Stored Procedures

Friday, May 26, 2017 4:07 PM

What is a Procedure

A unit of 2 or more lines of code executed for a unique process.

Statements NOT permitted in SPs

- Not all Transact-SQL statements are permitted in stored procedures
- In particular, the following list are not permitted

Statements not permitted	
CREATE AGGREGATE	CREATE RULE
CREATE DEFAULT	CREATE SCHEMA
CREATE or ALTER FUNCTION	CREATE or ALTER TRIGGER
CREATE or ALTER PROCEDURE	CREATE or ALTER VIEW
SET PARSEONLY	SET SHOWPLAN_ALL
SET SHOWPLAN_TEXT	SET SHOWPLAN_XML
USE databasename	

System Stored Procedures

System stored procedures are "special" in that you can execute them from within any database without needing to specify the master database as part of their name. They are typically used for administrative tasks that relate to configuring servers, databases, and objects or for retrieving information about them. System stored procedures are created within the sys schema. Examples of system stored procedures are sys.sp_configure, sys.sp_addmessage, and sys.sp_executesql.

System Extended Stored Procedures

System extended stored procedures are used to extend the functionality of the server in ways that you cannot achieve by using Transact-SQL code alone. Examples of system extended stored procedures are sys.xp_dirtree, sys.xp_cmdshell, and sys.sp_trace_create. (Note how the last example here has an sp_ prefix).

User Extended Stored Procedures

Although it is still possible to create user-defined extended stored procedures and attach them to SQL Server, the ability to do so is now deprecated. Extended stored procedures run directly within the memory space of SQL Server. This is not a safe place for users to be executing code. User-defined extended stored procedures are well known to the SQL Server product support group as a source of problems that are difficult to resolve.

You should now use managed-code stored procedures instead of user-defined extended stored procedures. The use of managed code to create stored procedures will be described in Module 12, Implementing Managed Code in SQL Server.

System Procedures

Friday, May 26, 2017 7:53 PM

SystemSPs.sql

SystemSPs.sql - (lo...nd (LABS\dellp (54))*

```sql
USE Northwind
GO

EXEC sp_tables ;
GO
EXEC sp_columns 'Orders';
GO
EXEC sp_help 'Customers'
GO
EXEC sp_depends @objname = 'Orders' ;
GO
```

127 %

Results | Messages | Execution plan

	Name	Owner	Type	Created_datetime
1	Customers	dbo	user table	2017-04-11 17:52:01.913

	Column_name	Type	Computed	Length	Prec	Scale	Nullable	TrimTrailingBlanks	FixedLenNullInSource	Collation
1	CustomerID	nchar	no	10			no	(n/a)	(n/a)	SQL_Latin1_General_CP1_CI_AS
2	CompanyName	nvarchar	no	80			no	(n/a)	(n/a)	SQL_Latin1_General_CP1_CI_AS
3	ContactName	nvarchar	no	60			yes	(n/a)	(n/a)	SQL_Latin1_General_CP1_CI_AS
4	ContactTitle	nvarchar	no	60			yes	(n/a)	(n/a)	SQL_Latin1_General_CP1_CI_AS
5	Address	nvarchar	no	120			yes	(n/a)	(n/a)	SQL_Latin1_General_CP1_CI_AS
6	City	nvarchar	no	30			yes	(n/a)	(n/a)	SQL_Latin1_General_CP1_CI_AS
7	Region	nvarchar	no	30			yes	(n/a)	(n/a)	SQL_Latin1_General_CP1_CI_AS
8	PostalCode	nvarchar	no	20			yes	(n/a)	(n/a)	SQL_Latin1_General_CP1_CI_AS

	Identity	Seed	Increment	Not For Replication
1	No identity column defined.	NULL	NULL	NULL

	RowGuidCol
1	No rowguidcol column defined.

