عنوان: آشنایی با SQL Server Common Table Expressions - CTE
نویسنده: عمران
تاریخ: ۱۳۹۲/۰۶/۰۷ ۱۴:۰
آدرس: www.dotnettips.info
گروه‌ها: SQL Server, T-SQL

مقدمه

تکنولوژی CTE از نسخه SQL Server 2005 رسمیت یافته است و شامل یک result set موقتی[1] است که دارای نام مشخص بوده و می-توان از آن در دستورات SELECT, INSERT, UPDATE, DELETEاستفاده کرد. همچنین از CTE می‌توان در دستور CREATE VIEW و دستور SELECT مربوط به آن استفاده کرد. در نسخه SQL Server 2008 نیز امکان استفاده از CTE در دستور MERGE فراهم شده است.
در SQL Serverاز دو نوع CTE بازگشتی[2] و غیر بازگشتی[3] پشتیبانی می-شود. در این مقاله سعی شده است نحوه تعریف و استفاده از هر دو نوع آن آموزش داده شود.

انواع روش-های ایجاد جداول موقت

برای استفاده از جداول موقتی در سرور اسکیوال، سه راه زیر وجود دارد.

روش اول: استفاده از دستوری مانند زیر است که سبب ایجاد جدول موقتی در بانک سیستمی tempdb می-شود. زمانی-که شما ارتباط خود را با سرور SQL قطع می-کنید به صورت اتوماتیک جداول موقت شما از بانک tempdb حذف می-شوند. این روش در برنامه نویسی پیشنهاد نمی-شود و فقط در کارهای موقتی و آزمایشی مناسب است.

```
SELECT * INTO #temptable FROM [Northwind].[dbo].[Products]
UPDATE #temptable SET [UnitPrice] = [UnitPrice] + 10
```

روش دوم: استفاده از متغیر نوع Table است، که نمونه آن در مثال زیر دیده می-شود. زمانیکه از محدوده[4] جاری کد[5] خودتان خارج شوید آن متغیر نیز از حافظه پاک میشود. از این روش، عموما در کدهای Stored Proceduresها و UserDefined Functionها استفاده می-شود.

```
DECLARE @tempTable TABLE
(
   [ProductID] [int] NOT NULL,
   [ProductName] [nvarchar](40) NOT NULL,
   [UnitPrice] [money] NULL
)

INSERT INTO @tempTable
SELECT
   [ProductID],
   [ProductName],
   [UnitPrice]
FROM [Northwind].[dbo].[Products]

UPDATE @temptable SET [UnitPrice] = [UnitPrice] + 10
```

روش سوم: استفاده از CTE است که مزیت-هایی نسبت به دو روش قبلی دارد و در بخش بعدی به نحوه تعریف و استفاده از آن خواهیم پرداخت.

