《交通大数据技术》

马晓磊

交通科学与工程学院

2024年

高级SQL函数

本节大纲

- 窗口函数
- 条件语句
- 变量
- 循环
- 存储过程

窗口函数

窗口函数根据对查询结果的某些操作为每一行返回一个值。

- 允许更复杂的数据分类和排序，包括难以捉摸的分位数函数

- 基本语法：

```
SELECT ROW_NUMBER() OVER (PARTITION BY <attributes>
                          ORDER BY <attributes>) AS name
  FROM relation
```

窗口函数

下面我们来看一个简单的例子，假如说我们希望将Employee表按照性别进行聚合，比如说我希望得到的结果是：“登录名，性别，该性别所有员工的总数”，如果我们使用传统的写法，那一定会涉及到子查询，如下所示。

```
SELECT [LoginID],gender,
(SELECT COUNT(*) FROM
[AdventureWorks2012].[HumanResources].[Employee] a
WHERE a.Gender=b.Gender) AS GenderTotal
FROM [AdventureWorks2012].[HumanResources].[Employe] b
```

使用窗口函数后，如下：

```
SELECT [LoginID],gender,
COUNT(*) OVER (Partition BY gender) AS GenderTotal
FROM [AdventureWorks2012].[HumanResources].[Employe]
```

窗口函数

PostgreSQL 支持的排名函数：

ROW_NUMBER():依据组显示每一条记录在该组中出现的位置，比如：若有两个第一名。则这两个第一名在一组内排名为1、2，下一组的排序仍从1 开始，依次类推。

RANK():在结果集中每一条记录所在的排名位置，但排名可能不连续，比如：若同一组内有两个第一名，则该组内下一个名次直接跳至第三名。

DENSE_RANK():功能与Rank相似。但排名的数值是连续的，比如：若同一组内有两个第一名，则该组内下一个名次为第二名。

NTILE():依据指定的分组数量将结果集分区，并记录其在组中的位置。

窗口函数-示例

高程

State	Route	Milepost	Elevation
WA	I-5	0	24.42
WA	I-5	0.001853	24.29
WA	I-5	0.003729	24.16
WA	I-90	0	15.52
WA	I-90	0.005632	15.48
WA	I-90	0.011288	13.79

如何根据道路上连续点的高程计算每个路段的坡度?

- 对于每个点，找到同一条道路上的下一个点。
- 计算高差，然后除以里程标差。
- 分别对每条路线重复此过程

窗口函数-示例

但是对于道路上的每个点，如何找到下一个点?

- 没有窗口功能，可能会非常困难且效率低下。
- 如果 A 是道路上的当前点，则下一个点 B 在所有里程点比 A 高的点中具有最小的里程点。

- 不使用窗口函数的解决方案:

计算坡度

```
SELECT a.*, (b.Elevation-a.Elevation)/(b.Milepost-a.Milepost)/5280
AS Grade
  FROM Elevation AS a INNER JOIN Elevation AS b
    ON a.Route = b.Route
       AND b.Milepost = (SELECT MIN(c.Milepost)
                           FROM Elevation AS c
                          WHERE c.Route = a.Route
                                AND c.Milepost > a.Milepost)
```

窗口函数-示例

查询结果

State	Route	Milepost	Elevation	Grade
WA	I-5	0	24.42	-0.01329
WA	I-5	0.001853	24.29	-0.01312
WA	I-5	0.003729	24.16	NULL
WA	I-90	0	15.52	-0.00135
WA	I-90	0.005632	15.48	-0.05659
WA	I-90	0.011288	13.79	NULL

这是我想要的结果。但这是一个好的解决方案吗?