	Data_located_on_filegroup
1	PRIMARY

	index_name	index_description	index_keys
1	City	nonclustered located on PRIMARY	City
2	CompanyName	nonclustered located on PRIMARY	CompanyName
3	PK_Customers	clustered, unique, primary key located on PRIMARY	CustomerID
4	PostalCode	nonclustered located on PRIMARY	PostalCode
5	Region	nonclustered located on PRIMARY	Region

	constraint_type	constraint_name	delete_action	update_action	status_enabled	status_for_replication	constraint_keys
1	PRIMARY KEY (clustered)	PK_Customers	(n/a)	(n/a)	(n/a)	(n/a)	CustomerID

	Table is referenced by foreign key
1	Northwind.dbo.CustomerCustomerDemo: FK_CustomerC...

Query executed successfully. | (local)\sqlexpress01 (13.0 ... | LABS\dellp (54) | Northwind | 00:00:05 | 23 rows

Create Schema Solution

Friday, May 26, 2017 6:41 PM

- [ ] Create Database
- [ ] Create Schema
- [ ] Create Schema Table
- [ ] Create Schema Procedure
- [ ] Save as ProcedureBySchema.sql

ProceduresBySchema...(LABS\dellp (54))*

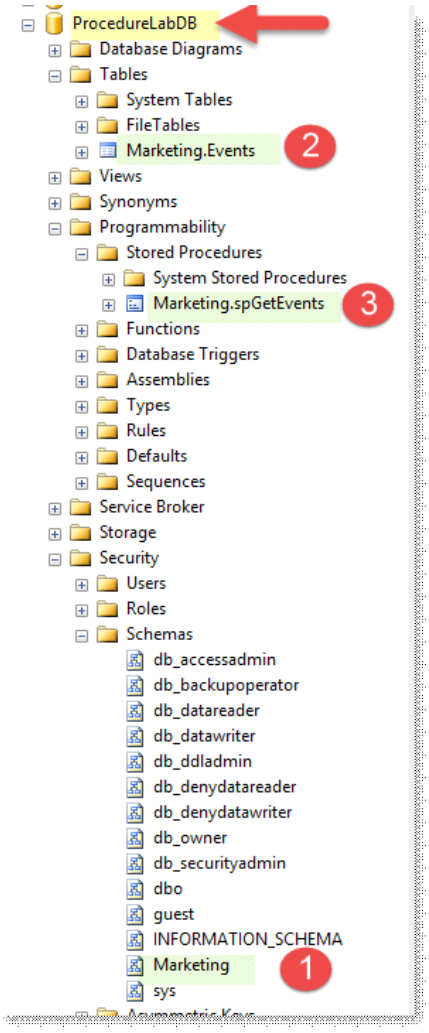
```sql
Create DataBase ProcedureLabDB;
GO

Use ProcedureLabDB;
GO

Create Schema Marketing;
Go

Create Table Marketing.Events(
    EventID int Identity(1,1) Primary Key,
    Name VarChar(25),
    Description VarChar(MAX),
    StartDate DateTime,
    EndDate DateTime
);
Go

Create Procedure Marketing.spGetEvents
As
Begin
    Select * From Marketing.Events
    Order By StartDate Desc
End;
GO

Exec Marketing.spGetEvents;

--Drop Table Marketing.Events;
--Drop Procedure Marketing.spGetEvents;
--Drop Database ProcedureLabDB;
```

127 %

Results | Messages

EventID	Name	Description	StartDate	EndDate

Query executed successfully. (local)\sqlexpress01 (13.0 ... LABS\dellp (54) ProcedureLabDB 00:00:00 0 rows



ALTER Schema Procedure

Friday, May 26, 2017 7:31 PM

- [ ] Alter Procedure
- [ ] Save as ProcedureBySchemaAlter.sql

ProceduresBySchema...(LABS\dellp (55)) | ProcedureBySchemaA...(LABS\dellp (54))*

```sql
USE ProcedureLabDB;
GO

Alter Procedure Marketing.spGetEvents
As
Begin
    Select Eventid, Name, Description, StartDate, EndDate
    From Marketing.Events
    Where StartDate Between '20160101' And '20161231'
    Order By StartDate Desc
End;
GO

Exec Marketing.spGetEvents;
GO
```

