CÁC HÀM RANKING MỚI TRONG SQL SERVER 2005

Cùng v

Cùng với SQL Server 2005, Microsoft đã giới thiệu một số tính năng mới và những tính năng này sẽ giúp cho chuyên viên về DBA hay SQL Server dễ dàng hơn trong việc viết mã và duy trì cơ sở dữ liệu SQL Server. Bài này sẽ thảo luận về các hàm ranking mới được cung cấp trong SQL Server 2005. Các tính năng mới đó sẽ giúp bạn dễ dàng viết mã T-SQL để kết hợp xếp loại được tập hợp kết quả của bạn. Bài sẽ hướng dẫn từng phần trong các hàm ranking mới và cung cấp một số ví dụ nhằm minh họa hoạt động của hàm.

Các hàm Ranking là gì?

Các hàm Ranking cho phép bạn có thể đánh số liên tục (xếp loại) cho các tập hợp kết quả. Các hàm này có thể được sử dụng để cung cấp số thứ tự trong hệ thống đánh số tuần tự khác nhau. Có thể hiểu đơn giản như sau: bạn có từng con số nằm trên từng dòng liên tục, tại dòng thứ nhất xếp loại số 1, dòng thứ 2 xếp loại số là 2… Bạn có thể sử dụng hàm ranking theo các nhóm số tuần tự, mỗi một nhóm sẽ được đánh số theo lược đồ 1,2,3 và nhóm tiếp theo lại bắt đầu bằng 1,2,3…

Dữ liệu chạy thử cho các ví dụ

Để có một vài ví dụ cho từng hàm ranking, tôi cần thiết lập một số dữ liệu chạy thử. Trong dữ liệu chạy thử, tôi sử dụng một bảng “Person” khá đơn giản. Bảng sẽ bao gồm 3 cột “FirstName”, “Age” và “Gender”. Đoạn mã dưới nhằm tạo ra và ghi lại dữ liệu chạy thử vào file.

```
SET NOCOUNT ON
CREATE TABLE Person(
FirstName VARCHAR(10),
Age INT,
Gender CHAR(1))
INSERT INTO Person VALUES ('Ted',23,'M')
INSERT INTO Person VALUES ('John',40,'M')
INSERT INTO Person VALUES ('George',6,'M')
INSERT INTO Person VALUES ('Mary',11,'F')
INSERT INTO Person VALUES ('Sam',17,'M')
INSERT INTO Person VALUES ('Doris',6,'F')
INSERT INTO Person VALUES ('Frank',38,'M')
```

```
INSERT INTO Person VALUES ('Larry',5,'M')
INSERT INTO Person VALUES ('Sue',29,'F')
INSERT INTO Person VALUES ('Sherry',11,'F')
INSERT INTO Person VALUES ('Marty',23,'F')
```

Hàm ROW_NUMBER

Hàm đầu tiên tôi muốn nói tới là ROW_NUMBER. Hàm này trả lại một dãy số tuần tự bắt đầu từ 1 cho mỗi dòng hay nhóm trong tập hợp kết quả. Hàm ROW_NUMBER sẽ có cú pháp sau:

```
ROW_NUMBER ( ) OVER ( [  ] )
```

Trong đó:

là cột hay tập hợp các cột được sử dụng để quyết định việc gộp nhóm cho hàm ROW_NUMBER áp dụng cho việc đánh số tuần tự.

là một cột hay tập hợp các cột được sử dụng để sắp xếp tập hợp kết quả trong nhóm (partition)

Để hiểu thêm về cách sử dụng hàm ROW_NUMBER, ví dụ dưới sẽ đánh số liên tục cho tất cả các dòng trong bảng Person và sắp xếp chúng theo trường Age

```
SELECT ROW_NUMBER() OVER (ORDER BY Age) AS [Row Number by Age],
      FirstName,
      Age
 FROM Person
```

Và đây là tập hợp kết quả mã T-SQL trên:

```
Row Number by Age    FirstName   Age
-------------------- ---------- ------
1                    Larry       5
2                    Doris       6
3                    George      6
4                    Mary        11
5                    Sherry      11
6                    Sam         17
7                    Ted         23
8                    Marty       23
9                    Sue         29
10                   Frank       38
11                   John        40
```

Bạn có thể thấy tôi đã đánh số liên tục cho toàn bộ các dòng trong bảng Person bắt đầu từ số 1, và

tập hợp kết quả được sắp xếp theo cột Age. Sự sắp xếp này được hoàn thiện là do tiêu chuẩn "ORDER BY Age" trong mệnh đề ORDER BY của hàm ROW_NUMBER.