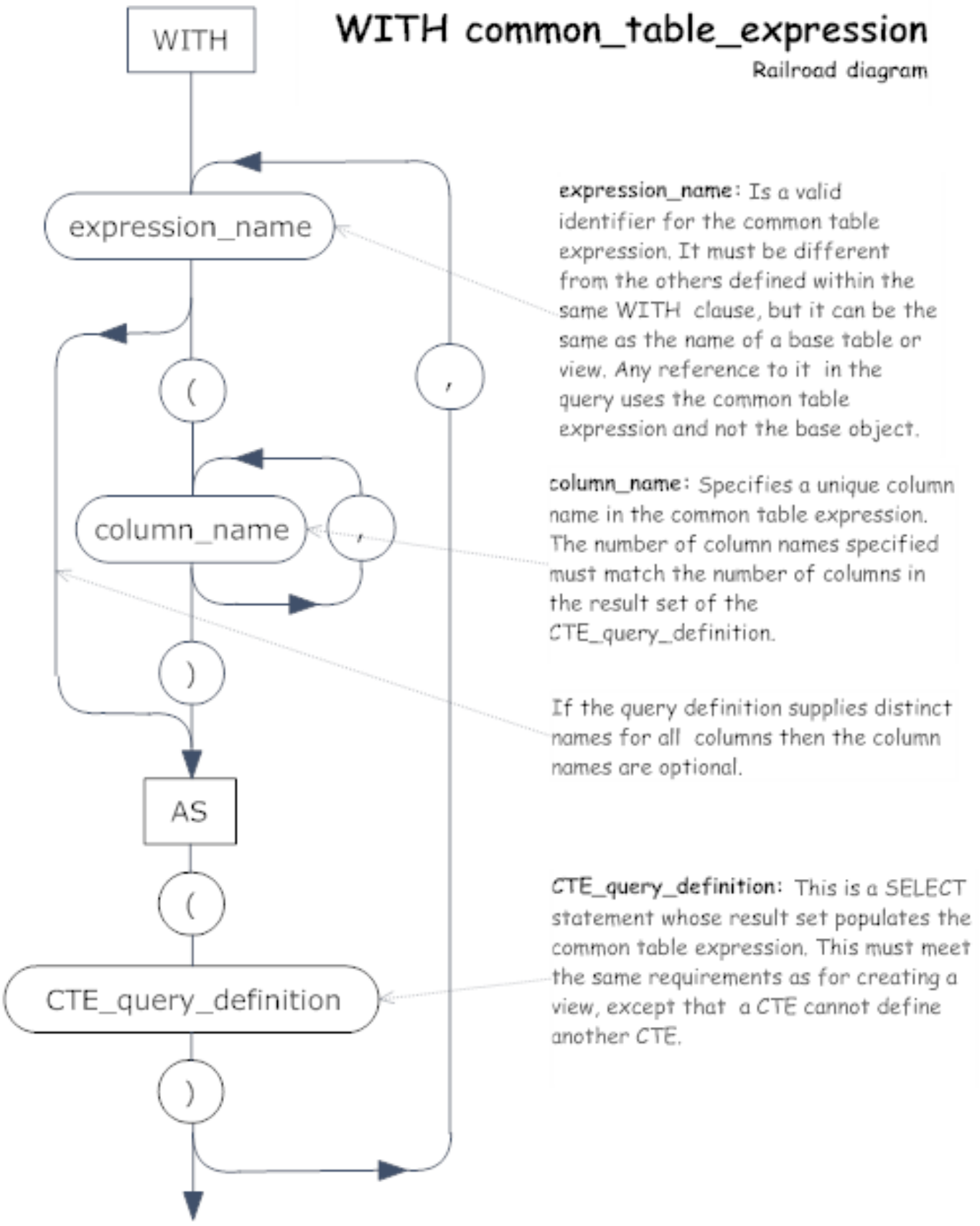
کار با CTE

ساده -ترین شکل تعریف یک CTE به صورت زیر است:

```
WITH yourName [(Column1, Column2, ...)]
AS
(
```

```
   your query
)
```

با کلمه WITH شروع شده و یک نام اختیاری به آن داده می-شود. سپس فهرست فیلدهای جدول موقت را درون زوج پرانتز، مشخص می-کنید. تعریف این فیلدها اختیاری است و اگر حذف شود، فیلدهای جدول موقت، مانند فیلدهای کوئری مربوطه خواهد بود.
your query شامل دستوری است که سبب تولید یک result set می-شود. قواعد تعریف این کوئری مشابه قواعد تعریف کوئری است که در دستور CREATE VIEW کاربرد دارد.



شکل کلی دستور

همانطور که از این تصویر مشخص است می-توان چندین بلوک از این ساختار را به دنبال هم تعریف نمود که با کاما از هم جدا می-شوند. در واقع یکی از کاربردهای CTE ایجاد قطعات کوچکی است که امکان استفاده مجدد را به شما داده و می-تواند سبب خواناتر شدن کدهای پیچیده شود.
یکی دیگر از کاربردهای CTE آنجایی است که شما نمی‌خواهید یک شی Viewی عمومی تعریف کنید و در عین حال می‌خواهید از مزایای Viewها بهرمند شوید.
و همچنین از کاربردهای دیگر CTE تعریف جدول موقت و استفاده از آن جدول به صورت همزمان در یک دستور است.

بعد از آنکه CTE یا CTEهای خودتان را تعریف کردید آنگاه می-توانید مانند جداول معمولی از آنها استفاده کنید. استفاده از این جداول توسط دستوری خواهد بود که دقیقا بعد از تعریف CTE نوشته می-شود.

