



دانشگاه صنعتی امیر کبیر
(پلی تکنیک تهران)

به نام خدا

آزمایشگاه پایگاه داده
جلسه پنجم
توابع، رویه‌ها و آغازگرها

توابع

- ▶ استفاده از توابع همانند زبان‌های برنامه نویسی کمک به عدم بازنویسی فرایندها می‌کند.
- ▶ در PostgreSQL استفاده از توابع باعث بوجود آمدن سرویس‌های آماده مختلفی می‌شود، که کاربر می‌تواند در هر زمانی از آن‌ها استفاده و پاسخ خود را دریافت کند. غالبا یک تابع شامل یک سری `statements` و `expressions` می‌باشد.
- ▶ در PostgreSQL تعداد زیادی از توابع به صورت آماده وجود دارد که مناسب انواع ساختارهای داده است. در این قسمت تمرکز بر روی توابع در PostgreSQL می‌باشد.
- ▶ علاوه بر توابعی که در خود PostgreSQL وجود دارد، کاربران می‌توانند تابع دلخواه خودشان را نیز با استفاده از زبان‌های مختلف برنامه نویسی و پرس و جو بنویسند.
- ▶ این زبان‌ها شامل `PL/Perl`, `PL/Python`, `PL/Tcl`, `PL/pgSQL`, `SQL`, `C` است. به غیر از زبان‌های `PL/pgSQL`, `SQL`, `C` که بصورت پیشفرض وجود دارد
- ▶ برای بقیه زبان‌ها باید ابزار مناسب آن‌ها در دیتابیس نصب کنید.

توابع

▶ بطور خاص می‌توان موارد زیر را به عنوان دلایل استفاده از پایگاه داده شمرد.

- پیاده سازی منطق‌های پیچیده در پایگاه داده
- به وجود آوردن اعمالی که پس از یا قبل از اجرا شدن کوئری‌ها با استفاده از trigger ها اعمال شوند.
- مرتب سازی و پاک سازی کد با کاهش کدهای تکراری

▶ بطور کلی می‌توان توابع را در سه نوع طبقه بندی کرد. بصورت پیشفرض نوع تابع در هنگام پیاده‌سازی بصورت volatile می‌باشد. و می‌توان دو نوع دیگر یعنی stable و immutable را برای تابع انتخاب کرد.

▶ انتخاب درست نوع تابع کمک شایانی در بهبود کارایی پایگاه داده می‌کند. در ادامه این سه نوع توضیح داده شده است.

▶ **Volatile** : این نوع تابع می‌تواند در هر بار فراخوانی موفق، مقدار خروجی متفاوتی را بوجود آورد. حتی با فرض این که مقدار ورودی‌های آن یکسان باشد. به عنوان مثال تابع random() به این صورت است.

▶ **Stable and immutable** : در این دو نوع تابع تضمین می‌شود که مقدار خروجی با مقادیر ورودی یکسان، یکتا باشد. در توابع stable این تضمین در یک محدوده می‌باشد ولی در immutable همه محدوده ها را دربر می‌گیرد. تابع round() مثالی از این دو نوع می‌باشد.

سوال: تابع now() از چه نوعی است؟

توابع

► توابع پایگاه داده را می‌توان از جنبه‌های دیگری نیز بررسی و دسته‌بندی کرد، که در ادامه در مورد هر کدام توضیحاتی آورده شده است.

► **توابع تجمعی (Aggregate Functions)** : این نوع از توابع بر روی مجموعه‌ای از داده‌ها عمل می‌کنند تنها یک مقدار را با محاسبه مقادیر یک ستون برمی‌گردانند. در لینک زیر لیستی از این توابع وجود دارد.

► <https://www.postgresql.org/docs/9.5/functions-aggregate.html>

► **توابع مقیاسی (Scalar Functions)** : این توابع برای هر کدام از رکوردها مقدار منحصر به فردی را برمی‌گردانند.

```
► SELECT  
  UPPER(s.ContactName)  
► FROM  
  Customers s
```

► در این مثال نام مشتریان به صورت حروف بزرگ نمایش داده می‌شود. تابع UPPER() بر روی تمام رکوردها عمل کرده و برای هر کدام از آن‌ها یک مقدار خروجی منحصر به فرد ایجاد کرده است.