127 %

Results | Messages

Eventid	Name	Description	StartDate	EndDate

Drop Procedure

Friday, May 26, 2017 7:31 PM

ProcedureBySchemaD...(LABS\dellp (57)) ProceduresBySchema...(LA

```
Use ProcedureLabDB;
GO

Drop Table Marketing.Events;
Drop Procedure Marketing.spGetEvents;
```

Create Input Parameters

Friday, May 26, 2017 8:51 PM

InputParamteresProcedure.sql

```sql
USE Northwind;
GO

Create Procedure dbo.spGetCustomerOrdersByDateRange
    @StartDate Date,
    @EndDate Date
AS
SELECT Customers.CustomerID, Customers.CompanyName, Orders.OrderDate,
       [Order Details].UnitPrice, [Order Details].Quantity
FROM   Customers INNER JOIN
            Orders ON Customers.CustomerID = Orders.CustomerID INNER JOIN
            [Order Details] ON Orders.OrderID = [Order Details].OrderID
WHERE Orders.OrderDate BETWEEN @StartDate AND @EndDate
ORDER BY Orders.OrderDate
GO
```

RUN it Twice and Note the Error

```
Msg 2714, Level 16, State 3, Procedure spGetCustomerOrdersByDateRange, Line 2 [Batch Start Line 2]
There is already an object named 'spGetCustomerOrdersByDateRange' in the database.
```

Execute Input Parameters

Friday, May 26, 2017 9:10 PM

InputParamteresProcedureExecute.sql

InputParametersPro...d (LABS\dellp (54)) | InputParametersPro...d (LABS\dellp (55))

```
USE Northwind;
GO

Declare @StartDate DateTime = '19970101';
Declare @EndDate DateTime = '19970131';
Exec spGetCustomerOrdersByDateRange @StartDate, @EndDate;
```

127 %

Results | Messages

	CustomerID	CompanyName	OrderDate	UnitPrice	Quantity
1	EASTC	Eastern Connection	1997-01-01 00:00:00.000	99.00	21
2	EASTC	Eastern Connection	1997-01-01 00:00:00.000	14.40	35
3	EASTC	Eastern Connection	1997-01-01 00:00:00.000	16.00	30
4	RATTC	Rattlesnake Canyon Grocery	1997-01-01 00:00:00.000	20.70	18
5	RATTC	Rattlesnake Canyon Grocery	1997-01-01 00:00:00.000	30.40	70
6	RATTC	Rattlesnake Canyon Grocery	1997-01-01 00:00:00.000	16.80	20
7	RATTC	Rattlesnake Canyon Grocery	1997-01-01 00:00:00.000	17.20	60
8	ERNSH	Ernst Handel	1997-01-02 00:00:00.000	7.20	60
9	ERNSH	Ernst Handel	1997-01-02 00:00:00.000	35.10	65
10	ERNSH	Ernst Handel	1997-01-03 00:00:00.000	13.90	21
11	ERNSH	Ernst Handel	1997-01-03 00:00:00.000	10.20	70
12	MAGAA	Magazzini Alimentari Riuniti	1997-01-03 00:00:00.000	24.90	30
13	MAGAA	Magazzini Alimentari Riuniti	1997-01-03 00:00:00.000	11.20	40
14	MAGAA	Magazzini Alimentari Riuniti	1997-01-03 00:00:00.000	16.00	30
15	LINOD	LINO-Delicateses	1997-01-06 00:00:00.000	8.00	50
16	QUEEN	Queen Cozinha	1997-01-07 00:00:00.000	14.40	10
17	QUEEN	Queen Cozinha	1997-01-07 00:00:00.000	8.00	30
18	QUEEN	Queen Cozinha	1997-01-07 00:00:00.000	36.40	42
19	QUEEN	Queen Cozinha	1997-01-07 00:00:00.000	15.20	5
20	QUEEN	Queen Cozinha	1997-01-07 00:00:00.000	14.70	2
21	OTTIK	Ottilies Käseladen	1997-01-07 00:00:00.000	16.80	30
22	OTTIK	Ottilies Käseladen	1997-01-07 00:00:00.000	28.80	15
23	OTTIK	Ottilies Käseladen	1997-01-07 00:00:00.000	17.20	15
24	FOLIG	Folies gourmandes	1997-01-08 00:00:00.000	20.80	10
25	FOLIG	Folies gourmandes	1997-01-08 00:00:00.000	5.90	6
26	FOLIG	Folies gourmandes	1997-01-08 00:00:00.000	39.40	35
27	OCEAN	Océano Atlántico Ltda.	1997-01-09 00:00:00.000	18.60	12
28	OCEAN	Océano Atlántico Ltda.	1997-01-09 00:00:00.000	8.00	12
29	BOTTM	Bottom-Dollar Markets	1997-01-10 00:00:00.000	2.00	49
30	BOTTM	Bottom-Dollar Markets	1997-01-10 00:00:00.000	44.00	16