ایجاد یک CTE غیر بازگشتی[6]

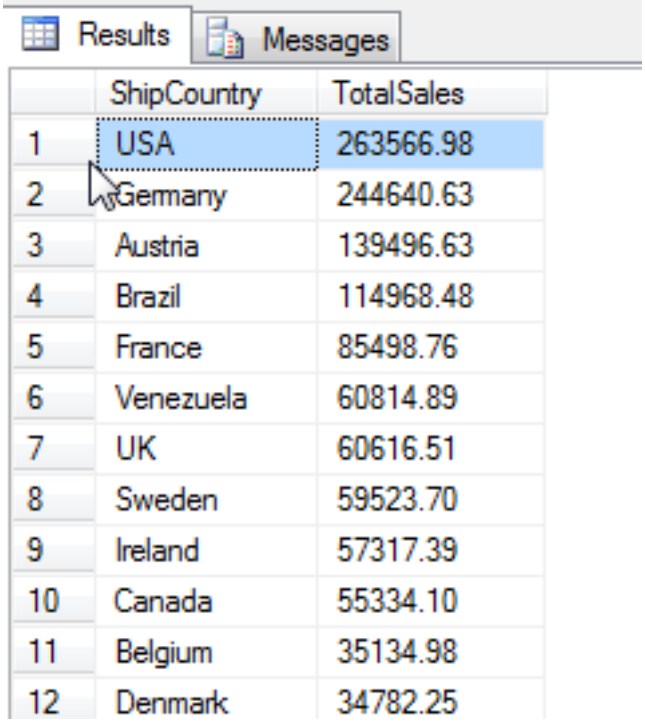
مثال اول، یک CTE غیر بازگشتی ساده را نشان میدهد.

```
WITH temp
AS
(
    SELECT
         [ProductName],
         [UnitPrice]
    FROM [Northwind].[dbo].[Products]
)
SELECT * FROM temp
ORDER BY [UnitPrice] DESC
```

مثال دوم نمونه-ای دیگر از یک CTE غیر بازگشتی است.

```
WITH orderSales (OrderID, Total)
AS
(
   SELECT
      [OrderID],
     SUM([UnitPrice]*[Quantity]) AS Total
   FROM [Northwind].[dbo].[Order Details]
   GROUP BY [OrderID]
)
SELECT
   O.[ShipCountry],
   SUM(OS.[Total]) AS TotalSales
FROM [Northwind].[dbo].[Orders] AS O INNER JOIN [orderSales] AS OS
ON O.[OrderID] = OS.[OrderID]
GROUP BY O.[ShipCountry]
ORDER BY TotalSales DESC
```

هدف این کوئری، محاسبه کل میزان فروش کالاها، به ازای هر کشور می-باشد. ابتدا از جدول Order Details مجموع فروش هر سفارش محاسبه شده و نتیجه آن در یک CTE به نام orderSales قرار می-گیرد و از JOIN این جدول موقت با جدول Orders محاسبه نهایی انجام شده و نتیجه-ای مانند این تصویر حاصل می‌شود.



Results | Messages

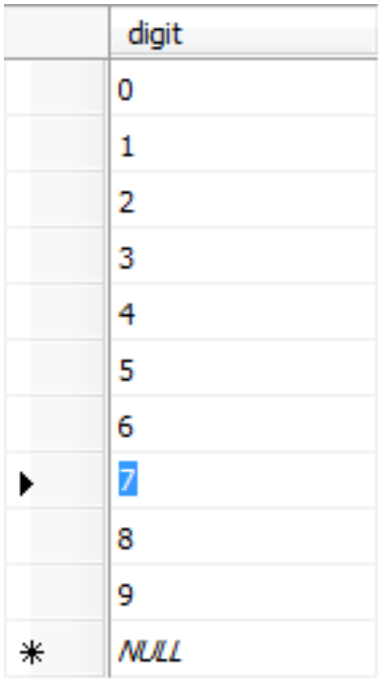
	ShipCountry	TotalSales
1	USA	263566.98
2	Germany	244640.63
3	Austria	139496.63
4	Brazil	114968.48
5	France	85498.76
6	Venezuela	60814.89
7	UK	60616.51
8	Sweden	59523.70
9	Ireland	57317.39
10	Canada	55334.10
11	Belgium	35134.98
12	Denmark	34782.25