توابع

سوال: خروجی پرس و جوی زیر چیست؟

```
▶ SELECT  
▶ SUBSTRING(ContactName, 1, CHARINDEX(' ', ContactName) - 1)  
▶ FROM  
▶ Customers
```

▶ **توابع رتبه بندی (Ranking functions)** : این نوع توابع برای هرکدام از سطرهای موجود در یک محدوده مقداری را به عنوان امتیازبندی آن سطر ارائه می کنند.

▶ **توابع تعریف شده توسط کاربر (User defined functions)** : پایگاه داده این اجازه را می دهد که کاربران بتوانند تابع تعریف کنند. برای این کار می توان هم از طریق محیط گرافیکی و هم از طریق یک پرس و جو انجام داد. با استفاده از قطعه کد زیر می توان یک تابع را پیاده سازی و ذخیره کرد.

```
create or replace function get_film_count(len_from int, len_to
int)
returns int
language 'plpgsql'
as $ $
  declare film_count integer;
begin
select
  count(*) into film_count
from
  film
where
  length between len_from
and len_to;
return film_count;
end;
$ $;
```

توابع تعریف شده توسط کاربر

- در خط اول بعد از create function می توان مشخص کرد که آیا این تابع باید جایگزین شود و یا خیر. و پس از آن نام تابع درج می شود.
- در قسمت بعدی باید مقادیر ورودی این تابع مشخص شود. این مقادیر می تواند خالی هم باشد.
- در خط دوم مشخص می شود که نوع خروجی این تابع چه چیزی می باشد.
- در ادامه باید مشخص شود که زبان مورد استفاده برای پیاده سازی این تابع چه چیزی می باشد.
- قسمت انتهای، بدنه اصلی تابع است که مقادیر تعریفی و منطق تابع در آن نوشته می شود.

توابع

► مقادیر ورودی و خروجی

► مقادیر ورودی در یک تابع می‌توانند سه نوع مختلف داشته باشند. که بصورت پیش فرض نوع اول برای مقادیری که نوعشان مشخص نشده است قرار می‌گیرد. و انواع آن‌ها به شرح زیر می‌باشد.

► IN : این نوع به معنای ورودی می‌باشد. در این نوع می‌توان یک مقداری را برای تابع ارسال کرد و درون تابع از آن استفاده کرد.

► OUT : این نوع درواقع خروجی تابع می‌باشد. و هنگام فراخوانی تابع لازم نیست مقدار دهی شود. هیچ‌گاه نمی‌توان به صورت همزمان آرگومان ورودی با این نوع داشته باشیم و هم مقدار خروجی (return) در بدنه تابع اجرا کنیم.

► INOUT : می‌تواند هم OUT و هم IN باشد.

- Create
- or **replace function** fun(inout n **int**)
- **returns int as \$ \$ begin** n := n + 1;
- **end;**
- \$ \$ language 'plpgsql';
- **SELECT**
- **public.func(20)**

	func	
	bigint	🔒
1		21

سوال: اگر در یک تابع تعداد آرگومان‌ها با نوع OUT یا INOUT بیشتر از یک باشد چه چیزی به عنوان خروجی تابع در نظر گرفته می‌شود؟

توابع

► نوع مقدار خروجی در ابتدا باید مشخص شود. نوع خروجی می‌تواند یک جدول باشد. یعنی پس از پایان یافتن اجرا، تابع به عنوان خروجی یک جدول برمی‌گرداند. به عنوان مثال خروجی تابع زیر یک جدول است.

```
CREATE OR REPLACE FUNCTION public.get_film4(IN p_pattern character varying)
  RETURNS TABLE(film_title character varying, film_release_year integer)
  LANGUAGE 'plpgsql'
```

```
AS $BODY$
begin
  return query
    select
      title,
      release_year
    from
      film
  where
    title ilike p_pattern;
end;
$BODY$;
```


توابع

► فراخوانی توابع

► برای فراخوانی توابع کافی است از نام آن تابع به همراه مقادیر ورودی آن استفاده کنید. مقادیر ورودی می-تواند به ۳ صورت به تابع ارسال شود.

