In the name of god

SQL Server

Function & Trigger

Amirkabir University of Technology

Behnaz Motavali

bs.motavali@yahoo.com

Function

زیر برنامه ای است که با دریافت صفر ، یک یا چند پارامتر ، یک و فقط یک مقدار را برمی گرداند.

در SQL Server 2005 توابع در حالت های زیر طبقه بندی شده اند :

System Functions

- Aggregate Functions
- Configuration Functions
- Cryptographic Functions
- Cursor Functions
- Date & Time Functions
- Mathematical Functions
- Metadata Functions
- Row Set Functions
- Ranking Functions
- Security Functions
- String Functions
- System Statistical Functions
- Text and Image Functions

User Defined Function

- Aggregate Functions
- Scalar-Valued Functions
- Table-Valued Functions
  - Inline Functions
  - Multi Statement Functions

Function

توابع سیستمی توابعی هستند که در دسته بندی های گوناگون جهت مصارف گوناگون ساخته شده اند

GetDate	تاریخ و زمان جاری را از سرور بر می گرداند
Day	شماره روز را از تاریخ داده شده بر می گرداند
Month	شماره ماه را از تاریخ داده شده بر می گرداند
Year	شماره سال را از تاریخ داده شده بر می گرداند
DateName	نام متناظر با پارامتر داده شده و تاریخ داده شده را بر می گرداند
DatePart	بخش خواسته شده از تاریخ داده شده را بر می گرداند
DateAdd	تاریخ جدید بر اساس تاریخ داده شده و اختلاف داده شده را بر می گرداند
DateDiff	اختلاف دو تاریخ را بر اساس بخش خواسته شده بر می گرداند

Function

برای ایجاد توابع User Defined به تر تیب زیر عمل می کنیم

دسته اول : Aggregate Functions

این دسته از توابع توسط زبانهای .Net مانند C# و یا VB.Net قابل پیاده سازی می باشند.(مراجعه به پیوست)

دسته دوم Scalar Functions

خروجی این توابع یک و فقط یک مقدار خواهد بود . قالب کلی تعریف آن به صورت زیر است :

```
Create   Function  نام
                   (فهرست پارامترها)
Returns  نوع داده ای
As
         Begin
                  ...
         End
```

چند نکته :

۱- توابع اسکالر ، امکان فراهم سازی خروجی به شکل یک Result Set را ندارند

۲- امکان فراخوانی بعضی دستورات در توابع مثل اجرای Stored Procedure وجود ندارد .

Function

مثال :

```
USE G3Q1
GO
Create    Function   Sum(
                              @FirstNumber      Int ,
                              @SecondNumber   Int
                              )
          Returns      BigInt
As
   Begin
          Return    @FirstNumber  + @SecondNumber
   End
--Test
GO
Select dbo.Sum(10,20)
```

Function

مثال :

```
CREATE FUNCTION dbo.CtoF(Celsius FLOAT)
        RETURNS FLOAT
RETURN (Celsius * 1.8) + 32
```

```
SELECT Name, CtoF(BoilingPoint)
  FROM Elements
```

Function

Table-Valued User-Defined Functions : دسته سوم

- User-defined functions that return a table data type can be powerful alternatives to views.
- A table-valued user-defined function can be used where table or view expressions are allowed in Transact-SQL queries.
- The table returned by a user-defined function can be referenced in the FROM clause of a Transact-SQL statement, but stored procedures that return result sets cannot.
- Table-Valued User-Defined Functions:
  - InLine Table Valued Functions (ITVF)
  - Multi Statement Table Valued Functions (MSTVF)

Function

InLine Table Valued Functions

توابع InLine از لحاظ ساختاری مشابه View ها هستند ، با این تفاوت که به عنوان ورودی پارامتر می پذیرند .