- 效率低下且缓慢
- 对于每一点，我需要查看整个表格以找到下一个点。

窗口函数-示例

使用窗口函数的解决方法：

1. 创建一个新列来显示表格的行号，在每条道路内，这些行均按Milepost排序。

```
SELECT *, ROW_NUMBER() OVER (PARTITION BY Route
                             ORDER BY Milepost) AS PointOrder
  INTO #Elev_Ordered
  FROM Elevation
```

State	Route	Milepost	Elevation	PointOrder
WA	I-5	0	24.42	1
WA	I-5	0.001853	24.29	2
WA	I-5	0.003729	24.16	3
WA	I-90	0	15.52	1
WA	I-90	0.005632	15.48	2
WA	I-90	0.011288	13.79	3

窗口函数-示例

使用窗口函数的解决方法：

2. 对于每个点，找到同一条道路上的下一个点。

3. 计算高差，然后除以里程标差。

```
SELECT a.*, (b.Elevation-a.Elevation)/(b.Milepost-a.Milepost)/5280
AS Grade
  FROM #Elev_Ordered AS a INNER JOIN #Elev_Ordered AS b
    ON a.Route = b.Route
       AND b.PointOrder = a.PointOrder + 1
```

B is the next point of A

State	Route	Milepost	Elevation	Grade
WA	I-5	0	24.42	-0.01329
WA	I-5	0.001853	24.29	-0.01312
WA	I-5	0.003729	24.16	NULL
WA	I-90	0	15.52	-0.00135
WA	I-90	0.005632	15.48	-0.05659
WA	I-90	0.011288	13.79	NULL

窗口函数

分位数函数: NTILE(n)

- 将行平均分为n组。
- 例如， NTILE(4)表示四分位数， NTILE(5)表示五分位等。
- 这不会为你提供切点的值，它只是将每一行分配给特定的分位数组。
- 对于每一行， NTILE(n)函数将返回该行所属的组（1∽n）的编号

- 能够对结果集的数据排序后。依照指定的数量把结果集分成N组，并给予每组一个组编号，分组的方式非常简单，将结果集的总记录数除以N，若有余数M，则前M组都多增一条记录，因此，并不是全部的组都有同样的记录数。

窗口函数-示例

创建CEOS的收入分位数：

```
SELECT *, NTILE(100) OVER (ORDER BY OneYrPay DESC) as Percentile
  FROM CEOs
 ORDER BY OneYrPay DESC
```

共有200位CEOS，每个百分位组中有两位

Name	Company	OneYrPay	FiveYrPay	Shares	Age	Percentile
John H Hammergren	McKesson	131.19	285.02	51.9	53	1
Ralph Lauren	Ralph Lauren	66.65	204.06	5010.4	72	1
Michael D Fascitelli	Vornado Realty	64.405	NULL	171.7	55	2
Richard D Kinder	Kinder Morgan	60.94	60.94	8582.3	67	2
David M Cote	Honeywell	55.79	96.11	21.5	59	3
George Paz	Express Scripts	51.525	100.21	47.3	57	3
Jeffery H Boyd	Priceline.com	50.185	90.3	128.2	55	4
…	…	…	…	…	…	…

窗口函数

可以在窗口上使用的常用:

- 排名函数: ROW_NUMBER(), RANK(), DENSE_RANK(), NTILE(n), etc.
- 聚合函数: AVG(), MIN(), MAX(), SUM(), COUNT(), etc.

在窗口上使用聚合函数时，在窗口中不使用ORDER BY （因为这没有意义）

单选题 1分

设置

根据不同的商品种类，按照销售单价从低到高的顺序创建排序表，结果返回product_name、product_type、sale_price和顺序ranking列，排序函数要保证排名不连续。

```
SELECT product_name,product_type,sale_price,
                    [填空1] OVER (PARTITION BY [填空2] ORDER BY sale_price) AS
ranking FROM Product
```

A
填空1:RANK()
填空2:product_name

B
填空1:DENSE_RANK()
填空2:product_name

C
填空1:RANK()
填空2:product_type

D
填空1:ROW_NUMBER()
填空2:product_type

Product（商品）表

product_id（商品编号）	product_name（商品名称）	product_type（商品种类）	sale_price（销售单价）
0001	T恤衫	衣服	500
0002	打孔器	办公用品	320
0003	运动T恤	衣服	2800
0004	菜刀	厨房用具	2800
0005	高压锅	厨房用具	6800