- `SELECT public.func(10,20)//positional notation`
- `SELECT public.func(arg1=>10,arg2=>20)//named notation`
- `SELECT public.func(arg1=>10,20)//mixed notation`

► توجه : روش named notion زمانی مورد استفاده قرار می گیرد که تعداد مقادیر ورودی زیاد باشد.

► حذف تابع

► برای حذف تابع می توان از دستور Drop به شکل زیر استفاده کرد.

- `DROP function [if exists] function_name(argument_list) [cascade | restrict]`

► چند نکته در حذف توابع وجود دارد. همانطور که مشخص است برای حذف یک تابع علاوه بر نام تابع، باید آرگومان های ورودی آنرا نیز وارد کنید. دلیل این امر یکتا بودن توابع با داشتن نام و آرگومان های ورودی آنها می باشد که به آن امضای تابع گفته می شود.

نکته: در زمان حذف یا DROP در یک جدول عبارت `if exists` جلوی خطا در هنگام نبود چنین تابعی را می گیرد. تابع ممکن است به یک موجودیت دیگر نیز وابسته باشد و در هنگام حذف دچار مشکل شود مثل `trigger`، لذا برای حذف یک تابع به همراه همه موجودیت های وابسته می توان از `cascade` استفاده کرد.

تفاوت توابع و رویه‌های ذخیره شده در PostgreSQL

- ▶ اصلی ترین تفاوت بین توابع و رویه‌ها در خروجی است که تولید می‌کنند.
- ▶ در توابع همیشه انتظار می‌رود که یک خروجی داشته باشد، اما در رویه‌ها خروجی وجود ندارد. در واقع هدف اصلی رویه‌ها صرفاً انجام یک سری فعالیت و سپس پایان یافتن آن‌ها می‌باشد. در حالی که توابع قرار است پردازشی انجام داده و خروجی تولید کنند.
- ▶ در نسخه‌های قدیمی postgres، رویه‌ها همان توابعی بوده‌اند که صرفاً خروجی برای آن‌ها تعبیه نشده بود ولی در نسخه‌های جدید به صورت خاص به آن‌ها پرداخته شده است.
- ▶ تفاوت دیگر امکان ایجاد تراکنش‌ها در یک رویه می‌باشد.
- ▶ یکی از معایب توابع عدم امکان ایجاد تراکنش در آن‌ها بوده است، که در نسخه‌های جدیدتر این پایگاه داده امکان ایجاد رویه با پشتیبانی تراکنش‌ها به آن اضافه شده است.

رویه‌های ذخیره شده

- ▶ نحوه ایجاد یک رویه مشابه ایجاد یک تابع است با در نظر گرفتن تفاوت‌های آن
- ▶ البته امکان اجرای دستور `return` در یک رویه وجود دارد اما بدون مقدار و رویه در لحظه اجرای این دستور متوقف می‌شود.
- ▶ در مثال صفحه بعد پرس و جویی است که یک رویه را ایجاد می‌کند. هدف این رویه، روالی است که به واسطه آن مقداری پول از حسابی به حساب دیگری منتقل می‌شود.

