

آزمایش چهارم

◀ توجه کنید: بخش‌هایی که با این علامت مشخص شده‌اند کارهایی است که باید در آزمایشگاه انجام و تحویل دهید.

جداول مجازی (View ها)

یکی از ویژگیهای ساختار بانک اطلاعات استقلال دید طراح و دید کاربر بانک اطلاعات است. یعنی الزامی نیست که آنچه طراح بر اساس دیدگاه فنی خود طراحی می کند دقیقا به همان شکل به کاربر نشان داده شود. بلکه می توان یک دید مجازی برای کاربر ایجاد کرد.

- View ها (جداول مجازی) برخلاف Table ها (که واقعا روی دیسک هستند)، وجود خارجی ندارند.

- اما برای ما بظاهر و از نظر کاربردی، View ها دقیقا مانند جدولها هستند، پس می توانیم:

- آنها را باز کرده و داده های آنها را ببینیم.
- بر روی آنها عملیات جبر رابطه ای را انجام دهیم، مثلا یک دستور Select روی یک View بنویسیم یا آن را با یک جدول Join کنیم.

ساخت یک View

- هرگاه یک دستور Select که جدول حاصل آن برای ما زیاد مورد استفاده است، میتوانیم آن را بصورت یک View ذخیره کنیم تا در مواقع لزوم از آن استفاده کنیم.

- هنگام ذخیره View داده ها ذخیره نمی شوند. بلکه خود دستور Select است که ذخیره می شود.

- هنگام رجوع به یک View خود سیستم سریعا دستور مربوط به آن را اجرا کرده و حاصل Select را به ما نشان می دهد بطوریکه بنظر می رسد این داده ها در داخل View ذخیره شده اند.

- دستور ساخت یک View:

```
CREATE View viewname as  
Select ....
```

بعنوان مثال در بانک اطلاعات زیر که مربوط به یک سیستم آموزشی است، کاربر برای بسیاری از پرس و جوها نیاز دارد ۴ جدول موجود (یا تعدادی از آنها را پیوند طبیعی دهیم)، و بناچار Query های او سخت و طولانی می شود.

رابطه S				رابطه T			رابطه C			رابطه STC (جدول نمرات)			
<u>Id</u>	SName	SFamily	Field	<u>Id</u>	TName	TFamily	<u>Id</u>	CName	Units	<u>SId</u>	<u>TId</u>	<u>CId</u>	mark
84110	Ali	Ahmadi	Computer	1	Hadi	Hamidi	01	DB	3	84110	2	01	12
84120	Reza	Rezaei	Math	2	Karim	Hassani	02	OS	3	84110	1	03	8
84130	Hassan	Hasani	Computer	3	Ali	Omidi				84130	2	02	18

- بعنوان طراح وظیفه داریم حاصل پیوند طبیعی ۴ جدول را بصورت یک View ذخیره و در اختیار کاربر قرار دهیم تا در هنگام لزوم از آن استفاده کند:

Create View WholeData as

Select * From S,C,T,STC

Where S.SId=STC.SId AND C.Id=STC.CId AND T.Id=STC.tId

حال کاربر برای این پرس و جو: نام اساتیدی که درس OS را تاکنون تدریس کرده اند، براحتی می نویسد:

Select tname from WholeData Where cname = 'OS'

روال های ذخیره شده (Stored Procedures)

روال ذخیره شده عبارت است از پیمانه ای از کد که قابل استفاده ی مجدد است. به عبارت دیگر مجموعه ای از کد است که برخلاف توابع یک مقدار بر نمی گرداند. روال ها می توانند پارامتر نیز دریافت کنند. بعضی از فواید استفاده از روال ها عبارتند 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. اگر پرس و جوهای به صورت مکرر از طریق نرم افزار برای سرویس دهنده ی پایگاه داده ارسال شود، ترافیک زیادی در شبکه ایجاد می کند که استفاده از روال های ذخیره شده این ترافیک را کاهش می دهد.

۴. روال های ذخیره شده نسبت به دستورهای SQL موردی معمولاً سریع تر و با کارایی مناسب تری اجرا می شوند.

تعریف یک روال بدون پارامتر

شکل کلی تعریف روال ذخیره شده به صورت زیر است:

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

که بعد از کلمه ی کلیدی AS دستور(های) T-SQL نوشته می شود. به عنوان مثال:

```
CREATE PROCEDURE uspCurrentDateOrders
AS
SELECT o.oid, c.cname
```

```
FROM ORDER1 AS o INNER JOIN CUSTOMER AS c ON o.cid=c.cid
WHERE DATEDIFF(day,o.oDate,GETDATE())=0
```

اجرای این دستورها یک روال با نام uspCurrentDateOrders تعریف می‌کند که می‌توانید آن را در پایگاه داده‌ی مورد نظر در بخش Programmability و تحت عنوان Stored Prosedures مشاهده کنید. برای اجرای این روال از دستور EXECUTE یا به‌صورت خلاصه EXEC استفاده می‌کنیم:

```
EXEC uspCurrentDateOrders
```

همان‌گونه که مشخص است اجرای این روال لیستی از شماره‌ی سفارش‌های انجام گرفته در روز جاری و نام مشتری آن‌ها را ارائه می‌کند.

