عنوان: **تجزیه یک رشته به کلمات تشکیل دهنده آن توسط Recursive CTE**
نویسنده: محمد سلیم ابادی
تاریخ: ۱۳۹۱/۱۰/۲۹ ۲۱:۴۰
آدرس: www.dotnettips.info
برچسب‌ها: T-SQL, recursive cte, splitting, sql server 2005

برای پردازش یک عبارت در بسیاری از موارد نیاز هست که عبارت به کلمات تشکیل دهنده اش تجزیه شود. روش‌های متنوعی برای انجام این عمل وجود دارد که یکی از شناخته شده‌ترین آنها استفاده از جدول اعداد می‌باشد (البته از بین روش‌های مجموعه گرا/set -based).
روشهایی که قرار هست در ادامه توضیح داده شوند بر اساس کوئری بازگشتی می‌باشند. الگوریتم‌های متنوعی بر اساس recursive CTE برای حل این مساله خلق شده اند. که من تنها به دو روش آن اکتفا می‌کنم.

Recursive CTE در نسخه‌ی 2005 به SQL Server اضافه شده است. توسط این تکنیک مسائل پیچیده و گوناگونی را میتوان بسادگی حل نمود. مخصوصا مسائلی که ماهیت بازگشتی دارند مثل پیمایش یک درخت یا پیمایش یک گراف وزن دار.

روش اول:

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.

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.

```
DECLARE @S VARCHAR(50)='I am a student I go to school ';
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

/* Result
rnk         word
----------- -------
1           I
2           am
3           a
4           student
5           I
6           go
7           to
8           school
*/
```

روش دوم:

در این روش در همان CTE عبارت تجزیه می‌شود و عمل تفکیک به مرحله بعدی واگذار نمی‌شود،

در گام اول، اولین کلمه انتخاب می‌شود. و سپس آن کلمه از رشته حذف می‌شود. با این روش همیشه اندیس شروع کلمه برابر با 1 خواهد بود و اندیس پایان کلمه توسط تابع charindex بدست خواهد آمد.

در گام بعدی اولین کلمه موجود در رشته ای که قبلا اولین کلمه از آن جدا شده است بدست می‌آید و باز مثل قبلی کلمه انتخاب شده از رشته جدا شده و رشته برش یافته به مرحله بعد منتقل می‌شود.

در این روش مثل روش قبلی آخر عبارتی که قرار هست تجزیه شود باید یک کارکتر خالی وجود داشته باشد.

```
DECLARE @a VARCHAR(50)='I am a student I go to school ';

WITH MyWords(ranking, word, string) AS(

    SELECT 1,
           CAST(SUBSTRING(@a, 1, CHARINDEX(' ', @a) - 1) AS VARCHAR(25)),
           STUFF(@a, 1, CHARINDEX(' ', @a), '')

    UNION ALL

    SELECT ranking + 1,
           CAST(SUBSTRING(string, 1, CHARINDEX(' ', string) - 1) AS VARCHAR(25)),
           STUFF(string, 1, CHARINDEX(' ', string), '')
      FROM MyWords
     WHERE CHARINDEX(' ', string) > 0
)
SELECT ranking, word FROM MyWords;
```

و خروجی:

```
ranking     word
----------- -------------------------
1           I
2           am
3           a
4           student
5           I
6           go
7           to
8           school
```

نظرات خوانندگان

نویسنده: محسن
تاریخ: ۱۳۹۱/۱۰/۲۹ ۲۱:۴۴

از مقاله شما دوست عزیز کمال تشکر را دارم.

نویسنده: محمد سلم ابادی
تاریخ: ۱۳۹۱/۱۰/۲۹ ۲۲:۲۳

ممنونم دوست گرامی

عنوان: **حذف نمودن کاراکتر های ناخواسته توسط Recursive CTE قسمت اول**
نویسنده: محمد سلیم ابادی
تاریخ: ۱۳۹۱/۱۱/۰۱ ۱۵:۳۵
آدرس: www.dotnettips.info
برچسب‌ها: SQL Server, T-SQL, recursive cte, replace

شاید برایتان تا حالا پیش آمده باشد که بخواهید یکسری کاراکترهای ناخواسته و اضافه را از یک رشته حذف کنید. بطور مثال تمام کاراکتر هایی غیر عددی را باید از یک رشته حذف نمود تا آن رشته قابلیت تبدیل به نوع integer را بدست بیاورد.