Query executed successfully. | (local)\sqlexpress01 (13.0 ... | LABS\dellp (54) | Northwind | 00:00:00 | 85 rows

Recompiling SPs

Monday, February 11, 2019 7:32 PM

- When a procedure is compiled for the first time or recompiled, the procedure's query plan is optimized for the current state of the database and its objects. If a database undergoes significant changes to its data or structure, recompiling a procedure updates and optimizes the procedure's query plan for those changes. This can improve the procedure's processing performance.
- There are times when procedure recompilation must be forced and other times when it occurs automatically. Automatic recompiling occurs whenever SQL Server is restarted. It also occurs if an underlying table referenced by the procedure has undergone physical design changes.

From <https://docs.microsoft.com/en-us/sql/relational-databases/stored-procedures/recompile-a-stored-procedure?view=sql-server-2017>

```sql
USE Northwind;
GO

IF EXISTS (SELECT * FROM sys.objects WHERE type = 'P' AND name = 'spGetCustomerOrdersByDateRange')
  DROP PROCEDURE spGetCustomerOrdersByDateRange
GO

Create Procedure dbo.spGetCustomerOrdersByDateRange
    @StartDate Date,
    @EndDate Date
With Recompile --Immediately update Query Plan
AS
Begin
    SELECT Customers.CustomerID, Customers.CompanyName, Orders.OrderDate,
           [Order Details].UnitPrice, [Order Details].Quantity
    FROM   Customers INNER JOIN
                Orders ON Customers.CustomerID = Orders.CustomerID INNER JOIN
                [Order Details] ON Orders.OrderID = [Order Details].OrderID
    WHERE Orders.OrderDate BETWEEN @StartDate AND @EndDate
    ORDER BY Orders.OrderDate
End
GO
```

If you were to **add** an index on **OrderDate** and change the Stored Proc to **Order By OrderDate**, then SQL will go with the last Query Plan; therefore Recompile is needed….

```sql
USE Northwind;
GO

Declare @StartDate DateTime = '19970101';
Declare @EndDate DateTime = '19970131';
Exec spGetCustomerOrdersByDateRange @StartDate, @EndDate With Recompile;

--With Recompile in this example does not overwrite the original Plan
```

You can **recompile** on the **Execute** statement but it only Recompiles for that Query.