تعریف روال پارامتردار

شکل کلی تعریف روال با پارامتر به‌صورت زیر است:

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

همان‌گونه که مشاهده می‌شود برای تعریف هر پارامتر باید نام و نوع آن را مشخص کرد. در ابتدای نام پارامتر باید علامت @ گذاشته شود. می‌توان مقدار پیش‌فرضی نیز (به‌صورت اختیاری) برای پارامتر تعریف کرد. پارامترها به دو دسته تقسیم می‌شوند: ورودی که با استفاده از آن‌ها اطلاعاتی برای روال ارسال می‌شوند و خروجی که اطلاعات حاصل از اجرای روال از طریق آن‌ها به فراخوان روال برگردانده می‌شود. این نوع پارامترها با کلمه‌ی OUTPUT مشخص می‌شوند. به‌عنوان مثال روال زیر پارامتری از نوع تاریخ دریافت می‌کند و لیست سفارش‌های آن تاریخ را می‌یابد:

```
CREATE PROCEDURE uspDesiredDateOrders
@ddate DATETIME
AS
SELECT o.oid, c.cname
FROM ORDER1 AS o INNER JOIN CUSTOMER AS c ON o.cid=c.cid
WHERE DATEDIFF(day,o.oDate,@ddate)=0
```

به‌هنگام فراخوانی روال باید به‌تعداد پارامترها مقادیر از نوع مناسب ارائه کرد. مثال:

```
EXEC uspDesiredDateOrders '10/10/2006'
```

یا به‌صورت زیر که نام پارامتر(ها) را نیز مشخص می‌کنیم:

```
EXEC uspDesiredDateOrders @ddate = '10/10/2006'
```

در مثال بعد نحوه‌ی به‌کارگیری پارامترهای خروجی نشان داده می‌شود. روال مورد نظر نام یک مشتری را به‌عنوان پارامتر ورودی دریافت می‌کند و تعداد سفارش‌های آن مشتری در سال جاری را با استفاده از پارامتر خروجی برمی‌گرداند.

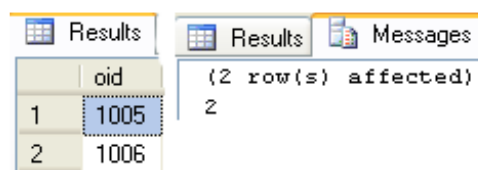
```
CREATE PROCEDURE uspCurrentYearNumberOfOrdersForCustomer
    @custName NCHAR(10), @orderNum INT OUTPUT
AS
    SELECT o.oid
    FROM ORDER1 AS o INNER JOIN CUSTOMER AS c ON o.cid=c.cid
    WHERE DATEDIFF(year,o.oDate,GETDATE())=0 AND
           c.cname = @custName

    SELECT @orderNum = @@ROWCOUNT
```

در این روال از تابع ROWCOUNT استفاده شده است. همان گونه که مشاهده می کنید نام توابع SQL Server با @@ آغاز می شوند. این تابع تعداد ردیف هایی را که آخرین دستورالعمل بر آنها تأثیر گذاشته است برمی گرداند. حاصل این تابع در پارامتر خروجی ذخیره می شود. مثالی از فراخوانی این روال در زیر آمده است:

```
DECLARE @res INT
EXEC uspCurrentYearNumberOfOrdersForCustomer 'cust2', @res OUTPUT
PRINT @res
```

همان گونه که مشاهده می شود برای ذخیره ی نتیجه، متغیری محلی تعریف شده است که حاصل پارامتر خروجی در آن قرار می گیرد. برای نشان دادن مقدار این متغیر از دستور PRINT استفاده شده است. نتیجه ی اجرا چیزی مشابه شکل ۱ است.



