

بنام خدا

پایگاه داده ۲

Structured Query Language (SQL)

بصیری

دانشکده برق و کامپیوتر

دانشگاه صنعتی اصفهان

قابلیت های پایگاه داده رابطه ای (عملیاتی)

■ امکانات

□ استفاده از توابع پیشرفته SQL

■ مانند Analytic Functions

□ نوشتن تابع

□ نوشتن پراسیجر

□ ...

■ هدف

□ ذخیره، بازیابی و پردازش داده ها

جدول Products زیر را در نظر بگیرید

P_Id	ProductName	UnitPrice	UnitsInStock	UnitsOnOrder
1	Jarlsberg	10.45	16	15
2	Mascarpone	32.56	23	
3	Gorgonzola	15.67	9	20

اگر بخواهیم ارزش موجود در سفارش و انبار را داشته باشیم:

```
SELECT ProductName, UnitPrice * (UnitsInStock + UnitsOnOrder)  
FROM Products;
```

روش فوق چه مشکلی دارد؟

جایگذاری مقدار برای مقادیر نال

SQL Server ■

ISNULL(string1, replace_with) □

```
SELECT ProductName,  
       UnitPrice * (UnitsInStock + ISNULL(UnitsOnOrder, 0))  
FROM Products
```

Oracle ■

NVL(string1, replace_with) □

```
SELECT ProductName,  
       UnitPrice * (UnitsInStock + NVL(UnitsOnOrder, 0))  
FROM Products
```

مثال: یافتن رکورد قبلی و بعدی مربوطه

■ SQL Server و Oracle یکی است

■ رکورد قبلی

LAG (expression [, offset [, default]])
OVER ([query_partition_clause] order_by_clause)

■ رکورد بعدی

LEAD (expression [, offset [, default]])
OVER ([query_partition_clause] order_by_clause)

جدول employees زیر را در نظر بگیرید

employee_number	last_name	first_name	salary	dept_id
12009	Sutherland	Barbara	54000	45
34974	Yates	Fred	80000	45
34987	Erickson	Neil	42000	45
45001	Parker	Sally	57500	30
75623	Gates	Steve	65000	30

```
SELECT dept_id, last_name, salary, LAG (salary,1) OVER (ORDER BY salary)
AS lower_salary FROM employees;
```



dept_id	last_name	salary	lower_salary
45	Erickson	42000	NULL
45	Sutherland	54000	42000
30	Parker	57500	54000
30	Gates	65000	57500
45	Yates	80000	65000

یافتن رکورد قبلی و بعدی مربوطه

■ یافتن اولین حقوق کمتر از حقوق هر فرد، در هر دپارتمان

■ `SELECT dept_id, last_name, salary,
LAG (salary, 1, 0) OVER (PARTITION BY deptno ORDER BY
salary) AS sal_prev FROM employees;`

توابع مهم مربوط به رشته در SQL Server

Function	Example	Result	Description
CONCAT	CONCAT('WWW','com')	WWW.com	Adds two or more strings together
LEN	LEN('com')	3	Returns the length of a string
LOWER	LOWER('SQL!')	sql!	Converts a string to lower-case
LTRIM	LTRIM(' SQL !')	SQL !	Removes leading spaces from a string
PATINDEX	PATINDEX('%sc%', 'W3Sc.com')	3	Returns the position of a pattern in a string
REPLACE	REPLACE('TSQL', 'T', 'M')	MSQL	Replaces all occurrences of a substring within a string, with a new substring
RTRIM	RTRIM('SQL ')	SQL	Removes trailing spaces from a string
SUBSTRING	SUBSTRING('SQL', 1, 2)	SQ	Extracts some characters from a string
TRIM	TRIM(' SQL ')	SQL	Removes leading and trailing spaces (or other specified characters) from a string
UPPER	UPPER('sql')	SQL	Converts a string to upper-case
REPLICATE	REPLICATE('SQ', 2)	SQSQ	Repeats a string a specified number of times
RIGHT	RIGHT('SQL', 2)	QL	Extracts a number of characters from a string (starting from right)
LEFT	LEFT('SQL', 2)	SQ	Extracts a number of characters from a string (starting from left)