اگر تعداد کاراکترهای ناخواسته محدود و مشخص هستند می‌توانید با دستور REPLACE آنها را حذف کنید، مثلا میخواهیم هر سه کاراکتر ~!@ از رشته حذف شوند:

```
DECLARE @s VARCHAR(50) = '~~~~~~!@@@@@@ salam';
SET @s = REPLACE(REPLACE(REPLACE(@s, '~', ''), '!', ''), '@', '');
SELECT @s AS new_string
```

ولی هنگامی که کاراکترها نامحدود بوده امکان نوشتن تابع REPLACE به کرات بی معنا است در این حالت باید دنبال روشی پویا و تعمیم پذیر بود.
با جستجویی که در اینترنت انجام دادم متوجه شدم تکنیک WHILE یا همون loop یکی از روش‌های رایج برای انجام اینکار هست، که احتمالا به دلیل سهولت در بکارگیری و سادگی آن بوده که عمومیت پیدا کرده است.
مستقل از این صحبتها هدف معرفی یک روش مجموعه گرا (set-based) برای این مساله می‌باشد.

حذف کاراکترها ناخواسته با تکنیک Recursive CTE
راه حل بر اساس جدول زیر است:

```
CREATE TABLE test_string
(id integer not null primary key,
 string_value varchar(500) not null);

INSERT INTO test_string
VALUES (1, '@@@@ #### salam 12345'),
       (2, 'good $$$$$ &&&&&& bye 00000');
```

حالا فرض کنید می‌خواهیم هر کاراکتری غیر از حروف الفبای انگلیسی و فاصله(space) از رشته حذف شود. پس دو داده فوق به صورت salam و good bye در انتها در خواهند آمد.
برای حذف کاراکترهای ناخواسته فوق query زیر را اجرا کنید.

```
WITH CTE (ID, MyString, Ix) AS
(
    SELECT id,
           string_value,
           PATINDEX('%[^a-z ]%', string_value)
    FROM   test_string

    UNION ALL

    SELECT id,
           CAST(REPLACE(MyString, SUBSTRING(MyString,Ix , 1), '') AS VARCHAR(500)),
           PATINDEX('%[^a-z ]%', REPLACE(MyString, SUBSTRING(MyString,Ix , 1), ''))
    FROM   CTE
    WHERE  Ix > 0
)
SELECT *
FROM   cte
--WHERE  Ix = 0;
ORDER BY id, CASE WHEN Ix = 0 THEN 1 ELSE 0 END, Ix;
```

توضیح query:

در قسمت anchor اندیس اولین کاراکتر ناخواسته (خارج از رنج حروف الفبا و فاصله) بدست می‌آید. سپس در قسمت recursive هر کاراکتری که برابر باشد با کاراکتر ناخواسته ای که در مرحله قبل بدست آمده از رشته حذف می‌شود این عملیات توسط تابع replace صورت می‌گیرد و اندیس کاراکتر ناخواسته بعدی بعد از حذف کاراکتر ناخواسته قبلی بدست می‌آید که به مرحله بعد منتقل می‌شود. این مراحل تا آنجایی پیش می‌رود که دیگر کاراکتر ناخواسته ای در رشته وجود نداشته باشد.

به جدول زیر توجه بفرمایید (خروجی query فوق)

	ID	MyString	lx
1	1	@@@@ #### salam 12345	1
2	1	#### salam 12345	2
3	1	salam 12345	9
4	1	salam 2345	9
5	1	salam 345	9
6	1	salam 45	9
7	1	salam 5	9
8	1	salam	0
9	2	good $$$$$ &&&&&& bye 00000	6
10	2	good &&&&&& bye 00000	7
11	2	good bye 00000	12
12	2	good bye	0

نتیجه مطلوب ما آن دو سطری است که در کادر بنفش هستند. که اگر به ستون Ix اشان توجه کنید مقدارش برابر با 0 است.

لطفا به سطر اول جدول توجه بفرمایید مشاهده می‌شود که هر 4 کاراکتر @ یکباره از رشته حذف شدند که بدلیل استفاده از تابع REPLACE میباشد.

نظرات خوانندگان

نویسنده: سید حمزه
تاریخ: ۱۳۹۲/۰۶/۲۷ ۱۶:۵۳

سلام
خیلی جامع بود
فقط من متوجه نشدم دقیقا کجای کوئریم باید اینو بنویسم.
و همینطور میخواستم اگر بشه یک فانکشن بسازم و از اون هم توی select استفاده کنم.
ممنون

عنوان: **حذف کاراکتر های ناخواسته توسط Recursive CTE قسمت دوم**
نویسنده: محمد سلیم ابادی
تاریخ: ۱۳۹۱/۱۱/۰۸ ۸:۵۵
آدرس: www.dotnettips.info
برچسب‌ها: SQL Server, recursive cte

مقدمه

در قسمت پیشین نشان داده شد که چگونه کاراکترهای خارج از رنج حروف الفبای انگلیسی از عبارات موجود در یک جدول حذف شدند.
اکنون شرایط کمی تغییر کرده است کاراکترهای ناخواسته در قالب یک مجموعه (جدول) به ما ارائه داده می‌شوند. ما بایستی تمام کاراکترهای داده شده را از عبارات (موجود در جدول) در صورت تطابق حذف کنیم.

