



آزمایشگاه پایگاه داده

جلسه چهارم
روال های ذخیره شده و توابع

محمود فرجی و محمد جواد آکوچکیان

سال تحصیلی ۱۴۰۱-۱۴۰۲



روال های ذخیره شده (Procedure Stored)

قطعه برنامه هایی هستند که عمدتاً از دستوراتی به زبان SQL-T تشکیل می شوند و می توانند هرگاه توسط کاربر فراخوانی می شوند، عملیات مورد نظر وی را بر روی اشیاء مختلف بانک اطلاعاتی انجام دهند .

مزایای استفاده از روال ها عبارت اند از:

- ۱- کدهای مربوط به دسترسی به داده ها در یک مکان متمرکز می شوند به جای آن که در بخش های مختلف نرم افزار کاربردی پراکنده باشند و بنابراین تغییر دادن و خطایابی آن ها ساده تر است.
- ۲- امکان استفاده ی مجدد از کد در بخش های مختلف نرم افزار کاربردی فراهم می شود.
- ۳- روال های ذخیره شده نسبت به دستورات SQL موردی معمول سریع تر و با کارایی مناسب تری انجام می شوند.
- ۴- بهبود امنیت



تعریف یک روال بدون پارامتر

شکل کلی آن به صورت زیر است:

```
CREATE PROCEDURE procedure_name  
AS <sql-statement(s)>
```

مثال:

```
create procedure firstSp  
as select * from tbl_sailor  
where sailor_rank>5
```

```
EXEC firstSP
```

برای اجرای یک procedure از execute و یا از exec استفاده میشود مانند مثال رو به رو:



تغییر روال های ذخیره شده

برای تغییر یک روال ذخیره شده می توان از دستور **ALTER PROCEDURE** استفاده کرد.
با استفاده از این دستور می توان کلیه ی اجزای یک روال ذخیره شده به جز نام آن را تغییر داد

```
Alter PROCEDURE procedure_name  
as ...
```

برای حذف یک روال ذخیره شده از دستور **DROP PROCEDURE** استفاده میشود.



تعریف روال پارامتر دار

شکل کلی تعریف یک روال پارامتر دار به صورت زیر است:

```
CREATE PROCEDURE procedure_name  
@param_name data_type [=default_value] [OUTPUT][,...]  
AS <sql-statement(s)>
```

برای تعریف هر پارامتر باید نام و نوع آن پارامتر مشخص شود.
پارامتر هایی که می توان تعریف کرد دارای دو نوع هستند:

۱- پارامتر ورودی

۲- پارامتر خروجی (این نوع پارامتر ها با کلمه **OUTPUT** مشخص میشوند.)



روال پارامتر دار

مثال : روال رو به پارامتری از نوع رشته ای دریافت کرده و ملوانی با آن نام را حذف می کند:

```
create procedure secondSP  
@temp varchar(20)  
as delete from tbl_sailor  
where sailor_name=@temp
```

```
EXEC secondSP 'ali'  
  
EXEC secondSP @temp = 'ali'
```

نمونه هایی از فراخوانی این روال:



نحوه تعریف روال های ذخیره شده با پارامتر خروجی

در این مثال نحوه به کارگیری پارامتر های خروجی نشان داده شده است. روال مورد نظر تعداد ملوانانی که رنک بیشتر و یا مساوی رنک داده شده دارند را بر میگرداند.

```
create procedure thirdSP
@inRank int, @countSailor int output
as
select @countSailor=count(*)
from tbl_sailor
where sailor_rank>=@inRank
```

برای اجرای این روال نیز به این صورت عمل میکنیم:

```
declare @res int
exec thirdSP
@inRank=5,
@countSailor=@res output
print @res
```

```
declare @res int
exec thirdSP 5 , @res output
print @res
```




تعریف تابع (function)

فرمت کلی تعریف تابع به صورت زیر است:

Create function نام تابع مورد نظر

(فهرست پارامترها)

Return تایپ خروجی

As

begin

لیست دستورات T-SQL

end



مثال برای تابع

```
create function sel (@id int)
returns int
as
begin
    declare @num int;
    select @num= BSD from book where ID=@id
    return @num;
end
```

```
declare @return int;
exec @return=sel @id=2
print @return;
```

id	bookname	BSD
1	war and peace	22
2	general	13
3	red and black	10

13



تفاوت های تابع با روال های ذخیره شده

- توابع باید حتما یک مقدار بازگشتی داشته باشند اما روال ها میتوانند مقدار بازگشتی داشته باشند یا نداشته باشند.
- توابع تنها میتوانند مقادیر ورودی را به عنوان ورودی دریافت کنند و مقدار نهایی را بازگردانند اما روال ها هم متغیر های ورودی و هم متغیر های خروجی را میتوانند به عنوان ورودی دریافت کنند.
- توابع میتوانند درون روال ها فراخوانی شوند اما روال ها نمیتوانند درون توابع فراخوانی شوند.



معرفی برخی توابع

زبان T-SQL از برخی توابع پشتیبانی میکند که کار را راحت تر می کنند:

توابع تاریخ و زمان:

- **DATEDIFF()** : این تابع برای مقایسه دو تاریخ با یکدیگر مقایسه میشود و فرمت کلی استفاده از آن به صورت زیر است:

```
DATEDIFF(datepart,firstdate,seconddate)
```

در این تابع **datepart** معیار مقایسه دو تاریخ است که میتواند **ss**(ثانیه)، **mi**(دقیقه)، **hh**(ساعت)، **dd**(روز)، **dw**(هفته)، **mm**(ماه) و **yy**(سال) باشد.

- **GETDATE** : تاریخ فعلی سیستم را باز میگرداند.



معرفی برخی توابع

توابع رنک دهی (Ranking Function): توابعی وجود دارند که می توانیم از آنها برای رنک دهی استفاده کنیم. این توابع متفاوت بوده و نحوه رنک دهی آنها متفاوت است.

- **ROW_NUMBER()**: به ردیف های خروجی به ترتیب رتبه میدهد.
- **RANK()**: این تابع اگرچه به ردیف های یکسان، رنک یکسان میدهد اما شمارنده در پس زمینه همچنان در حال شمارش است و برای ردیف جدید (با مقدار جدید)، رنکی که حاوی مقدار شمارنده است ثبت میشود.
- **DENSE_RANK()**: مشابه با تابع RANK() است با این تفاوت که شمارنده درونی در هنگام رنک دهی به ردیف های یکسان، متوقف میشود



توابع رنک دهی



الگوی کلی توابع رنک دهی به صورت زیر است:

```
select *,  
ROW_NUMBER() over(order by boatName) as rowNumber,  
RANK() over(order by boatName) as rankNumber,  
DENSE_RANK() over(order by boatName) as denseRanke  
from boat
```



مقایسه توابع رنک دهی

	boatID	boatName	boatColor	boatRank	rowNumber	rankNumber	denseRanke
1	101	R1	red	6	1	1	1
2	110	R10	red	15	2	2	2
3	102	R2	green	7	3	3	3
4	103	R3	black	8	4	4	4
5	104	R4	graan	9	5	5	5
6	105	R5	red	10	6	6	6
7	111	R5	red	10	7	6	6
8	112	R5	red	10	8	6	6
9	106	R6	orange	11	9	9	7