Create Output Parameter

Friday, May 26, 2017 9:27 PM

OutputParamtersProcedure.sql

OutputParametersPr...(LABS\dellp (57)) | InputParametersPro...d (LABS\dellp (54)) | InputParametersPro...d (LABS\dellp (55))

```sql
USE Northwind;
GO

CREATE PROCEDURE dbo.spGetCustomerCount
    @Country nvarchar(30),
    @CustomerCount int OUTPUT
AS
SELECT @CustomerCount = count(*)
    FROM Customers
    WHERE Country = @Country;
GO
```

127 %

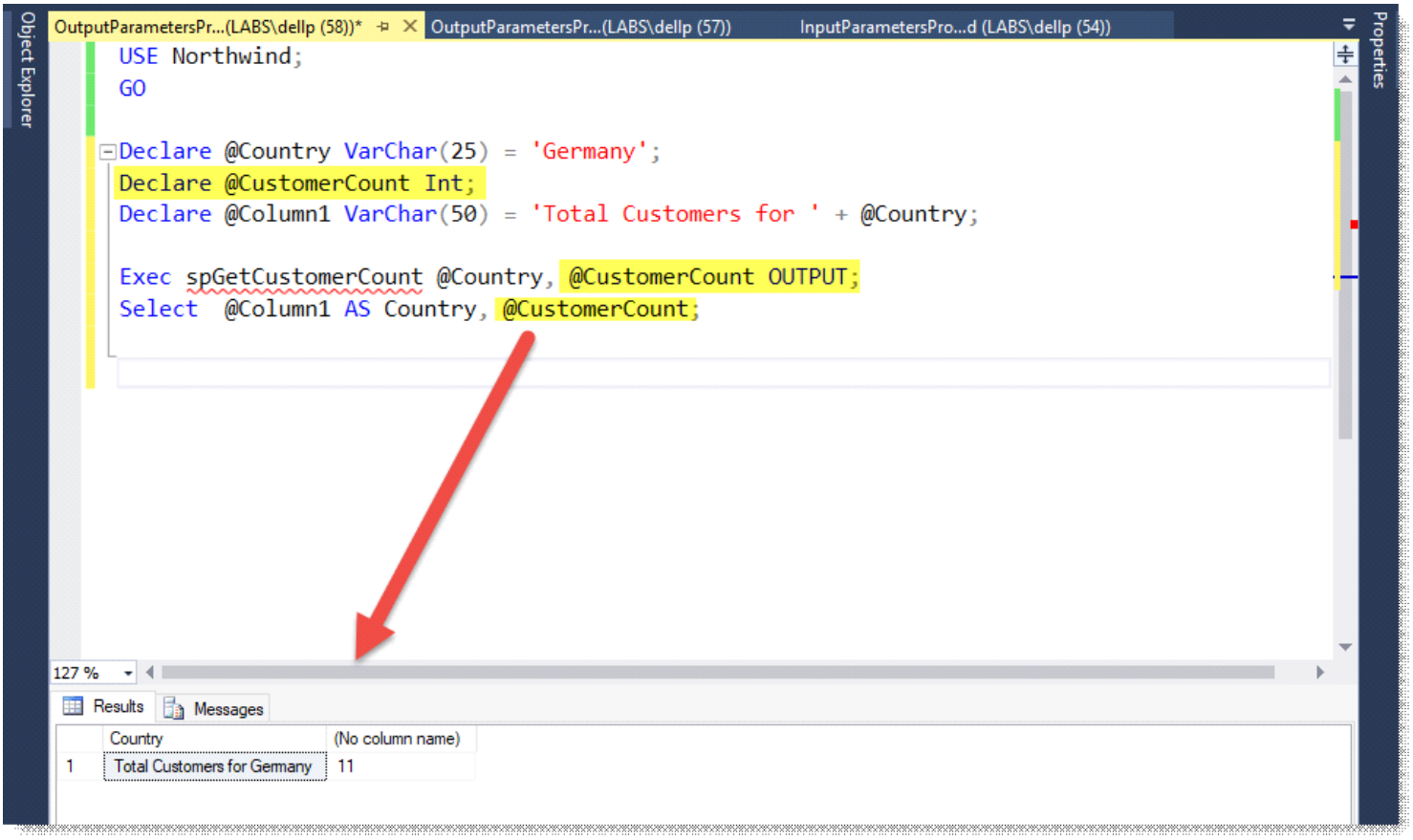
Messages

```
Command(s) completed successfully.
```