قالب کلی تعریف آن به صورت زیر است :

```
Create    Function  نام
                    (فهرست پارامترها)
Returns   Table
As
          Return
                    (Select
                              .....
                    )
```

Function

مثال :

```
CreateFunction   Quantity(   @ID        Int   )
   Returns Table
   As  Return
       (Select       Item.Title  As  Item ,      Color.Title  As  Color ,    Inventory.Quantity  As
              Quantity
              From          Item
                            Inner Join
                     Inventory
                            On     Item.ID         =    Inventory.Item_ID
                            Inner Join
                     Color
                            On     Color.ID    = Inventory.Color_ID
              Where
              Inventory.ID =   @ID
       )
-- Test
Select *    From      Quantity(3)
```

Function

Multi Statement Table Valued Functions

توابع Multi Statement نیز دارای خروجی از نوع Result Set می باشند و معمولاً در From مورد استفاده قرار می گیرند .

قالب کلی تعریف آن به صورت زیر است :

Create Function ***نام***

(فهرست پارامترها)

Returns ***نام خروجی*** **Table** (***فهرست ستونها***)

As

Begin

…

Return

End

Inline vs. Multi-statement Table Valued Functions

Inline Table-Valued Functions

An Inline Table-Valued Function created by this command:

```
CREATE FUNCTION datesales (@deadline as datetime)
RETURNS TABLE
AS
RETURN ( SELECT *
FROM sales WHERE ord_date > @deadline)
```

and called by this sequence:

```
USE PUBS
GO
select * from datesales('09/13/1994')
```

Inline vs. Multi-statement Table Valued Functions

will yield the following table:

stor_id	ord_num	ord_date	qty	payterms	title_id
6380	6871	09/14/94	5	Net 60	BU1032
7067	D4482	09/14/94	10	Net 60	PS2091
7131	N914008	09/14/94	20	Net 30	PS2091
7131	N914014	09/14/94	25	Net 30	MC3021
8042	423LL922	09/14/94	15	ON invoice	MC3021
8042	423LL930	09/14/94	10	ON invoice	BU1032

Inline vs. Multi-statement Table Valued Functions

Multi-statement Table-Valued Function

This UDS also returns a resultset, like the Inline variety UDF, but with a much more powerful result. The Multi-statement UFD can actually create a temporary table, specifying the fields, their type and characteristics.

```
CREATE FUNCTION datesales2 (@deadline datetime)
RETURNS @table TABLE (
stor_id varchar(6) null,
ord_num varchar(8) null,
ord_date datetime null,
qty int, payterms varchar(20),
title_id varchar(6))
AS
BEGIN
INSERT @table
SELECT *
FROM sales
WHERE ord_date > @deadline
RETURN
END
```

Inline vs. Multi-statement Table Valued Functions

When the function is called by this sequence:

```
USE PUBS
GO
select * from datesales2('09/13/1994')
```

The following resultset is displayed and available. (It is the same set displayed in the Inline UDF, but created with the temporary table.)

stor_id	ord_num	ord_date	qty	payterms	title_id
6380	6871	09/14/94	5	Net 60	BU1032
7067	D4482	09/14/94	10	Net 60	PS2091
7131	N914008	09/14/94	20	Net 30	PS2091
7131	N914014	09/14/94	25	Net 30	MC3021
8042	423LL922	09/14/94	15	ON invoice	MC3021
8042	423LL930	09/14/94	10	ON invoice	BU1032

Difference between Stored procedure and Functions

Functions
---
1) can be used with Select statement
2) Not returning output parameter but returns Table variables
3) You can join UDF
4) Cannot be used to change server configuration
5) Cannot be used with XML FOR clause
6) Cannot have transaction within function

Stored Procedure
---
1) have to use EXEC or EXECUTE
2) return output parameter
3) can create table but won't return Table Variables
4) you can not join SP
5) can be used to change server configuration
6) can be used with XML FOR Clause
7) can have transaction within SP

Trigger ها

زیر برنامه ای که بصورت خودکار و در هنگام رخ دادن رویداد خاصی ، اجرا می شود ، Trigger نامیده می شود

دو نوع Trigger در SQL Server وجود دارد :

۱- DDL Trigger

۲- DML Trigger

DDL Trigger ها اکثراً هنگامی اجرا می شوند ، که رویدادی سبب ایجاد تغییر در ساختار بانک اطلاعاتی یا اشیا در بانک اطلاعاتی شود .

