شروع کار با SQL Server

درس هشتم: برخی قابلیت‌های Transact SQL

سید کاوه احمدی

تعریف متغیر

- به منظور جلوگیری از شباهت نام فیلد و نام متغیر در SQL Server کلیه متغیرها با حرف @ معرفی می‌گردند.
- DECLARE @نام متغیر نوع متغیر [, ...]
  - تعریف متغیر x از نوع TinyInt:
    - DECLARE @x TinyInt

مقداردهی به متغیرها

- SET @نام متغیر = عبارت
- SELECT @نام متغیر = عبارت [, ...]
  - به خاطر مشابهت به دستور Select، استفاده از Set مرسوم‌تر است.
    - SET @x = 14
    - SET @x = @x + 2 * @y

تعریف و مقداردهی متغیرها

```
DECLARE @Now datetime= GETDATE(), @NewsDate datetime;
SELECT @NewsDate = Date FROM News WHERE ID = 5;

SELECT @Now, @NewsDate;
```

مثال

- دو جدول زیر را برای ذخیره اطلاعات کارمندان و شماره‌های تماس مربوط به او در نظر بگیرید:

```
Employee(Id, Name, License)

EmployeePhones(Id, PhoneNumber, EmployeeId)
```

  - می‌خواهیم بلافاصله پس از درج اطلاعات کارمند، شماره‌های تماس او را نیز در پایگاه داده ذخیره نماییم.

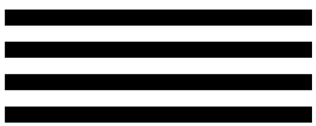
مثال

```
INSERT INTO
       Employee(Name, License)
VALUES
       ('emp name', 'license type');

DECLARE @employeeId int = @@IDENTITY;

INSERT INTO
       EmployeePhones (EmployeeId, PhoneNumber)
VALUES
       (@employeeId, '0912…');

INSERT INTO
       EmployeePhones (EmployeeId, PhoneNumber)
VALUES
       (@employeeId, '02122...');
```

بلوک دستورات

- یک بلوک از دستورات در T-SQL به شکل زیر مشخص می‌شود

BEGIN

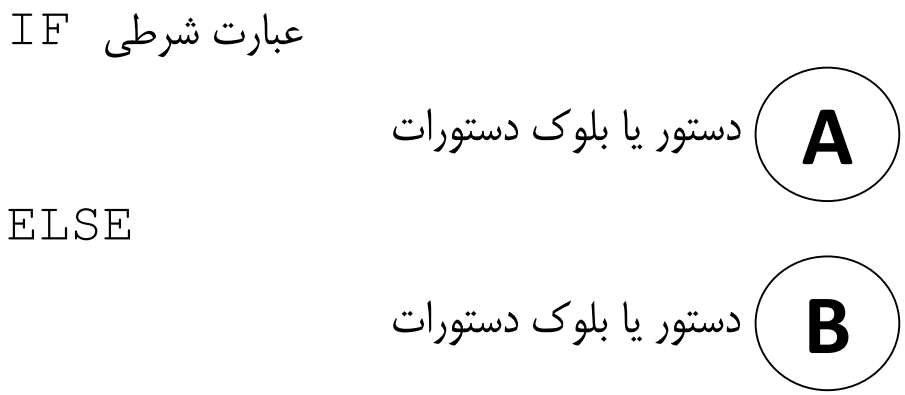


مجموعه دستورات
درون بلاک

END

دستورات شرطی

- در صورتیکه عبارت شرطی صحیح باشد دستورات بلوک A و در غیر این صورت دستورات بلوک B اجرا خواهد شد
- در صورتیکه نیازی به ساختار ELSE نباشد، می‌توان از این بخش صرف نظر کرد



دستورات شرطی

```
DECLARE @x Int, @y Int
SET @x = 200
SET @y= 300 - @x
IF @x > @y
     Set @y = @x + 2
ELSE
     BEGIN
          Set @y = @x + 3
          Set @x = @x + 1
     END
```