رویه های ذخیره شده

```
create
or replace procedure transfer(
    sender int,
    receiver int,
    amount dec
) language plpgsql as $ $ begin -- subtracting the amount from the sender's account
update
    accounts
set
    balance = balance - amount
where
    id = sender;
-- adding the amount to the receiver's account
update
    accounts
set
    balance = balance + amount
where
    id = receiver;
commit;
end;
$ $
```

✓ برای اجرایی کردن این رویه باید آن را به شکل زیر فراخوانی کرد.

```
CALL public.transfer(1,2,1000);
```

حذف رویه‌ها

► طریقه حذف رویه‌ها همانند حذف توابع با دستور Drop می‌باشد.



```
drop procedure [if exists] procedure_name (argument_list)
[cascade | restrict]
```

آغازگرها (Trigger)

- ▶ آغازگرها در PostgreSQL توابعی هستند که هنگام رخدادی مثل اضافه کردن به جدول، بروزرسانی جدول و حذف از جدول، بصورت خودکار فراخوانی می شود و می توان مشخص کرد که قبل یا بعد از رخداد اجرا شود.
- ▶ آغازگر وابسته به یک جدول و رخدادهای آن است. برای ایجاد یک آغازگر ابتدا یک تابع تعریف می شود و سپس این تابع به یک جدول متصل می شود.
- ▶ بصورت کلی دو نوع statement-level و row-level تعریف می شوند. تفاوت این دو در تعداد اجرای آن ها می باشد.
- ▶ به طور مثال زمانی که یک دستور بروزرسانی برای یک جدول اجرا می شود، ممکن است تعداد زیادی از ردیف های جدول را دربر بگیرد، در این حالت اگر نوع آغازگر استفاده شده بصورت row-level باشد، آنگاه به تعدادی که از ردیف های جدول بروزرسانی شده اند اجرا می شود ولی اگر بصورت statement-level باشد فقط یکبار اجرا می شود.
- ▶ مهم ترین کارکرد آغازگرها نقش نظارتی آن ها بر روی جداول است، از آن جای که یک پایگاه داده ممکن است از طریق نرم افزار های مختلفی مورد استفاده قرار بگیرد، باید بتوان به گونه ای بر روی اعمالی که بر روی پایگاه داده انجام می شود نظارت داشت.
- ❖ بطور مثال یکی از نیازمندی های سیستم ممکن است وجود تاریخچه ای از تغییرات جداول باشد.
- ▶ یا این که با استفاده از آغازگرها می توان جامعیت اطلاعات درون پایگاه داده را تضمین کرد.
- ❖ بطور مثال پس از اضافه شدن یک ردیف در جدول سبد خرید، باید از موجودی آن کالا کم شود و یا تغییراتی بر روی جداول دیگر صورت بگیرد.

ایجاد آغازگرها

► ایجاد آغازگرها در دو مرحله انجام می‌شود. در مرحله اول باید یک تابع ایجاد شود و در مرحله دوم تابع ایجاد شده باید به جدول مورد نظر متصل شود.

► برای ایجاد تابع، باید دقت شود که هیچ مقدار ورودی نباید داشته باشد و همچنین مقدار خروجی آن باید از نوع trigger باشد.

► به عنوان مثال پرس و جوی زیر تابعی را ایجاد می‌کند که قرار است قبل از هر بروزرسانی مقدار last_update رکورد جدید را برابر تاریخ آن لحظه قرار دهد. این تابع در مرحله بعدی به جدول یا جدول هایی می‌تواند متصل شود.

```
CREATE OR REPLACE FUNCTION public.last_updated()  
    RETURNS trigger  
    LANGUAGE 'plpgsql'  
AS $BODY$  
BEGIN  
    NEW.last_update = CURRENT_TIMESTAMP;  
    RETURN NEW;  
END  
$BODY$;
```

ایجاد آغازگرها

- ▶ با این حال که این تابع نباید ورودی داشته باشد، اما یکسری اطلاعات در یک قالب مشخص به نام TriggerData در لحظه اجرای آن، به تابع داده می‌شود. این ساختار داده حاوی تعدادی متغیر محلی می‌باشد.
- ▶ بطور مثال old و new دو متغیر کاربردی آن است، که مقدار اول بیانگر ردیف اطلاعات قبل از تغییر و مقدار دوم پس از آن می‌باشد.
- ▶ متغیرهای دیگری نیز وجود دارد که نام آن‌ها با عبارت TG_ شروع می‌شود. از قبیل TG_WHEN و TG_TABLE_NAME.
- ▶ برای درک کاربرد این متغیرها کافی است به این نکته دقت کنیم که یک تابع ممکن است برای تعداد زیادی جدول و یا تعداد زیادی رخداد اجرا شود، در نتیجه لازم است در مواردی بدانیم که این کدام جدول و یا کدام رخداد بوده است.
- ▶ حال پس از ایجاد تابع می‌توان یک آغازگر ایجاد کرد و آن را به تابع موردنظر متصل کرد. برای این کار از ساختار پرس و جوی زیر استفاده می‌کنیم.

