STORED PROCEDURE

- ✓ Stored Procedure
  - Create Stored Procedure
  - Alter Stored Procedure
  - Execute Stored Procedure
  - Parameter OUTPUT
  - Stored Procedure Return Data SCOPE_IDENTITY()
  - @@IDENTITY
  - IDENT_CURRENT

Tujuan Mata Kuliah

- ✓ Mahasiswa mampu membuat stored procedure

Tools yang digunakan

- ✓ SqlCmd
- ✓ SQL Server Management Studio SQL Query Editor

Stored Procedure

Suatu block T-SQL yang menyimpan sekumpulan perintah yang tersimpan pada server side.

Create Stored Procedure

```
CREATE PROC [ EDURE ] nama_procedure
    [ { @parameter data_type }
        [ = default ] [ OUTPUT ]
    ] [ ,...n ]

[ WITH
    { RECOMPILE | ENCRYPTION | RECOMPILE , ENCRYPTION } ]



AS sql statement [ ...n ]
```

Contoh Stored procedure tanpa parameter:

```
CREATE PROCEDURE sp_select_vendor
AS
SET NOCOUNT ON
SELECT * FROM VENDOR
```

Contoh Stored procedure berparameter:

```
CREATE PROCEDURE sp_insert_vendor
                  @nama_vendor varchar(35),
                  @alamat      varchar(50),
                  @kota        varchar(25),
                  @telepon     varchar(15)
AS
BEGIN
      INSERT INTO VENDOR VALUES(
                        @nama_vendor,
                        @alamat,
                        @kota,
                        @telepon
                  )
END
```

Alter Stored Procedure

Contoh :

```
ALTER PROCEDURE sp_insert_vendor
                  @nama_vendor varchar(35),
                  @alamat      varchar(50),
                  @kota        varchar(25),
                  @telepon     varchar(15)
AS
BEGIN
      SET NOCOUNT ON
      BEGIN TRY
            INSERT INTO VENDOR VALUES(
                              @nama_vendor,
                              @alamat,
                              @kota,
                              @telepon
                        )
            PRINT 'OK inserted'
      END TRY
      BEGIN CATCH
         PRINT 'GAGAL inserted'
      END CATCH
END
```

Set NoCount on digunakan untuk menghilangkan message hasil output (affected row)

Execute Stored Procedure

Untuk melakukan eksekusi terhadap suatu stored procedure.

```
EXECUTE <nama_stored_procedure> <param1>, <param2>, <param…n>
```

Atau

```
EXEC <nama_stored_procedure> <param1>, <param2>, <param…n>
```

Contoh:

```
EXECUTE sp_select_vendor
```

```
EXECUTE sp_insert_vendor 'c gw','ujung berung','bandung','081321975455'
```

Stored Procedure dengan parameter OUTPUT

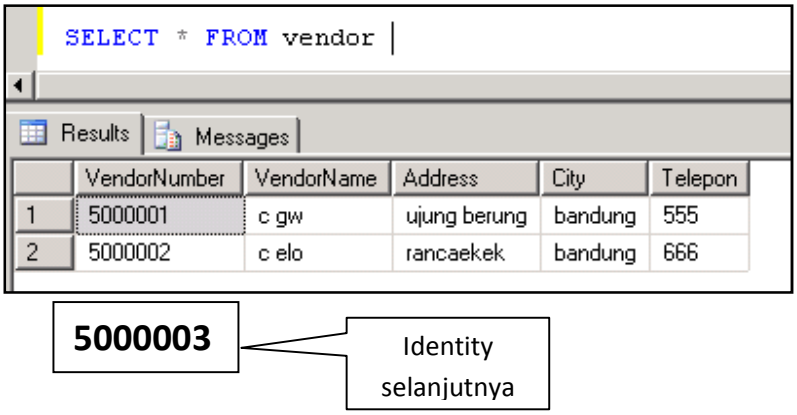
Selain parameter sebagai input pada stored procedure ini juga mengizinkan untuk menghasilkan keluaran (output) dengan menambah keyword ***output*** setelah parameter, dan eksekusinya pun menggunakan keyword ***output*** juga. Penampung output dideklarasi dengan variable.

```
CREATE PROCEDURE sp_select_vendor_by_number
                  @vendor_number int,
                  @vendor_name varchar(50) OUTPUT
AS

SET NOCOUNT ON

SELECT @vendor_name=vendorname
FROM VENDOR
WHERE vendorNumber = @vendor_number
```

```
DECLARE @nama_vendor varchar(50)
BEGIN
      EXECUTE sp_select_vendor_by_number 5000001,@nama_vendor OUTPUT
      PRINT 'Nama vendor : '+ @nama vendor
END
```

Stored Procedure Return Data

Ketika menggunakan Stored Procedure untuk menginsert data baru dimana primary key sebagai IDENTITY, kita memiliki kepentingan untuk menangkap counter/identity yang telah diinsert. Disini kita dapat menggunakan buil-in function SCOPE_IDENTITY() untuk mengambil data yang terakhir diinsert.