Execute Output Parameter

Friday, May 26, 2017 9:28 PM

OutputParamtersProcedure.sql



Events Loop

Tuesday, June 20, 2017 9:32 PM

EventDaysLoop.sql -...d (LABS\dellp (57))* | StoredProcedureLog...(LABS\dellp (58))

```sql
CREATE Procedure dbo.spGetEventDates
    @StartDate as Date,
    @Events Int,
    @DaysInBetween Int
AS
    CREATE TABLE #Event(EventDate DATETIME)
    DECLARE @i INT
    SET @i = 1

    WHILE @i <= @Events
    BEGIN
        INSERT INTO #Event(EventDate)
        VALUES(@StartDate)

        SET @StartDate = DATEADD(DAY,@DaysInBetween,@StartDate)
        SET @i = @i + 1
    END
    SELECT CONVERT(VarChar,EventDate,101) As [Date Ordered],
        DATEPART(QQ,EventDate) As Qtr,
        DATEPART(MM,EventDate) As Month,
        DATEPART(DD,EventDate) As [Day Of Month],
        DATEPART(YY,EventDate) As Year,
        DATEPART(DayOfYear,EventDate) As [Day Of Year],
        DATEPART(WEEKDAY,EventDate) As [Day Of Week]
    FROM #Event
    Drop Table #Event
Go
--                      Start       Events  Between
Exec spGetEventDates  '2017-01-01', 5,   50
Drop Procedure spGetEventDates
```

113 %

Results | Messages

	Date Ordered	Qtr	Month	Day Of Month	Year	Day Of Year	Day Of Week
1	01/01/2017	1	1	1	2017	1	1
2	02/20/2017	1	2	20	2017	51	2
3	04/11/2017	2	4	11	2017	101	3
4	05/31/2017	2	5	31	2017	151	4
5	07/20/2017	3	7	20	2017	201	5

Query executed successfully. | (local)\sqlexpress01 (13.0 ... | LABS\dellp (57) | Northwind | 00:00:00 | 5 rows

Stored Proc Logic

Saturday, May 27, 2017 11:39 AM

StoredProcedureLog...(LABS\dellp (58))*

```
Create Procedure spSampleData
      @Group varchar(25) = 'People',
      @TotalTables int = 3,
      @TopRows int = 5
AS
Declare @Iterator int = 1;
If (@Group = 'People')
    While (@Iterator <= @TotalTables)
        Begin
            --Print @Iterator;
            IF (@Iterator = 1)
                Select Top (@TopRows) FirstName + ' ' + LastName As Name From Employees;
            Else If(@Iterator = 2)
                Select Top (@TopRows) CompanyName From Customers;
            Else If(@Iterator = 3)
                Select Top (@TopRows) CompanyName From Suppliers;
            Else If(@Iterator = 4)
                Select Top (@TopRows) CompanyName From Shippers;
            Else
                Select 'Table #' + CONVERT(varchar(1), @Iterator) + '  is NOT - AVAILABLE...

            Set @Iterator = @Iterator + 1
        End
Else If (@Group = 'Product')
    While (@Iterator <= @TotalTables)
    Begin
        --Print @Iterator;
        IF (@Iterator = 1)
            Select Top (@TopRows) CategoryName As Name From Categories;
        Else If(@Iterator = 2)
            Select Top (@TopRows) ProductName From Products;
        Else
            Select 'Table #' + CONVERT(varchar(1), @Iterator) + '  is NOT - AVAILABLE...'

        Set @Iterator = @Iterator + 1
    End
GO
```

127 %

```
--EXEC spSampleData
EXEC spSampleData 'Product', 5, 5;

Drop Proc spSampleData;
```

Execution Content

Friday, May 26, 2017 9:57 PM

IF TIME PERMITS

Login token: A login token is valid across the instance of SQL Server. It contains the primary and secondary identities against which server-level permissions and any database-level permissions that are associated with these identities are checked. The primary identity is the login itself. The secondary identity includes permissions that are inherited from rules and groups.