ساختار Case

- ساختار اول

```
CASE
        WHEN  عبارت شرطی اول  THEN
                نتیجه اول
        WHEN  عبارت شرطی دوم  THEN
                نتیجه دوم
        .
        .
        .
        [ELSE  نتیجه پیش‌فرض]
END
```

- ساختار دوم

```
CASE  عبارت
        WHEN  مقدار اول  THEN
                نتیجه اول
        WHEN  مقدار دوم  THEN
                نتیجه دوم
        .
        .
        .
        [ELSE  نتیجه پیش‌فرض]
END
```

ساختار Case

- ساختار اول

```
DECLARE @x INT
SET @x = (
   CASE
      WHEN @y=1 THEN 102
      WHEN @y=2 THEN 203
      WHEN @y=3 THEN 304
      ELSE 1000
   END
)
```

- ساختار دوم

```
DECLARE @x INT
SET @x = (
   CASE @y
      WHEN 1 THEN 102
      WHEN 2 THEN 203
      WHEN 3 THEN 304
      ELSE 1000
   END
)
```

ساختار Case

- باتوجه به اینکه ساختار اول از عبارت شرطی بهره می‌گیرد قدرت بیشتری را ارائه می‌دهد

```
DECLARE @x INT
SET @x =
        CASE
                WHEN    @y BETWEEN 2 AND 10     THEN 300
                WHEN    @y = 15 OR @y = 17      THEN 310
                WHEN    @y = 18 OR @y >= 20     THEN 320
                WHEN    @y + @z <= 30           THEN 330
                WHEN    Not @z <= 40            THEN @z
                ELSE    @y
        END
```

مثال

- در پایگاه داده NewsSystem، عنوان خبر و نام دسته آنرا درج کنید. در صورتی که یک خبر دسته‌بندی نداشته باشد، عنوان بدون دسته درج شود.

```
SELECT
	N.Title,
	CASE
		WHEN C.Name IS NULL THEN
			'بدون دسته'
		ELSE
			C.Name
	END AS CategoryName
FROM
	News N
LEFT OUTER JOIN
	Categories C
ON
	N.CategoryID = C.ID
```

تمرین

- پرسجویی بنویسید که عنوان خبر و فیلدی به نام Popularity را بازگرداند که:
  - اگر خبر بیش از ۱۰۰ نظر دارد مقدارش داغ
  - اگر بین ۵۰ تا ۱۰۰ نظر دارد مقدارش دارد داغ می شود
  - و اگر کمتر از ۵۰ نظر دارد مقدار NULL باشد.

تمرین

- ابتدا به پرسجویی نیاز داریم که عنوان خبر و تعداد نظرات مربوط به آن خبر را بازگرداند:

```
SELECT
	N.Title, COUNT(C.Id)
FROM
	News N
LEFT OUTER JOIN
	Comments C
ON
	N.Id = C.NewsID
GROUP BY
	N.Title
```

تمرین

- اکنون با استفاده از COUNT انجام شده می‌توان فیلد مورد نظر را ایجاد کرد:

```
SELECT
        N.Title,
        CASE
                WHEN COUNT(C.Id) > 100 THEN
                        N'داغ'
                WHEN COUNT(C.Id) BETWEEN 100 AND 50 THEN
                        N'داره داغ می شه'
                ELSE
                        NULL
        END AS Popularity
FROM
        News N
LEFT OUTER JOIN
        Comments C
ON
        N.Id = C.NewsID
GROUP BY
        N.Title
```

حلقه تکرار

حلقه تکرار

```
Declare @x Int, @y BigInt
Set @x = 0
Set @y = 1
While @x < 10
        Begin
                Set @x = @x + 1
                Set @y = @y * @x
        END
```

```
Declare @x Int, @y BigInt
Set @x = 0
Set @y = 1
While 1 = 1
        Begin
                Set @x = @x + 1
                IF  @x>= 10
                        Break
                Set @y = @y * @x
        END
```