توابع مهم مربوط به رشته در Oracle

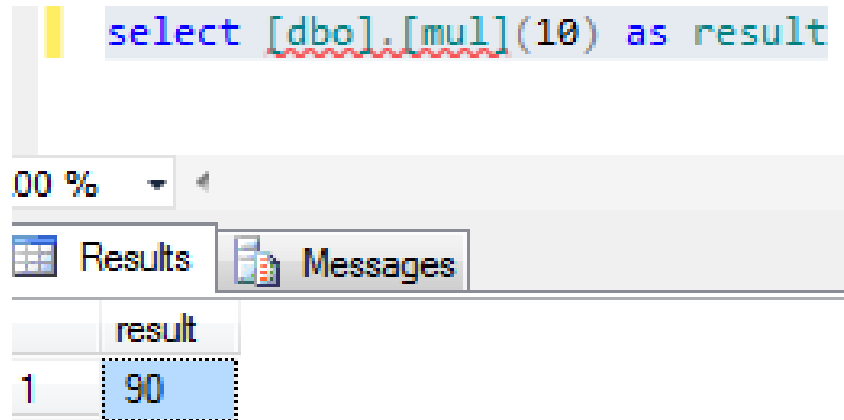
Function	Example	Result	Purpose
CONCAT	CONCAT('A','BC')	'ABC'	Concatenate two strings and return the combined string
INSTR	INSTR('This is a playlist', 'is')	3	Search for a substring and return the location of the substring in a string
LENGTH	LENGTH('ABC')	3	Return the number of characters (or length) of a specified string
LOWER	LOWER('Abc')	'abc'	Return a string with all characters converted to lowercase
LPAD	LPAD('ABC',5,'*')	'**ABC'	Return a string that is left-padded with the specified characters to a certain length.
LTRIM	LTRIM(' ABC ')	'ABC '	Remove spaces or other specified characters in a set from the left end of a string
REPLACE	REPLACE('JACK AND JOND','J','BL');	'BLACK AND BLOND'	Replace all occurrences of a substring by another substring in a string
RPAD	RPAD('ABC',5,'*')	'ABC**'	Return a string that is right-padded with the specified characters to a certain length.
RTRIM	RTRIM(' ABC ')	' ABC'	Remove all spaces or specified character in a set from the right end of a string
SUBSTR	SUBSTR('Oracle Substring', 1, 6)	'Oracle'	Extract a substring from a string
TRIM	TRIM(' ABC ')	'ABC'	Remove the space character or other specified characters either from the start or end of a string
UPPER	UPPER('Abc')	'ABC'	Convert all characters in a specified string to uppercase

تابع اسکالر: این توابع تک خروجی هستند

```
CREATE FUNCTION [ owner_name. ] function_name  
    ( @parameter_name [AS] data_type )  
RETURNS data_type  
AS  
BEGIN  
  
    function_body  
  
    RETURN scalar_expression  
END
```

تابعی که n را دریافت کرده و $n*(n-1)$ را خروجی می دهد.

```
create function [dbo].[mul] (@n int)
returns bigint
AS
begin
    declare @r bigint
    set @r=(@n-1)*@n
    return @r
end
```



The screenshot shows a SQL query window with the following text: `select [dbo].[mul](10) as result`. Below the query window, the 'Results' tab is active, displaying a table with one column named 'result' and one row containing the value '90'.

	result
1	90


توابع table-valued: یک جدول خروجی می دهند.

```
CREATE FUNCTION [ owner_name. ] function_name  
    ( @parameter_name [AS] data_type )  
RETURNS table  
AS  
BEGIN
```

function_body

```
    RETURN (select)  
END
```

تفاوت اصلی این تعریف با توابع اسکالر در این است که نوع خروجی یک جدول تعریف شده است. محدودیتی که روی این نوع تعریف وجود دارد این است که خروجی تابع باید توسط یک دستور **select** ایجاد شود.



In this example we will create an inline table-valued function that will retrieve records of all the students whose DOB is less than the DOB passed to the function.

```
CREATE FUNCTION BornBefore
(
    @DOB AS DATETIME
)
RETURNS TABLE
AS
BEGIN
    RETURN
    SELECT * FROM student
    WHERE DOB < @DOB
END
```



We will pass a date to "BornBefore" function. The student records retrieved by the function will have a DOB value lesser than that date.

```
SELECT
    name, gender, DOB
FROM
    dbo.BornBefore('1980-01-01')
ORDER BY
    DOB
```

سوال

- تابعی بنویسید که یک رشته به طول حداکثر ۲۰ و یک الگو به طول حداکثر ۳ دریافت نماید. این تابع باید تمام تکرارهای الگو را از رشته بزرگتر حذف کرده و رشته باقی مانده را در برگرداند. اگر با حذف یکباره دوباره رشته شامل آن الگو بود باید مجدد حذف شود.



Procedure

- A stored procedure is a set of Structured Query Language (SQL) statements with an assigned name, which are stored in a relational database management system as a group, so it can be reused and shared by multiple programs.
- Stored procedures can access or modify data in a database, but it is not tied to a specific database or object, which offers a number of advantages.



Benefits of using stored procedures

- A stored procedure provides an important layer of security between the user interface and the database. It supports security through data access controls because end users may enter or change data, but do not write procedures.
- A stored procedure preserves data integrity because information is entered in a consistent manner. It improves productivity because statements in a stored procedure only must be written once.



Benefits of using stored procedures (Cont.)

- Stored procedures offer advantages over embedding queries in a graphical user interface (GUI). Since stored procedures are modular, it is easier to troubleshoot when a problem arises in an application.
- Stored procedures are also tunable, which eliminates the need to modify the GUI source code to improve its performance. It's easier to code stored procedures than to build a query through a GUI.

نوآشتن پراسیجر

```
CREATE PROCEDURE [ owner_name. ] procedure_name  
    ( @parameter_name [AS] data_type )  
AS  
BEGIN  
  
    procedure_body  
  
END
```

سوال

پراسیجری بنویسید که از جدول مشتریان بنام Customer، که به اشتباه برای یک مشتری ممکن است حاوی چند رکورد باشد، صرفاً اولین رکورد را بر اساس فیلد تاریخ ثبت (RegDat) در جدول Customer ذخیره کرده و رکوردهای اضافه را حذف نماید.

