**بسمه تعالی**



برنامه نویسی بانک اطلاعاتی SQL Server

**نویسنده : محمد حسین فخرآوری**

**ایجاد یک CTE غیر بازگشتی**

|  |
| --- |
| WITH temp  AS  (  SELECT \* FROM Security.Users  )  SELECT \* FROM temp  ORDER BY temp.Id DESC |
| WITH temp  AS  (  SELECT \* FROM Security.Users  )  SELECT \* FROM temp x  JOIN CRM.Contactus c ON c.IdUsers = x.Id  ORDER BY x.Id DESC |

**استفاده از دو CTE را به صورت همزمان**

|  |
| --- |
| WITH Usr  AS  (  SELECT \* FROM Security.Users  ),Ulog  AS  (  SELECT \* FROM Security.UserLogin  )  SELECT \* FROM Ulog lg JOIN Usr u ON lg.IdUser=u.Id |

**ایجاد یک CTE بازگشتی**

یک کوئری بازگشتی دارای دو بخش هست به نام‌های Anchor و recursive در بخش دوم کوئری باز خودش را فراخوانی می‌کند تا به داده هایی که در مرحله قبل تولید شده اند دسترسی پیدا کند در اولین فراخوانی توسط عضو recursive، داده‌های تولید شده در قسمت Anchor قابل دسترسی هستند. در قسمت دوم، کوئری آنقدر خود را فراخوانی می‌کند تا دیگر سطری از مرحله قبل وجود نداشته باشد که به آن مراجعه کند.

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

|  |
| --- |
| CREATE TABLE dbo.Employees  (  EmployeeID INT NOT NULL PRIMARY KEY,  FirstName varchar(50)NOT NULL,  LastName varchar(50)NOT NULL,  ManagerID INT NULL  )  INSERT INTO Employees VALUES (101,'Alireza','Nematollahi',NULL)  INSERT INTO Employees VALUES (102,'Ahmad','Mofarrahzadeh', 101)  INSERT INTO Employees VALUES (103,'Mohammad','BozorgGhommi', 102)  INSERT INTO Employees VALUES (104,'Masoud','Narimani', 103)  INSERT INTO Employees VALUES (105,'Mohsen','Hashemi', 103)  INSERT INTO Employees VALUES (106,'Aref','Partovi', 102)  INSERT INTO Employees VALUES (107,'Hosain','Mahmoudi', 106)  INSERT INTO Employees VALUES (108,'Naser','Pourali', 106)  INSERT INTO Employees VALUES (109,'Reza','Bagheri', 102)  INSERT INTO Employees VALUES (110,'Abbas','Najafian', 102)  SELECT \* FROM dbo.Employees |

**مثال اول:** می­خواهیم فهرست کارمندان را به همراه نام مدیر آنها و شماره سطح درخت نمایش دهیم. کوئری زیر نمونه‌ای از یک کوئری بر اساس CTE بازگشتی می­باشد.

|  |
| --- |
| WITH cteReports(EmpID, FirstName, LastName, MgrID, EmpLevel)  AS  (  SELECT EmployeeID, FirstName, LastName, ManagerID, 1  FROM Employees  WHERE ManagerID IS NULL  UNION ALL  SELECT e.EmployeeID, e.FirstName, e.LastName, e.ManagerID,r.EmpLevel + 1  FROM Employees e JOIN cteReports r ON e.ManagerID = r.EmpID  )  SELECT  FirstName +' '+ LastName AS FullName,EmpLevel,  (SELECT FirstName +' '+ LastName FROM Employees WHERE EmployeeID = cteReports.MgrID)AS Manager  FROM cteReports  ORDER BY EmpLevel, MgrID |

کوئری اول در بدنه CTE رکورد مدیری را می­دهد که ریشه درخت بوده و بالاسری ندارد و شماره سطح این رکورد را 1 در نظر می­گیرد.

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 می­کند.

در نهایت با استفاده از CTE و یک subquery جهت بدست آوردن نام مدیر هر کارمند، نتیجه نهایی تولید می­شود.

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

|  |
| --- |
| WITH cteReports(EmpID, FirstName, LastName, MgrID, EmpLevel)  AS  (  SELECT EmployeeID, FirstName, LastName, ManagerID, 1  FROM Employees  WHERE EmployeeID = 110  UNION ALL  SELECT e.EmployeeID, e.FirstName, e.LastName, e.ManagerID,r.EmpLevel + 1  FROM Employees e JOIN cteReports r  ON e.EmployeeID = r.MgrID  )  SELECT  FirstName +' '+ LastName AS FullName,  EmpLevel  FROM cteReports  ORDER BY EmpLevel |

اگر دقت کنید اولین تفاوت در خط اول مشاهده می­شود. در اینجا مشخص می­کند که اولین سری از رکوردها چگونه انتخاب شود. مثلا کارمندی را می­خواهیم که شناسه آن 110 باشد.

دومین تفاوت اصلی این کوئری با مثال قبلی، در قسمت دوم دیده می­شود. شما می‌خواهید مدیر (پدر) کارمندی که در آخرین پردازش در جدول موقت قرار گرفته است را استخراج کنید.