User token: A user token is valid only for a specific database. It contains the primary and secondary identities against which database-level permissions are checked. The primary identity is the database user itself. The secondary identity includes permissions that are inherited from database roles. User tokens do

Object Explorer

LoginTokens.sql - (...nd (LABS\dellp (59)) | OutputParametersPr...(LABS\dellp (58)) | OutputParametersPr...(LABS\dellp (57))

```sql
USE Northwind;
GO

Select * From Sys.user_token;
Select * From sys.login_token;
```

127 %

Results | Messages

	principal_id	sid	name	type	usage
1	1	0x0105000000000005150000000675D60B6B42D3806E4042C...	dbo	WINDOWS LOGIN	GRANT OR DENY

	principal_id	sid	name	type	usage
1	259	0x0105000000000005150000000675D60B6B42D3806E4042C...	LABS\dellp	WINDOWS LOGIN	GRANT OF
2	2	0x02	public	SERVER ROLE	GRANT OF
3	3	0x03	sysadmin	SERVER ROLE	GRANT OF
4	263	0x0102000000000005200000021020000	BUILTIN\Users	WINDOWS GROUP	GRANT OF
5	0	0x0101000000000000100000000	\Everyone	WINDOWS GROUP	GRANT OF
6	0	0x0101000000000000572000000	NT AUTHORITY\Local account and member of Admini...	WINDOWS GROUP	DENY ONL
7	0	0x0105000000000005150000000675D60B6B42D3806E4042C...	LABS\HelpLibraryUpdaters	WINDOWS GROUP	GRANT OF
8	0	0x0102000000000005200000020020000	BUILTIN\Administrators	WINDOWS GROUP	DENY ONL
9	0	0x0102000000000005200000042020000	BUILTIN\Hyper-V Administrators	WINDOWS GROUP	GRANT OF
10	0	0x0102000000000005200000002F020000	BUILTIN\Performance Log Users	WINDOWS GROUP	GRANT OF
11	263	0x0102000000000005200000021020000	BUILTIN\Users	WINDOWS GROUP	GRANT OF
12	0	0x0101000000000000504000000	NT AUTHORITY\INTERACTIVE	WINDOWS GROUP	GRANT OF
13	0	0x0101000000000000201000000	\CONSOLE LOGON	WINDOWS GROUP	GRANT OF
14	0	0x010100000000000050B000000	NT AUTHORITY\Authenticated Users	WINDOWS GROUP	GRANT OF
15	0	0x010100000000000050F000000	NT AUTHORITY\This Organization	WINDOWS GROUP	GRANT OF
16	0	0x010B00000000000B600000008F88F9D7FCE30000B04900...	MicrosoftAccount\dellpayne@msn.com	WINDOWS GROUP	GRANT OF
17	0	0x0101000000000000571000000	NT AUTHORITY\Local account	WINDOWS GROUP	GRANT OF
18	0	0x0101000000000000200000000	\LOCAL	WINDOWS GROUP	GRANT OF
19	0	0x0102000000000005400000024000000	NT AUTHORITY\Cloud Account Authentication	WINDOWS GROUP	GRANT OF
20	0	0x0109000000000005200000000CC5018F0D8341BA7F65BD1...	S-1-5-32-4028125388-2803578072-1053907958-34141...	WINDOWS GROUP	GRANT OF
21	0	0x0109000000000005200000000C18FA7A33AF413AF3B3AE7...	S-1-5-32-2745667521-2937320506-1424439867-41642...	WINDOWS GROUP	GRANT OF
22	0	0x0109000000000005200000021BEA73D17E9B9F5FA05F7...	S-1-5-32-1034403361-4122601751-838272506-684212...	WINDOWS GROUP	GRANT OF

Query executed successfully. | (local)\sqlexpress01 (13.0 ... | LABS\dellp (59) | Northwind | 00:00:00 | 23 rows