نتیجه خروجی

مثال سوم استفاده از دو CTE را به صورت همزمان نشان می-دهد:

```
WITH customerList
AS
(
   SELECT
      [CustomerID],
      [ContactName]
   FROM [Northwind].[dbo].[Customers]
   WHERE [Country] ='UK'
)
,orderList
AS
(
   SELECT
      [CustomerID],
      [OrderDate]
   FROM [Northwind].[dbo].[Orders]
   WHERE YEAR([OrderDate])< 2000
)
SELECT
   cl.[ContactName],
   YEAR(ol.[OrderDate]) AS SalesYear
FROM customerList AS cl JOIN orderList AS ol
ON cl.[CustomerID] = ol.[CustomerID]
```

مثال چهارم استفاده مجدد از یک CTE را نشان می-دهد. فرض کنید جدولی به نام digits داریم که فقط یک فیلد digit دارد و دارای 10 رکورد با مقادیر 0 تا 9 است. مانند تصویر زیر



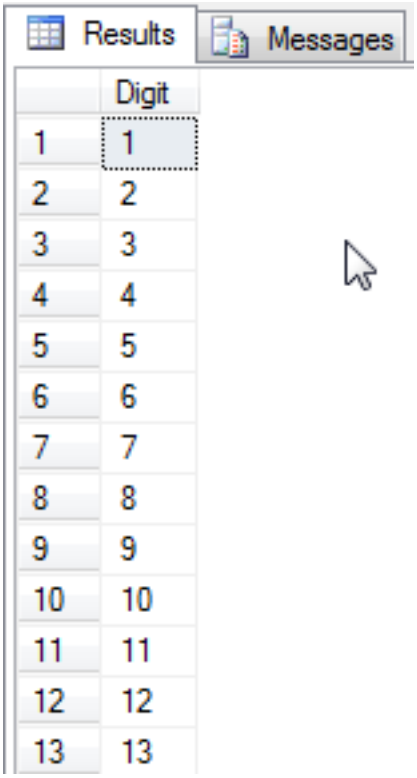
	digit
	0
	1
	2
	3
	4
	5
	6
▶	7
	8
	9
*	NULL

نتیجه خروجی

حال می-خواهیم از طریق CROSS JOIN اعداد 1 تا 100 را با استفاده از مقادیر این جدول تولید کنیم. کد زیر آنرا نشان می-دهد:

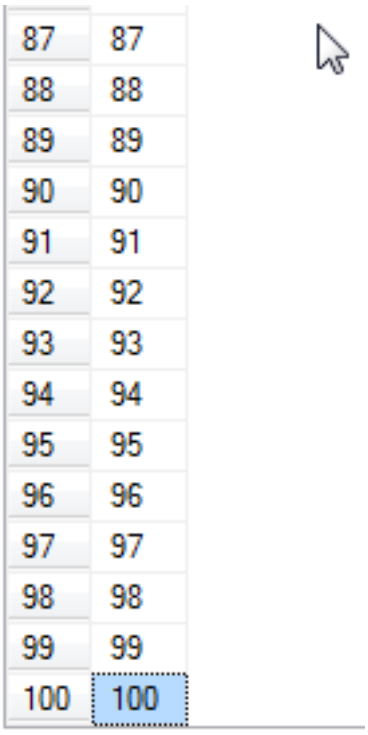
```
WITH digitList
AS
(
   SELECT [digit] from [digits]
)
SELECT
   a.[digit] * 10 + b.[digit] + 1 AS [Digit]
FROM [digitList] AS a CROSSJOIN [digitList] AS b
```

در این کد یک CTE تعریف شده و دو بار مورد استفاده قرار گرفته است. مثلا اگر بخواهید اعداد 1 تا 1000 را تولید کنید می-توانید سه بار از آن استفاده کنید. حاصل این دستور result setی مانند زیر است.