DML Trigger ها در هنگام رخ دادن دستورات DML ، (Insert , Update , Delete) روی جداول و View ها اجرا می شوند . DML Trigger ها براساس زمان رخ داد خود به دسته های زیر تقسیم می شوند :

۱- Instead Of Trigger

۲- After Trigger

Instead of / After Triggers

Instead of Trigger همانطور که از نامش پیداست ، به جای هر یک از عملیات , Insert Update , Delete نشسته و در واقع در زمان وقوع هر یک از دستورات فوق ، به جای آن فعال می شود و عملیات مورد نظر را به شیوه خود انجام می دهد .

After Trigger ها زمانی وارد عمل می شوند ، که عملیات مورد نظر انجام شده و نیازی به اعمال آن توسط Trigger نیست . در این هنگام این نوع Trigger به منظور اعمال هدف خود ، اجرا می شود.

کاربرد Trigger

تعدادی از کاربردهای Trigger ها عبارتنداز :

۱- رسیدن به هدف جامعیت داده ها در جایی که Constraint ها کارایی لازم را ندارند .

۲- واقعه نگاری Logging

۳- کنترل و بومی سازی خطا ها در زمان اجرای دستورات DML

و ...

ایجاد Trigger ها

برای ایجاد Trigger از قالب زیر استفاده می شود :

Create Trigger *نام*

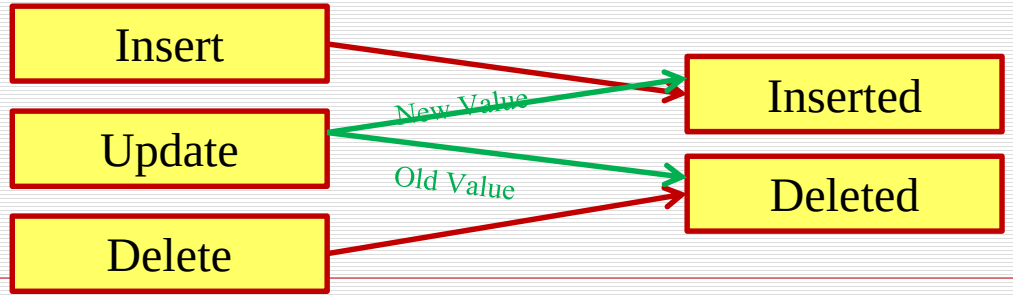
On *نام جدول یا View*

{After | Instead Of} [Insert] [,] [Update] [,] [Delete]

As Begin

End

در یک Trigger از دو جدول Inserted و Deleted برای دسترسی به رکوردهایی که تحت تاثیر قرار گرفته اند ، می توان استفاده نمود .



ایجاد Trigger ها

جداول Inserted و Deleted در یک تراکنش از نوع Auto-Commit از ابتدا در دسترس می باشند ، SQL Server به منظور مدیریت تراکنش های Auto Commit این دو جدول را در Log File خود بوجود می آورد و پس از خاتمه تراکنش آنها را پاک می کند ، این دو جدول دارای ساختاری معادل با جدول یا View مورد نظر می باشند .

نکته : در صورتیکه دستورات Insert , Update , Delete با اجرای خود ، چند رکورد را تحت تاثیر قرار دهند ، Trigger به ازای تمامی رکوردها ، یک بار فعال می شود ، بدیهی است که جداول Inserted و Updated نیز به تناسب دستور اجرایی ، شامل چندین رکورد خواهند بود . به جدول زیر دقت کنید :

	Inserted Table Rows	Deleted Table Rows
Insert	**>0**	**0**
Update	**>0**	**>0**
Delete	**0**	**>0**

Trigger مثال

```
CREATE TRIGGER [dbo].[scoretg] ON [dbo].[score]
INSTEAD OF INSERT AS
Begin
    DECLARE @g REAL;
    SELECT @g=inserted.grade FROM inserted;
    INSERT INTO score(grade,typeN) VALUES(@g,CASE WHEN
            @g<10 THEN 'B' else 'A' end)
End
```