提交

条件语句

SQL中的CASE语句是在查询中返回条件值的一种方法。

与常规的基于集合的操作相比，它们可能较慢，但在某些情况下可能非常有用。

CASE语句的基本形式如下：

```
SELECT CASE <column name>
          WHEN <condition 1> THEN <value 1>
          WHEN <condition 1> THEN <value 1>
          ...
          ELSE <valune x>
       END
```

解释为：当列为<condition 1>时，返回<value 1>，当列为<condition 2>时，然后返回<value 2>，…，否则，返回<<value x>。

条件语句

示例：

```
SELECT Name, Section,
       Case Section
         WHEN 'CEE 412' THEN 50
         WHEN 'CEE 599' THEN 60
       END AS TotalPoints
  FROM Students
```

Name	Section
Mary Brooks	CEE 412
Christina Parker	CEE 599
Craig Price	CEE 599
Elizabeth Howard	CEE 412
Lillian Cooper	CEE 599
Evelyn Bailey	CEE 412
…	…

Name	Section	TotalPoints
Mary Brooks	CEE 412	50
Christina Parker	CEE 599	60
Craig Price	CEE 599	60
Elizabeth Howard	CEE 412	50
Lillian Cooper	CEE 599	60
Evelyn Bailey	CEE 412	50
…	…	

单选题 1分

设置

使用CASE表达式从上表得到下表的结果。sex列1表示男性，2表示女性

```
SELECT pref_name,
            SUM(CASE WHEN [填空1] THEN  [填空2]  ELSE 0 END) AS cnt_male,
            SUM(CASE WHEN sex='2' THEN population ELSE 0 END) AS
cnt_female
FROM Poptb
GROUP BY pref_name
```

Poptb（人口）表

pref_name（县名）	sex（性别）	population（人口）
德岛	1	60
德岛	2	40
香川	1	100
香川	2	100
爱媛	1	100
爱媛	2	50

pref_name（县名）	cnt_male（男性人口）	cnt_female（女性人口）
德岛	60	40
香川	100	100
爱媛	100	50

A 填空1:sex=1
填空2:population

B 填空1:sex='1'
填空2:population

C 填空1:sex='1'
填空2:sex

D 填空1:sex=1
填空2:pref_name

提交

SQL中的变量

在SQL中，局部变量是一个对象，可以保存特定类型的单个数据值。

- 声明变量的句法：

```
DECLARE @variable_name <data type>
SET @variable_name = <some value>
```

- 或者：

```
DECLARE @variable_name <data type> = <some value>
```

SQL中的变量-示例

Player (Name, Salary, Height, Weight, Team)

问题:找到薪水最高的球员的名字.

- 使用子查询的解决方案:

```
SELECT name, salary
  FROM player
 WHERE salary = (SELECT MAX(salary) FROM player)
```

- 使用局部变量的解决方案

```
DECLARE @max_salary INT = (SELECT MAX(salary) FROM player)

SELECT name, salary
  FROM player
 WHERE salary = @max_salary
```

name	salary
Peyton Manning	15000000.00

SQL中的循环语句

SQL中有几种循环类型，我们将介绍WHILE循环

SQL不是一种常规的编程语言，实际上似乎可以用循环解决的大多数事情都可以使用SQL“基于集合”的方法来解决。

当常规查询可以执行时，请不要使用循环（速度慢且占用大量资源）

IF/BREAK循环

创建一个包括10个随机数的表

1. 创建空表并计数:

```
CREATE TABLE #temp(ID INT, RandNum DECIMAL(5,4))
DECLARE @counter INT = 1
```

2. 利用While循环插入数据:

```
WHILE @counter <= 10
  BEGIN
    INSERT INTO #temp VALUES(@counter, RAND())
    SET @counter = @counter + 1
  END
```