**مثال**

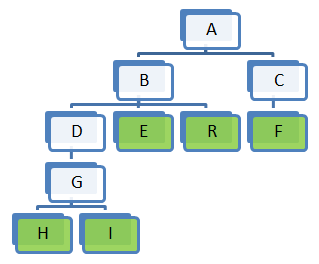
توضیح تکنیک:  
در گام اول اندیس شروع و پایان کلمه اول را بدست می‌آوریم.   
سپس در گام بعدی از اندیس پایان کلمه قبلی به عنوان اندیس شروع کلمه جدید استفاده می‌کنیم.   
و اندیس پایان کلمه توسط تابع charindex بدست می‌آید.  
کوئری تا زمانی ادامه پیدا میکند که کلمه برای تجزیه کردن در رشته باقی مانده باشد. فقط فراموش نکنید که حتما باید آخر عبارت یک کارکتر space داشته باشید.

**بدست آوردن برگهای**

|  |
| --- |
| DECLARE @a NVARCHAR(350)=N'محمدحسین فخرآوری متولد 1369/01/07 از بوشهر'+SPACE(1);  WITH MyWords(ranking, word, string) AS  (  SELECT 1,CAST(SUBSTRING(@a, 1, CHARINDEX(' ', @a) - 1) AS NVARCHAR(200)),  STUFF(@a, 1, CHARINDEX(' ', @a), '')  UNION ALL  SELECT ranking + 1,CAST(SUBSTRING(string, 1, CHARINDEX(' ', string) - 1) AS NVARCHAR(200)),  STUFF(string, 1, CHARINDEX(' ', string), '')  FROM MyWords WHERE CHARINDEX(' ', string) > 0  )  SELECT ranking, word FROM MyWords; |
| DECLARE @S NVARCHAR(350) =N'محمدحسین فخرآوری متولد 1369/01/07 از بوشهر'+SPACE(1);  WITH CTE AS  (  SELECT 1 rnk, 1 start, CHARINDEX(' ', @S)-1 ed  UNION ALL  SELECT rnk+1, ed+2, CHARINDEX(' ', @S, ed+2)-1 FROM CTE WHERE CHARINDEX(' ', @S, ed+2)>0  )  SELECT rnk, SUBSTRING(@S, start, ed-start+1) AS word FROM CTE; |

**یک درخت توسط Recursive CTE**

به  ساختار سلسله مراتبی زیر توجه بفرمایید:

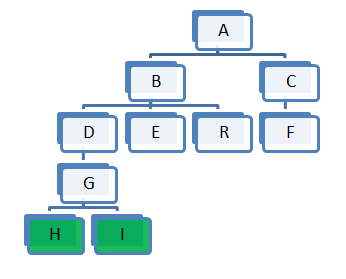
[](https://www.dotnettips.info/file/image?name=pic1-7bf2951e0e4f46acb28d82ff94308136.png)

گره هایی که با رنگ سبز علامت گذاری شده اند را گره‌های برگ می‌نامیم چون که آن گره‌ها بدون زیر شاخه هستند.  
فرض کنید از ما خواسته شده است با داشتن گره A تمام برگهای این شاخه را بدست بیاوریم.  
دو مرحله را باید طی کنیم ابتدا تمام گره هایی که زیر شاخه گره A هستند را باید بدست آورد سپس توسط یک گزاره گره‌های برگ را فیلتر کنیم.

در واقع گره هایی برگ هستند که پدر هیچ گره‌ی دیگری نباشند.

|  |
| --- |
| declare @t table  (id char(1) primary key,  parent char(1));  insert @t values  ('A',null), --Level 1  ('B', 'A'), ('C', 'A'), --Level 2  ('D', 'B'), ('E', 'B'),('R','B'), ('F', 'C'), --Level 3  ('G', 'D'), --Level 4  ('H', 'G'), ('I', 'G'); --Level 5 |
| ;WITH cte AS  (  SELECT id, rnk=0 FROM @t WHERE parent='A'  UNION ALL  SELECT t.id, rnk+1 FROM cte JOIN @t t ON cte.id=t.parent  )  SELECT \* FROM cte WHERE NOT EXISTS (SELECT \* FROM @t WHERE parent=cte.id); |

و حالا به درخت زیر توجه بفرمایید:

[](https://www.dotnettips.info/file/image?name=pic2-d940fcf020de44d482e2b1ae1b01700f.png)

هدف پیدا کردن برگ هایی از شاخه مورد نظر است که در پایین‌ترین سطح قرار گرفته باشند. برای این منظور از همان query بازگشتی استفاده کرده و با کمک تابع dense\_ranke گره‌های مورد نظر را بدست میاوریم.

|  |
| --- |
| ;WITH cte AS  (SELECT id, rnk=0 FROM @t WHERE parent='A'  UNION ALL  SELECT t.id, rnk+1 FROM cte JOIN @t t ON cte.id=t.parent  )  SELECT \*  FROM(SELECT \*, DENSE\_RANK() OVER (ORDER BY rnk DESC) rk FROM cte)t  WHERE rk=1; |