جدول کاراکترهای ناخواسته Unwanted و جدول داده‌ها Data نامگذاری شده اند.

```
CREATE TABLE Data
(
id INTEGER NOT NULL PRIMARY KEY IDENTITY,
data VARCHAR(50) NOT NULL
);

INSERT INTO Data
VALUES ('~!hasan @#$%^&*(reza)[ali]^^^^^^^^'),
('(Ja[][][]va~!@#$*d-mohammad)'),
('Mohammad'), ('Maryam');

CREATE TABLE Unwanted
(
id INT NOT NULL PRIMARY KEY IDENTITY,
chars CHAR(1) NOT NULL UNIQUE
);

INSERT Unwanted
VALUES ('~'),('!'),('@'),('#'),('$'),('%'),
('['),(']'),('^'),('&'),('*');
```

الگوریتم

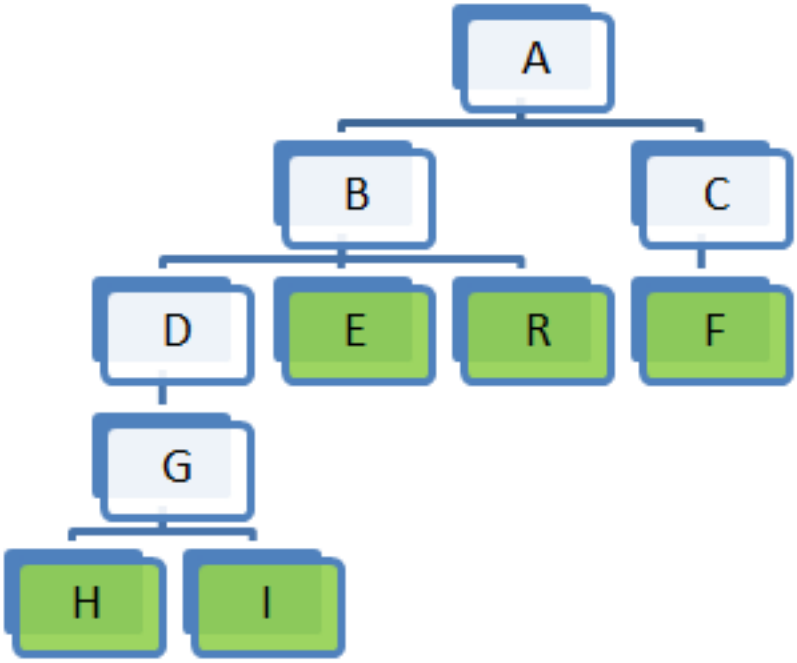
قبل از هر چیزی فرض میگیریم که داده‌ها ستون Id جدول Unwanted کاملا متوالی نیستند و gap بین مقادیر id وجود دارد. پس برای داشتن مقادیر متوالی از تابع row_number استفاده شده است.
بعد از آن باید عباراتی که حداقل با یک کاراکتر ناخواسته تطابق پیدا میکنند انتخاب شوند. و عملیات پاکسازی روی این عبارات به تعداد کاراکترهای موجود در جدول unwanted انجام می‌شود. یعنی در مرحله اول شمارنده برابر است با تعداد کاراکترهای ناخواسته سپس در هر فراخوانی یک مقدار از این شمارنده کم شده تا اینکه به 0 برسد در اینجا کار به اتمام می‌رسد.
بعد از پایان یافتن فراخوانی‌های query بازگشتی باید سطرهایی که شمارنده آنها برابر با 0 است انتخاب شده و عباراتی که در مرحله اول به دلیل عدم تطابق انتخاب نشدن بایستی به نتیجه اضافه شوند. این کار را توسط Union all انجام شده است.

```
WITH pre AS
(
SELECT ROW_NUMBER() OVER(ORDER BY id) AS id, chars
  FROM Unwanted
),
 cte AS
(
SELECT  data.id,
        nbr,
        CAST(Data AS VARCHAR(50)) AS Data
  FROM Data
       CROSS JOIN (SELECT COUNT(*) AS nbr FROM Unwanted)t
 WHERE EXISTS
       (SELECT *
          FROM Unwanted
         WHERE Data LIKE '%#' + chars + '%' ESCAPE '#')

 UNION ALL

 SELECT C.id,
        nbr - 1,
        CAST(REPLACE(C.data, U.chars, '') AS VARCHAR(50))
   FROM cte AS C
```

```
        INNER JOIN pre U
        ON C.nbr = U.id
  WHERE nbr > 0
)
SELECT data
  FROM cte
 WHERE nbr = 0

 UNION ALL

SELECT data
  FROM Data
 WHERE NOT EXISTS
       (SELECT id
          FROM cte
         WHERE cte.id = data.id);
```