随机数生成函数

IF/BREAK循环

当随机数总和超过3时停止循环

```
CREATE TABLE #temp(ID INT, RandNum DECIMAL(5,4))
DECLARE @counter INT = 1

WHILE @counter <= 10
  BEGIN
    INSERT INTO #temp VALUES(@counter, RAND())
    SET @counter = @counter + 1
    IF (SELECT SUM(RandNum) FROM #temp) > 3.0 BREAK
    ELSE CONTINUE
  END
```

IF/BREAK循环

前两个查询的结果：

ID	RandNum
1	0.9953
2	0.8091
3	0.9167
4	0.2714
5	0.1149
6	0.9743
7	0.7772
8	0.8559
9	0.7972
10	0.4414

对比

ID	RandNum
1	0.2238
2	0.7479
3	0.4861
4	0.9626
5	0.5082
6	0.1083

单选题 1分

设置

利用随机函数RAND和函数FLOOR，产生30个1到20之间的随机整数，使用WHILE语句显示这30个随机数

```
CREATE TABLE #tmp(x INT)
DECLARE @i INT =1
WHILE  [填空1]
        BEGIN
                INSERT INTO #tmp
VALUES(FLOOR(RAND()*20)+1)
                SET [填空2]
        END
```

A
填空1:i<=30
填空2:i=i+1

B
填空1:@i<=30
填空2:i=i+1

C
填空1:@i<30
填空2:@i=@i+1

D
填空1:@i<=30
填空2:@i=@i+1

提交

存储过程

一组在SQL中保存的命令，可以随时轻松执行，甚至可以输入参数值（例如函数）

Why?

- 提高性能（存储过程是预编译的）
- 降低网络开销（只需提供存储过程名及参数）
- 便于进行代码移植（仅对存储过程修改）
- 更强的安全性（权限限制，无法看到表内容）

Why Not?

存储过程需要专门的数据库开发人员进行维护

- 设计逻辑变更，修改存储过程没有SQL灵活

存储过程-示例

- 给学生额外的学分：

```
CREATE PROCEDURE ExtraCredit
  @student_name VARCHAR(10),
  @extra_credit FLOAT
AS
UPDATE #student
   SET Grade = Grade + @extra_credit
 WHERE name = @student_name
RETURN(1)
```