Giả sử bạn không muốn tập hợp kết quả của bạn được sắp xếp mà muốn đưa bảng trở lại sắp xếp theo số bản ghi của từng dòng. Hàm ROW_NUMBER lại luôn yêu cầu phải có mệnh đề ORDER BY, vậy bạn cần phải đưa một giá trị nào đó vào trong mệnh đề này. Trong hàm truy vấn bên dưới tôi đã chỉ định "SELECT 1" vào trong mệnh đề ORDER BY, điều này sẽ chỉ trả lại kết quả là bảng như đã lưu trữ ban đầu và tất nhiên cách đánh số tuần tự vẫn bắt đầu từ 1:

```
SELECT ROW_NUMBER() OVER (ORDER BY (SELECT 1)) AS [Row Number by Record Set],
      FirstName,
      Age
  FROM Person
```

Đây là tập hợp kết quả khi chạy hàm truy vấn trên:

```
Row Number by Record FirstName   Age
-------------------- ---------- ------
1                    Ted        23
2                    John       40
3                    George     6
4                    Mary       11
5                    Sam        17
6                    Doris      6
7                    Frank      38
8                    Larry      5
9                    Sue        29
10                   Sherry     11
11                   Marty      23
```

Hàm ROW_NUMBER không chỉ cho phép bạn sắp xếp toàn bộ tập hợp dòng mà còn có thể sử dụng mệnh đề PARTITION để lọc ra nhóm dòng cần đánh số. Các dòng sẽ được đánh số tuần tự trong từng giá trị PARTITION độc nhất. Các dãy số được đánh sẽ luôn bắt đầu từ 1 cho từng giá trị PARTITION mới trong tập hợp bản ghi của bạn. Hãy xem hàm truy vấn dưới đây

```
SELECT ROW_NUMBER() OVER (PARTITION BY Gender ORDER BY Age) AS [Partition by Gender],
      FirstName,
      Age,
      Gender
  FROM Person
```

Khi chạy truy vấn trên, tập hợp kết quả sẽ ra như sau:

```
Partition by Gender  FirstName  Age        Gender
-------------------- ---------- ---------- ------
1                    Doris      6          F
2                    Mary       11         F
3                    Sherry     11         F
4                    Sue        29         F
1                    Larry      5          M
2                    George     6          M
3                    Sam        17         M
4                    Ted        23         M
5                    Marty      23         M
6                    Frank      38         M
7                    John       40         M
```

Trong ví dụ này tôi đã phân vùng bởi Gender và sắp xếp theo Age. Thực hành theo ví dụ này sẽ cho phép tôi đánh số tuần tự các bản ghi là Female trong bảng Person theo độ tuổi, và sau đó việc đánh số sẽ bắt đầu lại với nhóm là Male.

Hàm RANK

Đôi khi bạn muốn một dòng có cùng sắp xếp giá trị cột như các dòng khác có cùng một xếp loại. Nếu thế thì hàm RANK () có thể giúp bạn. Hàm RANK có cú pháp như sau:

RANK ( ) OVER ([])

Trong đó:

là một cột hay tập hợp các cột được sử dụng để quyết đinh việc đánh số liên tục trong hàm RANK

là một cột hay tập hợp các cột được sử dụng để sắp xếp tập hợp kết quả trong nhóm (partition)

Hàm RANK sẽ đánh số liên tục một tập hợp bản ghi nhưng khi có 2 dòng có cùng giá trị sắp xếp thì hàm sẽ đánh giá là cùng bậc giá trị. Giá trị xếp loại vẫn sẽ tăng kể cả khi có 2 dòng cùng giá trị, vì vậy khi đánh giá một giá trị sắp xếp tiếp theo thì số thứ tự vẫn tiếp tục được đánh nhưng sẽ tăng thêm 1 giá trị vào các dòng tiếp theo trong tập hợp.

Đây là ví dụ của hàm rank trong tập hợp bản ghi sắp xếp theo Age:

```
SELECT RANK() OVER (ORDER BY Age) AS [Rank by Age],
      FirstName,
      Age
  FROM Person
```

Và kết quả trả về:

Rank by Age	FirstName	Age
1	Larry	5
2	Doris	6
2	George	6
4	Mary	11
4	Sherry	11
6	Sam	17
7	Ted	23
7	Marty	23
9	Sue	29
10	Frank	38
11	John	40

Như bạn thấy, với các dòng trùng giá trị Age thì ở phần Rank by Age cũng có cùng giá trị. Bạn có thể thấy Doris và George, Mary và Sherry, cũng tương tự là Ted và Marty, từng cặp một đều có cùng giá trị Rank by Age. Lưu ý rằng Doris và George cùng có xếp loại là 2 nhưng xếp loại của Mary (có giá trị Age tiếp theo) lại không phải 3 mà là 4. Nguyên nhân ở đây là Mary được trả về bản ghi thứ 4 trong tập hợp bản ghi, và hàm RANK() đã lấy số liệu đó khi thiết lập giá trị xếp loại tiếp theo trong Rank by Age

Nếu bạn muốn có một nhiều xếp loại trong tập hợp bản ghi của mình thì với từng xếp loại bạn cần đặt một nhóm cụ thể bằng cách sử dụng mệnh đề PARTITION BY trong hàm RANK. Ví dụ dưới sẽ cho thấy tác dụng khi tôi nhóm xếp loại theo Gender và sắp xếp theo Age

```
SELECT RANK() OVER (PARTITION BY Gender ORDER BY Age) AS [Partition by Gender],
      FirstName,
      Age,
      Gender
  FROM Person
```