ایجاد آغازگرها

- ▶ **CREATE TRIGGER** trigger_name
- ▶ {BEFORE | **AFTER**} { event }
- ▶ **ON** table_name
- ▶ [**FOR** [EACH] { ROW | STATEMENT }]
- ▶ **EXECUTE PROCEDURE** trigger_function

▶ همانطور که مشخص است، پس از نام آغازگر باید مشخص شود که قبل یا بعد از چه رخدادی باید این آغازگر فعال شود. بطور مثال می توان گفت که بعد از هر insert در جدول table_name. در انتها هم می توان نوع آغازگر را که در ابتدا توضیح داده شد مشخص کرد.

حذف آغازگرها

► برای حذف و یا تغییر یک آغازگر می‌توان از Drop و Alter همانند تابع استفاده کرد.

► **DROP TRIGGER** [IF EXISTS] **name ON table** [**CASCADE** | **RESTRICT**]

► **ALTER TRIGGER** trigger_name
► **ON** table_name
► **RENAME TO** new_trigger_name;

تمرین

۱. تابعی بنویسید که یک عدد ۱۱ رقمی به عنوان ورودی دریافت کند و تشخیص دهد مربوط به شماره تلفن همراه است یا شماره تلفن ثابت یا هیچکدام. در صورتی که شماره مربوط به تلفن همراه است عبارت "mobile phone number" را خروجی دهد. در غیر اینصورت ۳ رقم مربوط به کد شهر و ۸ رقم دیگر شماره را بصورت مجزا چاپ کند و مشخص کند شماره مربوط به اصفهان است یا تهران. در صورتی که خروجی مربوط به هیچ یک از موارد نبود عبارت "invalid phone number" را در خروجی نشان دهد.

▶ مثال:

خروجی: mobile phone number	09135479645
خروجی: city code =031 ,last 8 digits=22222222 city =Esfahan	03122222222
خروجی: invalid phone number	33333333333

تمرین

۲. تابعی بنویسید که دو تاریخ مشخص به عنوان ورودی دریافت کند و فیلم هایی که در بین این دو تاریخ به اجاره رفته اند را خروجی دهد.

۳. با استفاده از تابع سوال قبل، تابع دیگری بنویسید که مشخصات مشتریانی را در خروجی دهد که در بین دو تاریخ داده شده فیلمی را اجاره کرده اند ولی هنوز تحویل نداده اند. (مقدار return_date در صورت عدم بازگشت NULL می باشد).

تمرین

۴. آغازگری برای جدول rental بنویسید که هنگام اضافه یا بروزرسانی هر کدام از ردیف های جدول، بررسی کند آیا زمان return_data با توجه به مقدار rental_duration در جدول film، معتبر است یا خیر (تاخیر داشته است یا خیر). به این صورت که اگر ردیفی اضافه یا تغییر کرد و مدت زمان اجاره فیلم بیشتر از مقدار تعیین شده در جدول film باشد یک ردیف در جدولی به نام Log ایجاد شود و مشخص شود که کدام مشتری، چند روز تاخیر داشته است. (جدول log را باید ایجاد کنید و حداقل id مشتری در آن وجود داشته باشد با تعداد روز های تاخیر)

۵. یک رویه بنویسید که دو مقدار ورودی دریافت کند و مقدار اول را از یکی از ستون های جدول برای تمام سطر ها کم کند. بطور مثال فرض کنید این رویه قرار است مقدار rental_duration در جدول film را برای هر ردیفی به یک مقدار افزایش دهد.