程序名称

输入参数

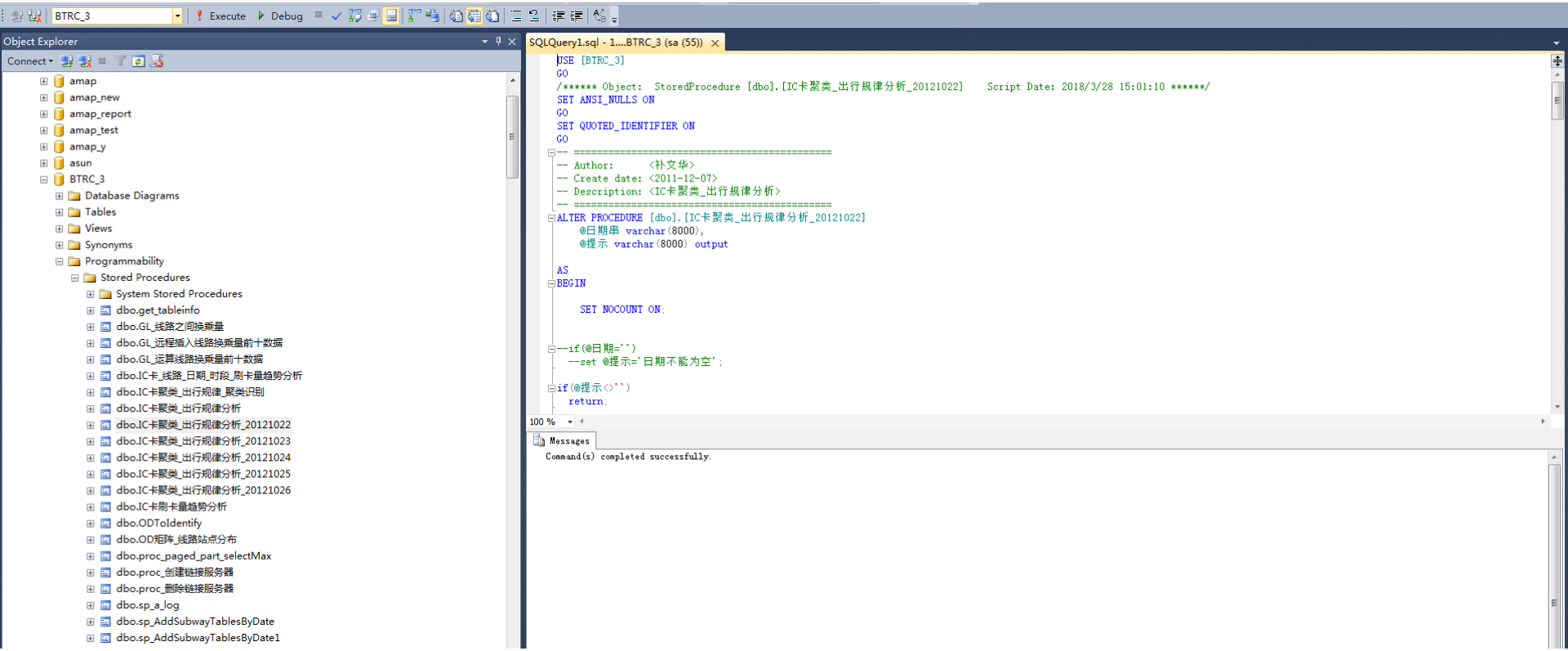
过程命令

返回值（可选）

- 执行以下步骤：

```
EXEC ExtraCredit @student_name = 'Peter', @extra_credit = 5.0
```

存储过程-示例


BTRC_3
Execute
Debug
Object Explorer
Connect
amap
amap_new
amap_report
amap_test
amap_y
asun
BTRC_3
Database Diagrams
Tables
Views
Synonyms
Programmability
Stored Procedures
System Stored Procedures
dbo.get_tableinfo
dbo.GL_线路之间换乘量
dbo.GL_远程插入线路换乘量前十数据
dbo.GL_运算线路换乘量前十数据
dbo.IC卡_线路_日期_时段_刷卡量趋势分析
dbo.IC卡聚类_出行规律_聚类识别
dbo.IC卡聚类_出行规律分析
dbo.IC卡聚类_出行规律分析_20121022
dbo.IC卡聚类_出行规律分析_20121023
dbo.IC卡聚类_出行规律分析_20121024
dbo.IC卡聚类_出行规律分析_20121025
dbo.IC卡聚类_出行规律分析_20121026
dbo.IC卡刷卡量趋势分析
dbo.ODToIdentify
dbo.OD矩阵_线路站点分布
dbo.proc_paged_part_selectMax
dbo.proc_创建链接服务器
dbo.proc_删除链接服务器
dbo.sp_a_log
dbo.sp_AddSubwayTablesByDate
dbo.sp_AddSubwayTablesByDate1
SQLQuery1.sql - 1....BTRC_3 (sa (55))
USE [BTRC_3]
GO
/****** Object: StoredProcedure [dbo].[IC卡聚类_出行规律分析_20121022] Script Date: 2018/3/28 15:01:10 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- ===
-- Author: <补文华>
-- Create date: <2011-12-07>
-- Description: <IC卡聚类_出行规律分析>
-- ===
ALTER PROCEDURE [dbo].[IC卡聚类_出行规律分析_20121022]
@日期串 varchar(8000),
@提示 varchar(8000) output
AS
BEGIN
SET NOCOUNT ON;
--if(@日期='')
--set @提示='日期不能为空';
if(@提示<>'')
return;
100 %
Messages
Command(s) completed successfully.