مثال

- رشته را معکوس و به حروف کوچک تبدیل می‌کند.

```
Declare @x VarChar(50), @C TinyInt, @y VarChar(50)
Set @x = 'Hello'
Set @y = ''
Set @C = 0
While @C < Len(@x)
      Begin
            Set @C = @C + 1
            Set @y = Lower(SubString(@x , @C , 1)) + @y
      END
```

Cursor

- با استفاده از Cursorها می‌توان روی دسته‌ای از رکوردهای انتخاب شده مکررا عمل پیمایش را انجام داد. به این مفهوم که اگر در برنامه یا روال خود به یک مجموعه از رکوردها به طور مکرر نیاز دارید بهتر است به جای انتخاب مکرر آنها به وسیله پرسش‌های select یک بار آنها را از پایگاه داده خواند و در دفعات بعد به کمک یک Cursor آنها را پیمایش کنید.

Cursor

- مراحل استفاده از Cursor
  - ایجاد

```
DECLARE cursor_name CURSOR
FOR select_statment
```

  - باز کردن

```
OPEN cursor_name
```

  - پیمایش

```
FETCH [FIRST|LAST|NEXT|PRIOR]
FROM cursor_name INTO parameters
```

  - بستن

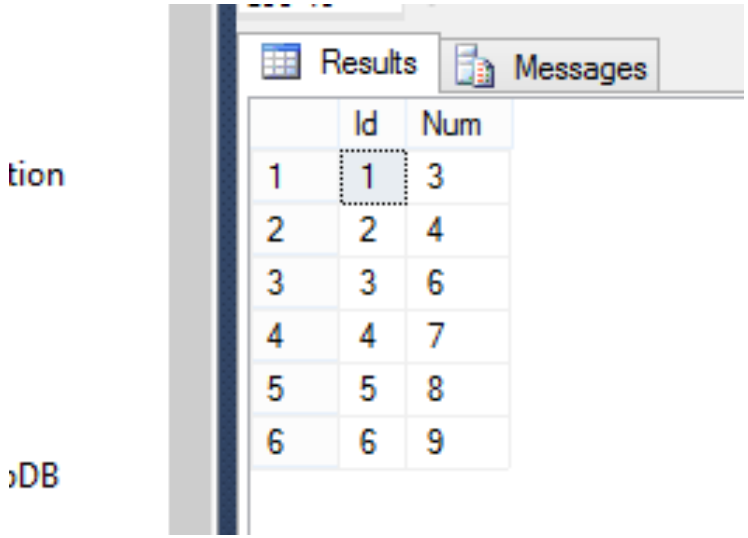
```
CLOSE cursor_name
DEALLOCATE cursor_name
```

Cursor

```
DECLARE @Name AS nvarchar(50)
DECLARE @Email AS nvarchar(50)

DECLARE AuthorCursor CURSOR FOR SELECT Name, Email FROM Authors

OPEN AuthorCursor
        FETCH NEXT FROM AuthorCursor INTO @Name, @Email

        WHILE @@FETCH_STATUS = 0
                BEGIN
                        PRINT 'Name: ' + @Name
                        PRINT 'Email: ' + @Email

                        FETCH NEXT FROM AuthorCursor INTO @Name, @Email
        END
CLOSE AuthorCursor

DEALLOCATE AuthorCursor
```

تمرین جمع تجمعی

- جدول زیر را در نظر بگیرید. می‌خواهیم در ستونی با عنوان `CumulativeSum`، برای هر رکورد، جمع مقادیر ثبت شده در فیلد `Num` از ابتدا تا آن رکورد را بر حسب ترتیب `Id`ها محاسبه کنیم:



تمرین جمع تجمعی

- روش اول: پیوند جدول با خودش

```
SELECT
        n1.Id,
        n1.Num,
        SUM(n2.Num) AS CumulativeSum
FROM
        Numbers n1
INNER JOIN
        Numbers n2
ON
        n2.Id <= n1.Id
GROUP BY
        n1.Id,
        n1.Num
```

هر رکورد را با تمامی رکوردهای با Id کوچکتر از خود پیوند می‌دهد:

تعداد رکوردهای حاصل:

```
[256,000 * (1+256,000)] / 2 = 32,768,128,000
```

روی این تعداد رکورد عمل GROUP BY و جمع انجام می‌شود!

زمان اجرا روی ۱۰۰۰۰ رکورد: ۹ ثانیه
زمان اجرا روی ۲۵۶۰۰۰ رکورد: ۲۵ دقیقه

تمرین جمع تجمعی

- روش دوم: استفاده از یک پرسش فرعی

```
SELECT
	Id,
	Num,
	(
		SELECT
			SUM(Num)
		FROM
			Numbers n2
		WHERE
			n2.Id <= n1.Id
	) AS CumulativeSum
FROM
	Numbers n1
```

زمان اجرا روی ۱۰۰۰۰ رکورد: ۵ ثانیه
زمان اجرا روی ۲۵۶۰۰۰ رکورد: ۱ دقیقه و ۳۰ ثانیه

تمرین جمع تجمعی

- روش سوم:
  - در دو روش قبل برای بدست آوردن هر مقدار، عمل جمع روی تمامی مقادیر Num با Id کوچکتر از Id سطر جاری باید انجام شود. این امر به این معنی است که محاسبات به شکل تکراری انجام می‌شود.
  - می‌خواهیم محاسبات را یکبار انجام دهیم و از مقادیری که قبلا مورد محاسبه قرار گرفته است مجددا استفاده کنیم.

تمرین جمع تجمعی

- روش سوم:
  - ایده این است که هر مقدار یکبار محاسبه شده، در یک جدول موقت ذخیره شود و برای محاسبه‌ی سایر مقادیر، از این جدول استفاده گردد.
  - ایجاد جدول موقت:
    - ایجاد جدول‌های موقت همانند جدول‌های عادی است با این تفاوت که نام آنها باید با # شروع شود.
    - این جدول‌ها تا زمان فعال بودن sessionی که جدول در آنها ایجاد شده است وجود خواهند داشت مگر آنکه پیش از آن DROP شوند!

```
IF OBJECT_ID('tempdb..#TMP') IS NOT NULL
        DROP TABLE #TMP
GO

CREATE TABLE #TMP (
        Id int,
        Num int,
        CumulativeSum int
);
```

تمرین جمع تجمعی

- روش سوم
  - مقادیر جدول با استفاده از یک کرسر خوانده شده و برای هر سطر محاسبه جمع تجمعی با استفاده از داده‌های جدول موقت محاسبه می‌شود.

```
DECLARE @Id int, @Num int, @CumulativeSum int

DECLARE rt_cursor CURSOR
  FOR SELECT Id, Num FROM Numbers ORDER BY Id

DECLARE @LastCumulativeSum int = 0

OPEN rt_cursor
  FETCH NEXT FROM rt_cursor INTO @Id, @Num

  WHILE @@FETCH_STATUS = 0 BEGIN
    SET @LastCumulativeSum = @Num + @LastCumulativeSum

    INSERT INTO #TMP VALUES(@Id, @Num, @LastCumulativeSum)

    FETCH NEXT FROM rt_cursor INTO @Id, @Num
  END
CLOSE rt_cursor
DEALLOCATE rt_cursor

SELECT * FROM #TMP
DROP Table #TMP
```

زمان اجرا روی ۱۰.۰۰۰ رکورد: کمتر از ۱ ثانیه
زمان اجرا روی ۲۵۶۰۰۰ رکورد: ۹ ثانیه