	Digit
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	11
12	12
13	13

نتیجه



87	87
88	88
89	89
90	90
91	91
92	92
93	93
94	94
95	95
96	96
97	97
98	98
99	99
100	100

نتیجه

حتی می-توان از یک CTE در کوئری CTE بعدی مانند کد زیر استفاده کرد.

```
WITH CTE_1 AS
(
   ....
),
CTE_2 AS
(
   SELECT  ... FROM CTE_1 JOIN ...
)
SELECT   *
FROM FOO
LEFTJOIN  CTE_1
LEFTJOIN  CTE_2
```

ایجاد یک CTE بازگشتی[7]

از CTE بازگشتی برای پیمایش جداولی استفاده می-شود که رکوردهای آن دارای رابطه سلسله مراتبی یا درختی است. نمونه این جداول، جدول کارمندان است که مدیر هر کارمند نیز مشخص شده است یا جدولی که ساختار سازمانی را نشان می-دهد یا جدولی که موضوعات درختی را در خود ذخیره کرده است. یکی از مزایای استفاده از CTE بازگشتی، سرعت کار آن در مقایسه با روش-های پردازشی دیگر است.

ساختار کلی یک دستور CTE بازگشتی به صورت زیر است.

```
WITH cteName AS
(
   query1
   UNION ALL
   query2
)
```

در بدنه CTE حداقل دو عضو[8] (کوئری) وجود دارد که بایستی با یکی از عبارت-های زیر به هم متصل شوند.

UNION

UNION ALL

INTERSECT

EXCEPT

query1 شامل دستوری است که اولین سری از رکوردهای result set نهایی را تولید می-کند. اصطلاحا به این کوئری anchor memberمی-گویند.
بعد از دستور query1، حتما بایستی از UNION ALL و امثال آنها استفاده شود.
سپس query2 ذکر می-شود. اصطلاحا به این کوئری recursive member گفته می-شود. این کوئری شامل دستوری است که سطوح بعدی درخت را تولید خواهد کرد. این کوئری دارای شرایط زیر است.
حتما بایستی به CTE که همان cteName است اشاره کرده و در جایی از آن استفاده شده باشد. به عبارت دیگر از رکوردهای موجود در جدول موقت استفاده کند تا بتواند رکوردهای بعدی را تشخیص دهد.
حتما بایستی مطمئن شوید که شرایط کافی برای پایان حلقه پیمایش رکوردها را داشته باشد در غیر این صورت سبب تولید حلقه بی پایان[9] خواهد شد.

بدنه CTE می-تواند حاوی چندین anchor member و چندین recursive member باشد ولی فقط recursive memberها هستند که به CTE اشاره می-کنند.

برای آنکه نکات فوق روشن شود به مثال-های زیر توجه کنید.

فرض کنید جدولی از کارمندان و مدیران آنها داریم که به صورت زیر تعریف و مقداردهی اولیه شده است.

```
IFOBJECT_ID('Employees','U')ISNOTNULL
DROPTABLE dbo.Employees
GO

CREATETABLE dbo.Employees
(
  EmployeeID intNOTNULLPRIMARYKEY,
  FirstName varchar(50)NOTNULL,
  LastName varchar(50)NOTNULL,
  ManagerID intNULL
)

GO


INSERTINTO Employees VALUES (101,'Alireza','Nematollahi',NULL)
INSERTINTO Employees VALUES (102,'Ahmad','Mofarrahzadeh', 101)
INSERTINTO Employees VALUES (103,'Mohammad','BozorgGhommi', 102)
INSERTINTO Employees VALUES (104,'Masoud','Narimani', 103)
INSERTINTO Employees VALUES (105,'Mohsen','Hashemi', 103)
INSERTINTO Employees VALUES (106,'Aref','Partovi', 102)
INSERTINTO Employees VALUES (107,'Hosain','Mahmoudi', 106)
INSERTINTO Employees VALUES (108,'Naser','Pourali', 106)
INSERTINTO Employees VALUES (109,'Reza','Bagheri', 102)
INSERTINTO Employees VALUES (110,'Abbas','Najafian', 102)
```