The screenshot shows two windows from SQL Server Enterprise Manager. The 'Results' window displays a table with two rows of order IDs. The 'Messages' window shows a message indicating that 2 rows were affected.

	oid
1	1005
2	1006

Messages: (2 row(s) affected)
2

شکل ۱: نتایج اجرای روال تعداد سفارش های سال جاری

تغییر روال های ذخیره شده

برای تغییر دادن یک روال ذخیره شده می توان از دستور ALTER PROCEDURE استفاده کرد. با استفاده از این دستور می توان تمام اجزای یک روال (غیر از نام آن) را تغییر داد. به عنوان مثال دستور زیر روال مثال قبل را به گونه ای تغییر می دهد که حاصل کار دو مجموعه ی جواب (لیست سفارش ها و تعداد آنها) باشد و پارامتر خروجی حذف می شود.

```
ALTER PROCEDURE uspCurrentYearNumberOfOrdersForCustomer
    @custName NCHAR(10)
AS
    SELECT o.oid
    FROM ORDER1 AS o INNER JOIN CUSTOMER AS c ON o.cid=c.cid
    WHERE DATEDIFF(year,o.oDate,GETDATE())=0 AND
           c.cname = @custName

    SELECT @@ROWCOUNT AS orderCount
```

نحوه ی فراخوانی مشابه زیر است:

```
EXEC uspCurrentYearNumberOfOrdersForCustomer 'cust2'
```

حاصل اجرا در شکل ۲ نشان داده شده است.

Results		Messages
oid		
1	1005	
2	1006	
orderCount		
1	2	

شکل ۲: نتایج اجرای روال تغییر یافته‌ی تعداد سفارش‌های سال جاری

حذف روال‌های ذخیره‌شده

برای حذف یک روال از دستور DROP PROCEDURE می‌توان استفاده کرد.

دستورهای پردازش شرطی در T-SQL

زبان T-SQL حاوی دستورهای شرطی برای انتخاب یک نتیجه با توجه به مقدار یک عبارت شرطی است. دو دستور مهم عبارت‌اند از CASE و IF.

دستور CASE

دستور یا تابع CASE با توجه به مقدار یک عبارت یک مقدار را برمی‌گرداند. شکل کلی این دستور به‌صورت زیر است:

```
CASE input_expression
  WHEN when-exp1 THEN result_exp1
  WHEN when-exp2 THEN result_exp2
  ...
  WHEN when-expn THEN result_expn
  [ELSE else_result_exp]
END
```

که در آن input_expression مقداری است که باید توسط دستور CASE ارزیابی شود و اگر مقدار آن با هریک از when_exp برابر باشد، result_exp معادل آن به‌عنوان نتیجه بازگردانده می‌شود. اگر با هیچ مقداری برابر نباشد، else_result_exp حاصل دستور خواهد بود (البته ELSE اختیاری است).

به‌عنوان مثال فرض کنید که در جدول CUSTOMER فیلدی عددی به‌نام ctype اضافه کنیم. حال می‌خواهیم در لیستی از مشتریان با توجه به مقدار این فیلد نوع مشتری را مشخص کنیم. پرس‌وجوی T-SQL زیر چنین کاری انجام می‌دهد:

```
SELECT cid, cname,
CASE ctype
  WHEN 1 THEN 'normal'
  WHEN 2 THEN 'special'
  ELSE 'unknown'
END AS customertype
FROM CUSTOMER
```

حاصل اجرای این پرس و جو مشابه شکل ۳ است.

	cid	cname	customertype
1	100	cust1	normal
2	101	cust2	special
3	102	cust3	unknown

شکل ۳: پرس و جویی با استفاده از CASE

عبارت بعد از WHEN می تواند عبارتی منطقی هم باشد که در آن مقدار یک یا چند فیلد بررسی می شود. در این حالت بعد از CASE چیزی نوشته نمی شود. مثال قبل را با این شکل دیگر نیز می توان نوشت:

```
SELECT cid, cname,
CASE
    WHEN ctype=1 THEN 'normal'
    WHEN ctype=2 THEN 'special'
    ELSE 'unknown'
END AS customertype
FROM CUSTOMER
```

دستور IF

این دستور در صورت درست بودن مقدار یک عبارت منطقی، یک یا چند دستور sql را اجرا می کند. در صورتی که قسمت ELSE هم وجود داشته باشد، غلط بودن شرط باعث اجرای بلوک دستور(های) بعد از ELSE می شود. شکل کلی این دستور به صورت زیر است:

```
IF boolean_expression
    sql_statement | statement_block
[ELSE
    sql_statement | statement_block]
```

به عنوان مثال روال زیر پارامتری رشته ای دریافت می کند و با توجه به مقدار آن تعدادی مشتری را که کم ترین یا بیش ترین تعداد سفارش ها را داشته اند لیست می کند. این تعداد را پارامتر دوم مشخص می کند.

```
CREATE PROCEDURE uspTestIF
    @hilo CHAR(2), @num INT
AS
    IF @hilo='LO'
    BEGIN
        SELECT TOP (@num)
            c.cname AS customerName, COUNT(*) AS totalOrders
        FROM CUSTOMER AS c INNER JOIN ORDER1 AS o ON o.cid=c.cid
        GROUP BY c.cname
        ORDER BY totalOrders
    END
    ELSE
    BEGIN
        SELECT TOP (@num)
            c.cname AS customerName, COUNT(*) AS totalOrders
        FROM CUSTOMER AS c INNER JOIN ORDER1 AS o ON o.cid=c.cid
        GROUP BY c.cname
        ORDER BY totalOrders DESC
    END
```

