rogede sild	9.5	11
12	Seafood	Konbu	6	12

3. در این قسمت از percent_rank استفاده می‌کنیم که درصد سطر هایی که مقدار درون over برای آن ها کمتر است را می‌دهد. همانطور که مشاهده می‌شود، برای $\text{percent rank} = 100\%$ بیشترین مقدار total را داریم.

```
select od.order_id, od.quantity*od.unit_price*(1-od.discount) as total,
       percent_rank() over (order by od.quantity*od.unit_price*(1-od.discount))
from order_details od
order by percent_rank desc;
```

order_details

select od.order_id, od.quantity*od.unit_price | Enter a SQL expression to filter results (use Ctrl+Sp

	order_id	total	percent_rank
1	10,981	15,810	1
2	10,865	15,019.4999882206	0.9995357474
3	10,417	10,540.0001525879	0.9990714949
4	10,889	10,540	0.9986072423
5	10,897	9,903.2000732422	0.9981429898
6	10,353	8,432.0000906587	0.9976787372
7	10,424	8,263.3600888455	0.9972144847
8	10,817	7,905	0.9962859796
9	10,540	7,905	0.9962859796
10	10,816	7,509.7499941103	0.995821727
11	11,032	6,587.5	0.9953574745
12	10,479	6,324.0000915527	0.9944289694
13	10,372	6,324.0000915527	0.9944289694
14	11,017	6,050	0.9939647168
15	10,776	6,041.9999952614	0.9935004643
16	11,030	5,570.5500411987	0.9925719591
17	10,912	5,570.5500411987	0.9925719591
18	10,515	5,268.0001831055	0.9921077066
19	10,691	4,951.6000366211	0.991643454
20	10,993	4,642.1250343323	0.9911792015
21	10,510	4,456.440032959	0.9902506964
22	10,666	4,456.440032959	0.9902506964
23	11,072	4,322.5	0.9897864438