مثال اول: می-خواهیم فهرست کارمندان را به همراه نام مدیر آنها و شماره سطح درخت نمایش دهیم. کوئری زیر نمونه‌ای از یک کوئری بر اساس CTE بازگشتی می-باشد.

```
WITHcteReports(EmpID, FirstName, LastName, MgrID, EmpLevel)
AS
(
SELECT EmployeeID, FirstName, LastName, ManagerID, 1
FROM Employees
WHERE ManagerID ISNULL
UNIONALL
SELECT e.EmployeeID, e.FirstName, e.LastName, e.ManagerID,r.EmpLevel + 1
FROM Employees e INNERJOINcteReports r
ON e.ManagerID = r.EmpID
)
SELECT
FirstName +' '+ LastName AS FullName,
EmpLevel,
(SELECT FirstName +' '+ LastName FROM Employees
WHERE EmployeeID = cteReports.MgrID)AS Manager
FROMcteReports
ORDERBY EmpLevel, MgrID
```

کوئری اول در بدنه CTE رکورد مدیری را می-دهد که ریشه درخت بوده و بالاسری ندارد و شماره سطح این رکورد را 1 در نظر می-گیرد.
کوئری دوم در بدنه CTE از یک JOIN بین Employees و cteReports استفاده کرده و کارمندان زیر دست هر کارمند قبلی (فرزندان) را بدست آورده و مقدار شماره سطح آنرا به صورت Level+1 تنظیم می-کند.
در نهایت با استفاده از CTE و یک subquery جهت بدست آوردن نام مدیر هر کارمند، نتیجه نهایی تولید می-شود.

مثال دوم: می-خواهیم شناسه یک کارمند را بدهیم و نام او و نام مدیران وی را به عنوان جواب در خروجی بگیریم.

```
WITHcteReports(EmpID, FirstName, LastName, MgrID, EmpLevel)
AS
(
SELECT EmployeeID, FirstName, LastName, ManagerID, 1
FROM Employees
WHERE EmployeeID = 110
UNIONALL
SELECTe.EmployeeID, e.FirstName, e.LastName, e.ManagerID,r.EmpLevel + 1
FROM Employees e INNERJOINcteReports r
ON e.EmployeeID = r.MgrID
```

```
)
SELECT
FirstName +' '+ LastName AS FullName,
EmpLevel
FROMcteReports
ORDERBY EmpLevel
```

اگر دقت کنید اولین تفاوت در خط اول مشاهده می-شود. در اینجا مشخص می-کند که اولین سری از رکوردها چگونه انتخاب شود. مثلا کارمندی را می-خواهیم که شناسه آن 110 باشد.
دومین تفاوت اصلی این کوئری با مثال قبلی، در قسمت دوم دیده می-شود. شما می‌خواهید مدیر (پدر) کارمندی که در آخرین پردازش در جدول موقت قرار گرفته است را استخراج کنید.

[1] a temporary named result set

[2] recursive

[3] nonrecursive

[4] Scope

[5]مثلا محدوده کدهای یک روال یا یک تابع

[6] nonrecursive

[7] recursive

[8] member

[9] Infinite loop

نظرات خوانندگان

نویسنده: همراز
تاریخ: ۱۳۹۳/۰۱/۲۶ ۱۱:۷

ضمن تشکر از پست مفید جناب عمران یکی از استفاده‌های CTE افزودن شماره ردیف به ساختار خروجی و محدود کردن نتیجه باتوجه به شماره ردیف است.
مثلاً ردیف 20 تا 30 ... که البته با پارامتر پاس می‌شوند.

```
WITH RCTE AS
(
SELECT TOP (100)
ROW_NUMBER() OVER (ORDER BY Invoice.InsertDate ASC) AS RowNumber,
Invoice.ID,
Invoice.PreInvoiceNo,
Invoice.InvoiceNo,
Invoice.IssueDate,
Invoice.CustomerID,  ...
FROM
Invoice
WHERE
Invoice.HistorySequence = 1
)
SELECT DISTINCT
RCTE.ID,
RCTE.PreInvoiceNo,
RCTE.InvoiceNo,
(dbo.fnc_Calendar_Gregorian_to_Persian(RCTE.IssueDate) + 'T' + CONVERT(CHAR(8), RCTE.IssueDate, 14)) AS
IssueDate,
RCTE.CustomerID,
Customer.NameEn AS CustomerNameEn,
Customer.NameFa AS CustomerNameFa,
FROM
RCTE
INNER JOIN Customer ON RCTE.CustomerID = Customer.ID
WHERE
RowNumber BETWEEN @StartFrom AND (@RowsCount + @StartFrom - 1)
```