عنوان: **بدست آوردن برگهای یک درخت توسط Recursive CTE**
نویسنده: محمد سلیم ابادی
تاریخ: ۱۳۹۱/۱۱/۱۶ ۱۷:۳۵
آدرس: www.dotnettips.info
برچسب‌ها: SQL Server, recursive cte, query, not exists, dense_rank

امروز در یک تالار سوالی مطرح شد با این عنوان "چگونه می‌توانم گره‌های برگ یک شاخه را بدست بیاورم". خب راه حلی که فورا به ذهنم رسید استفاده از یک query بازگشتی (recursive) بود.
به ساختار سلسله مراتبی زیر توجه بفرمایید:



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.

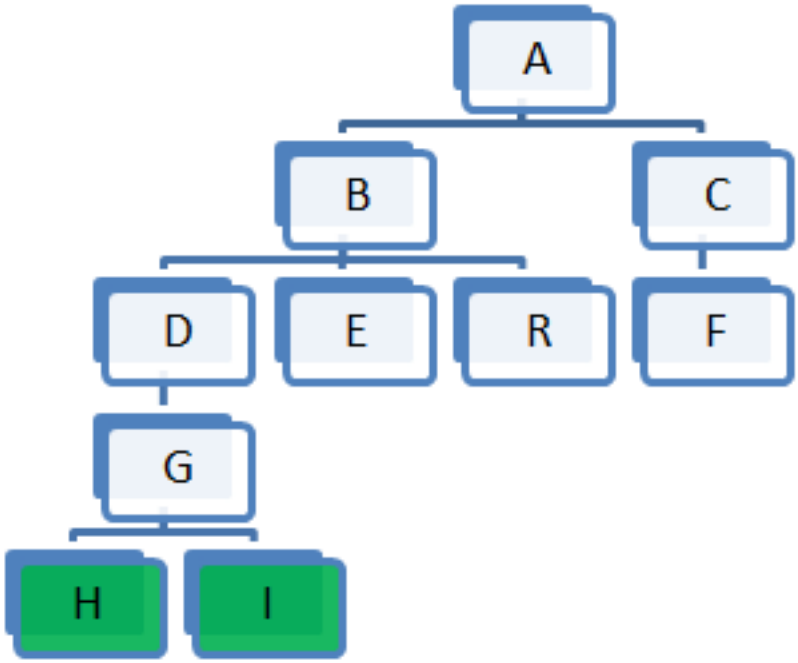
در واقع گره هایی برگ هستند که پدر هیچ گره‌ی دیگری نباشند.

```
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:



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
```

نظرات خوانندگان

نویسنده: بهمن خلفی
تاریخ: ۱۳۹۱/۱۱/۱۷ ۱۴:۴۱

ممنون از مطلب مفیدتون
اما یک سوال : چگونه میتوان به نتیجه زیر دست پیدا کرد

```
a , ab , ac , abd , abe , abr , abdg , abdgh , ....
```

نویسنده: محمد سلم ابادی
تاریخ: ۱۳۹۱/۱۱/۱۷ ۱۵:۳۰

من دقیقا متوجه نشدم نتیجه مورد نظر شما چیست.
آیا نتیجه مورد نظر شما به صورت الحاق یافته (concatenated) هست یا نه؟
در هر صورت باید یکی از دو query زیر نتیجه مورد نظر شما را تولید کند.

```
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,
       concats = cast(id as varchar(10))
from @t
where parent is null

union all

select t.id, rnk+1,
       cast(cte.concats + t.id as varchar(10))
from cte join @t t
on cte.id = t.parent
)
select * from cte
/*
id   rnk         concats
---- ----------- ----------
A    0           A
B    1           AB
C    1           AC
F    2           ACF
D    2           ABD
E    2           ABE
R    2           ABR
G    3           ABDG
H    4           ABDGH
I    4           ABDGI
*/
;with cte as
(
select id, rnk=0,
       concats = cast(id as varchar(10))
from @t
where parent is null

union all

select t.id, rnk+1,
       cast(cte.concats + t.id as varchar(10))
from cte join @t t
on cte.id = t.parent
)
```

```
select stuff(d.list,1,1,'') as concats
from (select ','+concats
      from cte
  for xml path(''))d(list)
/*
concats
---------------------------------------
A,AB,AC,ACF,ABD,ABE,ABR,ABDG,ABDGH,ABDGI
*/
```

موفق باشید