Customer

Natcod	Name	Job	RegDat

سوال

- جدولی بنام Turn_Over در نظر بگیرید که تراکنشهای سپرده های بانک را در خود ذخیره می کند. پراسیجری بنویسید که جدولی بنام Factdeptrn را پر کند که حاوی فیلد مانده بعد از تراکنش، علاوه بر فیلدهای جدول Turn_Over می باشد.

Turn_Over

Dep_Id	Trn_Time	Trn_over
1022	2018-06-15 14:00	100
1022	2018-06-15 14:28	-50
1022	2018-06-16 14:58	25
1067	2019-07-18 23:32	300

Factdeptrn

Dep_Id	Trn_Time	Trn_over	Balance
1022	2018-06-15 14:00	100	100
1022	2018-06-15 14:28	-50	50
1022	2018-06-16 14:58	25	75
1067	2019-07-18 23:32	300	300

سوال

■ آیا بانک می تواند با یک کوئری ساده به نتیجه دلخواه خود برسد؟

```
Query1.sql - (loc...siri-PC\Basiri (54)) * X
SELECT [id]
      ,[date]
      ,[turnover]
      ,(select sum(t.turnover) from [Test].[dbo].[turnover] t
        where t.date <= [Test].[dbo].[turnover].date and
              t.id = [Test].[dbo].[turnover].id) bal
FROM [Test].[dbo].[turnover];

select id, date, turnover, sum(turnover)
over(partition by id order by date rows between unbounded preceding and current row) bal
from [Test].[dbo].[turnover]
```

Results Messages

id	date	turnover	bal
1	2019-04-26 13:54:08.053	100	100
1	2019-04-26 13:54:14.337	100	200
1	2019-04-26 13:54:24.967	-100	100
1	2019-04-26 13:54:33.113	-10	90
1	2019-04-26 13:54:42.643	-20	70
1	2019-04-26 13:54:51.217	200	270

■ فرض کنیم می خواهیم ۱۰۰ مشتری برتر بانک از لحاظ مانده را بدست آوریم!

■ فرض کنیم می خواهیم ۱۰۰ مشتری برتر بانک را به تفکیک ماه از اول سال ۹۹ تا کنون بدست آوریم!!

مثال

بانکی را در نظر بگیرید که مشتریان در آن سپرده گذاری کرده اند و برای هر مشتری ممکن است تعدادی سپرده (**Deposit**) وجود داشته باشد. اطلاعات پایه ای سپرده مانند شماره سپرده و نوع ارز آن، شماره مشتری و... را در جدولی بنام **dimdeposit** ذخیره کرده ایم.

هر سپرده در یک تاریخ افتتاح شده و با واریز و برداشتهایی که مشتری روی حسابش انجام می دهد می تواند مانده سپرده اش را تغییر دهد. فرض کنید در یک جدول بنام **factdeposit** مانده تمام سپرده ها را در هر روز نگهداری می کنیم. بر اساس این اطلاعات میانگین موجودی هر سپرده را طی سال ۹۹ در جدولی ذخیره کنید.

حل مثال

```
CREATE OR REPLACE PROCEDURE "PROC1" is
currdate date;
begin
currdate := to_date('990101', 'yymmdd', 'nls_calendar=persian');
delete from temp1; delete from temp2;
delete from temp3; delete from Final;
commit;
insert into temp3
  select d.Depositkey,
  0 active_date_count,
  0 avg_balance
from dimdeposit d;
```


ادامہ حل مثال

```
while currrdate <= to_date('991229', 'yymmdd', 'nls_calendar=persian') loop

    insert into temp1

        select f.Depositkey, 1 active_date_count, f.bal avg_balance

        from factdeposit f where f.effdate = currrdate;

commit;

insert into temp2

select p.Depositkey,

    case when t.active_date_count is null then p.active_date_count

else p.active_date_count + 1 end active_date_count,

case when t.active_date_count is null then


    p.avg_balance

Else ((p.active_date_count*p.avg_balance)+t.avg_balance)/ p.active_date_count+1)

end avg_balance

from temp3 p left outer join temp1 t

on t.Depositkey = p.Depositkey;
```



```
        truncate temp3; commit;

insert into temp3

        select * from temp2;

truncate temp1; truncate temp2;

commit;

currdate := currdate + 1;

end loop;

insert into Final

        select t.Depositkey,
        t.active_date_count,
        t.avg_balance,
        from temp3 t;

commit;

end PROC1;
```

some useful links

- <https://www.red-gate.com/simple-talk/databases/sql-server/t-sql-programming/sql-server/introduction-to-t-sql-window-functions/>
- <https://www.codeproject.com/Articles/34142/Understanding-Set-based-and-Procedural-approaches>
- <https://www.aspsnippets.com/Articles/Using-Cursor-in-SQL-Server-Stored-Procedure-with-example.aspx>