Đây là kết quả khi chạy các hàm truy vấn trên:

Partition by Gender	FirstName	Age	Gender
1	Doris	6	F
2	Mary	11	F
2	Sherry	11	F
4	Sue	29	F
1	Larry	5	M
2	George	6	M
3	Sam	17	M
4	Ted	23	M

```
4                Marty      23          M
6                Frank      38          M
7                John       40          M
```

Bạn có thể thấy là Gioitinh là "F" được bắt đầu xếp loại từ 1 cho đến 4, sau đó bắt đầu đánh số lại từ 1 cho Gioitinh là "M"

Hàm DENSE_RANK

Hàm DENSE_RANK cũng giống như hàm RANK, tuy vậy, hàm này không cung cấp khoảng cách giữa các số xếp loại. Thay vào đó, hàm này sẽ xếp loại liên tục cho từng giá trị ORDER BY cụ thể. Với hàm DENSE_RANK, kể cả khi có hai dòng có cùng giá trị xếp loại thì dòng tiếp theo vẫn chỉ tăng thêm một giá trị so với dòng trên. Hàm DENSE_RANK có cú pháp như hàm RANK.

Đây là hàm DENSE_RANK được tôi sử dụng để xếp loại cho toàn bộ các bản ghi trong bảng Person theo trường Age

```
SELECT DENSE_RANK() OVER (ORDER BY Age) AS [Dense Rank by Age],
      FirstName,
      Age
  FROM Person
```

Đoạn mã trên sẽ xuất ra như sau:

```
Dense Rank by Age    FirstName  Age
-------------------- ---------- -----------
1                    Larry      5
2                    Doris      6
2                    George     6
3                    Mary       11
3                    Sherry     11
4                    Sam        17
5                    Ted        23
5                    Marty      23
6                    Sue        29
7                    Frank      38
8                    John       40
```

Như bạn thấy các số trong cột "Dense Rank By Age" vẫn đảm bảo tính liên tục, không hề bị ngắt quãng kể cả khi có hai dòng cùng giá trị ORDER BY và giá trị xếp loại như Ted và Marty.

Hàm NTILE

Hàm cuối cùng là hàm NTILE. Đây là hàm được sử dụng để phá vỡ tập hợp bản ghi trong một số cụ thể của các nhóm. Hàm NTILE cũng sử dụng cú pháp như các hàm ranking khác.

Trong ví dụ đầu của hàm này, tôi sẽ nhóm các bản ghi trong bảng Person thành 3 nhóm khác nhau. Tôi muốn các nhóm này dựa trên cột Age. Để làm được điều này, tôi sẽ chạy T-SQL sau:

```
SELECT FirstName,
      Age,
      NTILE(3) OVER (ORDER BY Age) AS [Age Groups]
  FROM Person
```

Đây là tập hợp kết quả của tôi từ câu lệnh T-SQL trên:

```
FirstName  Age         Age Groups
---------- ----------- --------------------
Larry      5           1
Doris      6           1
George     6           1
Mary       11          1
Sherry     11          2
Sam        17          2
Ted        23          2
Marty      23          2
Sue        29          3
Frank      38          3
John       40          3
```

Trong tập hợp kết quả đã có ở trên với 3 nhóm Age khác nhau. Nhóm đầu tiên bắt đầu từ 5 đến 11 tuổi, nhóm thứ 2 bắt đầu từ 11 đến 23 và nhóm cuối cùng là từ 29 đến 40. Hàm NTILE chỉ có tác dụng chia đều số lượng các bản ghi và đưa vào từng nhóm số. Sử dụng hàm NTILE cho từng bản ghi trong một nhóm sẽ đưa gia các xếp loại giống nhau.

Hàm NTILE là một hàm rất có ích nếu bạn chỉ muốn trả lại một nhóm cụ thể trong các bản ghi. Dưới đây là một ví dụ khi tôi muốn trả lại chỉ nhóm người có độ tuổi chung bình (Nhóm Age 2) từ ví dụ trên.

```
SELECT FirstName,
      Age,
      Age AS [Age Group]
```

```
FROM ( SELECT FirstName,
            Age,
            NTILE(3) OVER (ORDER BY Age) AS AgeGroup
       FROM Person) A
WHERE AgeGroup = 2
```

Kết quả của câu lệnh trên:

```
FirstName  Age         Age Group
---------- ----------- -----------
Sherry     11          11
Sam        17          17
Ted        23          23
Marty      23          23
```

Kết luận

Mã hóa một quy trình sắp xếp các số tuần tự trong tập hợp bản ghi được sử dụng để lấy một số trong các dòng của mã. SQL Server 2005 đã đưa ra một vài hàm ranking mới. Hy vong trong thời gian tới bạn sẽ cần xếp loại cho các tập hợp bản ghi của mình và một trong các hàm đã được giới thiệu trong bài sẽ giúp bạn hoàn thành công việc đó, nó là một việc hoàn toàn đơn giản.