فراخوانی زیر را در نظر بگیرید. این فراخوانی دو مشتری با بیشترین تعداد سفارش را مشخص می‌کند:

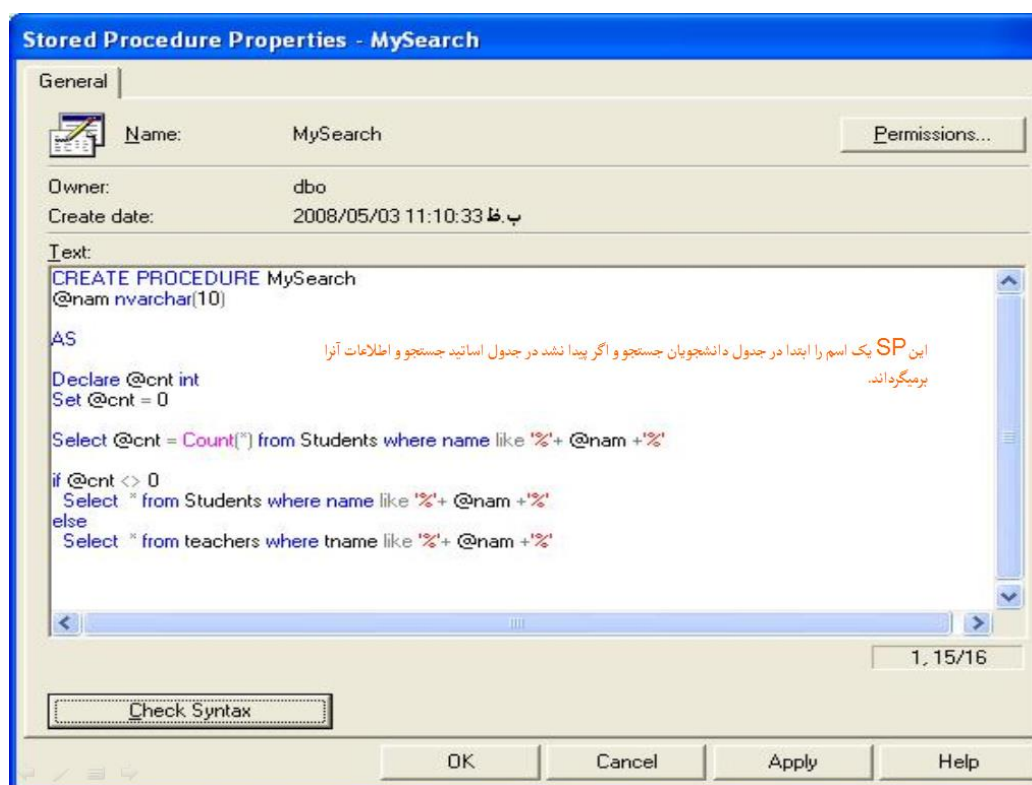
```
EXEC uspTestIF 'HI', 2
```

نتیجه‌ی اجرا مشابه شکل چهار است.

	customerName	totalOrders
1	cust1	4
2	cust2	3

شکل ۴: نتیجه‌ی پرس‌وجویی با کمک دستور IF

مثالی دیگر:



◀ مسایل خواسته شده را با استفاده از پایگاه داده‌ی آزمایشهای قبل حل کنید.
 ۱. یک view بنام `inbox_Complete` تعریف کنید بطوریکه همان لیست کل پیامها باشد با این تفاوت که بجای نمایش تنها کد کاربر فرستنده پیام، نام و نام خانوادگی او را نمایش دهد.

۲. یک view بنام `sentbox_Complete` تعریف کنید بطوریکه همان لیست کل پیامها باشد با این تفاوت که بجای نمایش تنها کد کاربر گیرنده پیام، نام و نام خانوادگی او را نمایش دهد.
۳. روالی بنویسید که کد یک کاربر خاص را بعنوان پارامتر دریافت کند و لیست `inbox` او را نمایش دهد. (بازیابی از جدول `inbox_Complete`)
۴. روالی بنویسید که کد یک کاربر خاص را بعنوان پارامتر دریافت کند و لیست `sent` او را نمایش دهد. (بازیابی از جدول `sentbox_Complete`)
۵. روالی بنویسید که کد یک کاربر خاص را بعنوان پارامتر دریافت کند و لیست پیامهای حذف شده از `inbox` او (`trash`) را نمایش دهد. (بازیابی از جدول `inbox_Complete`)