Create DML Trigger

•**Trigger on an INSERT, UPDATE, or DELETE statement to a table or view (DML Trigger)**

```
CREATE TRIGGER [ schema_name . ]trigger_name
 ON { table | view }
[ WITH <dml_trigger_option> [ ,...n ] ]
{ FOR | AFTER | INSTEAD OF }
{ [ INSERT ] [ , ] [ UPDATE ] [ , ] [ DELETE ] }
[ WITH APPEND ]
[ NOT FOR REPLICATION ]
AS { sql_statement [ ; ] [ ,...n ] | EXTERNAL NAME <method specifier [ ; ] > }
<dml_trigger_option> ::= [ ENCRYPTION ] [ EXECUTE AS Clause ]
<method_specifier> ::= assembly_name.class_name.method_name
```

Create DDL Trigger

- **Trigger on a CREATE, ALTER, DROP, GRANT, DENY, REVOKE, or UPDATE STATISTICS statement (DDL Trigger)**

```
CREATE TRIGGER trigger_name
ON { ALL SERVER | DATABASE }
[ WITH <ddl_trigger_option> [ ,...n ] ]
{ FOR | AFTER } { event_type | event_group } [ ,...n ]
AS { sql_statement [ ; ] [ ,...n ] | EXTERNAL NAME < method specifier > [ ; ] }
<ddl_trigger_option> ::= [ ENCRYPTION ] [ EXECUTE AS Clause ]
<method_specifier> ::= assembly_name.class_name.method_name
```

Create Logon Trigger

- **Trigger on a LOGON event (Logon Trigger)**

```
CREATE TRIGGER trigger_name
ON ALL SERVER
[ WITH <logon_trigger_option> [ ,...n ] ]
{ FOR| AFTER } LOGON
AS { sql_statement [ ; ] [ ,...n ] | EXTERNAL NAME < method specifier > [ ; ] }
<logon_trigger_option> ::= [ ENCRYPTION ] [ EXECUTE AS Clause ]
<method_specifier> ::= assembly_name.class_name.method_name
```

ALTER Trigger

- DML Trigger

ALTER TRIGGER schema_name.trigger_name

ON (table | view)

[WITH <dml_trigger_option> [,...n]]

(FOR | AFTER | INSTEAD OF) { [DELETE] [,] [INSERT] [,] [UPDATE] }

[NOT FOR REPLICATION]

AS { sql_statement [;] [...n] | EXTERNAL NAME <method specifier> [;] }

<dml_trigger_option> ::= [ENCRYPTION] [<EXECUTE AS Clause>]

<method_specifier> ::= assembly_name.class_name.method_name

ENABLE/DISABLE Trigger

- Enables a DML, DDL, or logon trigger.

ENABLE TRIGGER { [schema_name .] trigger_name [,...n] | ALL }

ON { object_name | DATABASE | ALL SERVER } [;]

- Disable Trigger

DISABLE TRIGGER { [schema_name .] trigger_name [,...n] | ALL }

ON { object_name | DATABASE | ALL SERVER } [;]

- **Example:**

```
USE G2T2;
GO
DISABLE TRIGGER Person.uAddress ON Person.Address;
GO
ENABLE Trigger Person.uAddress ON Person.Address;
GO
```

APPENDIX

Creating a User-Defined Aggregate

The process of creating a user-defined aggregate function has two steps:
Step 1: Implement the ODCIAggregate interface:

```
CREATE TYPE SpatialUnionRoutines(
  STATIC FUNCTION ODCIAggregateInitialize( ... ) ...,
MEMBER FUNCTION ODCIAggregateIterate(...) ... ,
MEMBER FUNCTION ODCIAggregateMerge(...) ...,
MEMBER FUNCTION ODCIAggregateTerminate(...)
);

CREATE TYPE BODY SpatialUnionRoutines IS
...
END;
```

Step 2: Create the User-Defined Aggregate:

```
CREATE FUNCTION SpatialUnion(x Geometry) RETURN
Geometry AGGREGATE USING SpatialUnionRoutines;
```

SQL Aggregate Functions

SQL aggregate functions return a single value, calculated from values in a column.

Useful aggregate functions:

- AVG() - Returns the average value
- COUNT() - Returns the number of rows
- FIRST() - Returns the first value
- LAST() - Returns the last value
- MAX() - Returns the largest value
- MIN() - Returns the smallest value
- SUM() - Returns the sum

Configuration Functions

The following scalar functions return information about current configuration option settings:

- @@DATEFIRST
- @@OPTIONS
- @@REMSERVER
- @@TEXTSIZE
- …