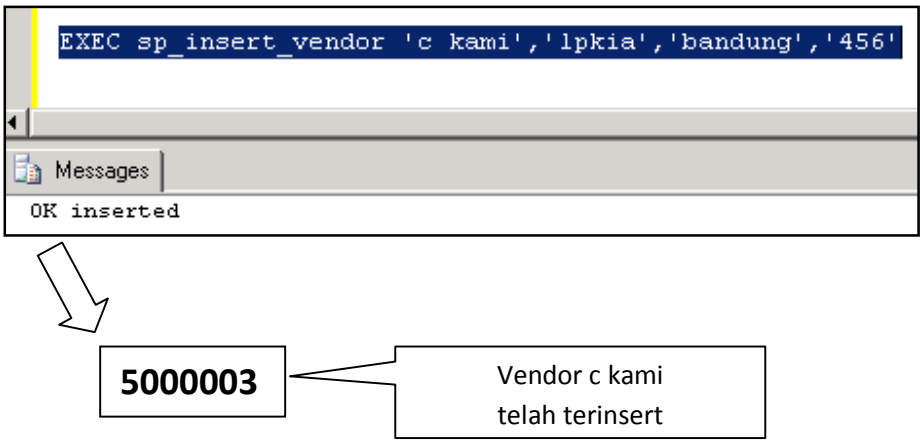
Kita akan insert record baru dengan eksekusi stored procdure dibawah

```
EXEC sp_insert_vendor 'c kami','lpkia','bandung','456'
```

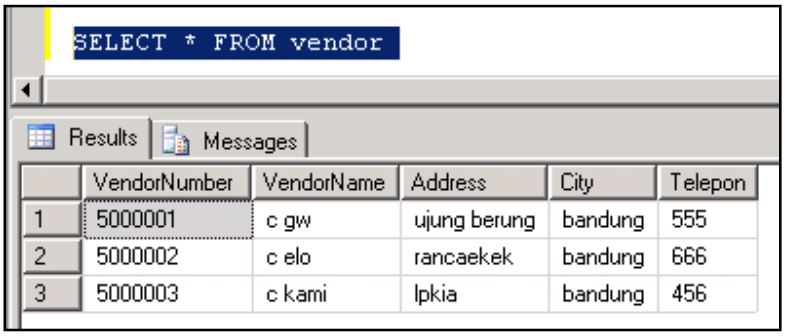
Sehingga nantinya akan menjadi

5000003 | c kami | lpkia | bandung | 456

OK Eksekusi



OK Select kembali



	VendorNumber	VendorName	Address	City	Telepon
1	5000001	c gw	ujung berung	bandung	555
2	5000002	c elo	rancaekek	bandung	666
3	5000003	c kami	lpkia	bandung	456

Biasanya untuk mencari nomor terakhir kita menggunakan MAX dari vendornunber. Akan tetapi terkadang terjadi disamping nomor yang bisa saja terjadi tidak sequence juga masalah dengan performance. Maka solusinya dengan menggunakan SCOPE_IDENTITY().

Alter stored procedure `sp_insert_vendor, menjadi`

```
ALTER PROCEDURE sp insert vendor
                  @nama_vendor varchar(35),
                  @alamat      varchar(50),
                  @kota        varchar(25),
                  @telepon     varchar(15)
AS
BEGIN
      SET NOCOUNT ON
      BEGIN TRY
            INSERT INTO VENDOR VALUES(
                              @nama_vendor,
                              @alamat,
                              @kota,
                              @telepon
                        )
            PRINT 'OK inserted'
            RETURN SCOPE_IDENTITY()
      END TRY
      BEGIN CATCH
         PRINT 'GAGAL inserted'
      END CATCH
END
```

Block eksekusi

```
DECLARE @nomor_baru int
BEGIN
  SET NOCOUNT ON
  EXECUTE @nomor_baru = sp_insert_vendor 'c kita','lpkia','bandung','456'
  PRINT 'Nomor yang baru diinsert '+ltrim(str(@nomor_baru))
END
```

Maka setelah dieksekusi

```
DECLARE @nomor_baru int
BEGIN
  SET NOCOUNT ON
  EXECUTE @nomor_baru = sp_insert_vendor 'c kita','lpkia','bandung','456'
  PRINT 'Nomor yang baru diinsert '+ltrim(str(@nomor_baru))
END
```

Messages

```
OK inserted
Nomor yang baru diinsert 5000004
```

Select kembali untuk memastikan

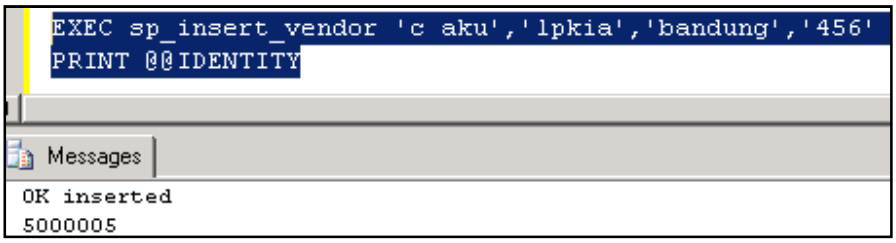
```
SELECT * FROM vendor
```

Results | Messages

	VendorNumber	VendorName	Address	City	Telepon
1	5000001	c gw	ujung berung	bandung	555
2	5000002	c elo	rancaekek	bandung	666
3	5000003	c kami	lpkia	bandung	456
4	5000004	c kita	lpkia	bandung	456

@@IDENTITY

Identity adalah global variable yang fungsinya sama seperti SCOPE_IDENTITY yaitu menangkap indentity record yang terakhir di insert dalam session yang sama.



Jika tidak identity pada current connection maka @@IDENTITY bernilai NULL, dan jika terjadi multiple insert record maka record terakhirlah yang akan mengembalikan nilai identity terakhir diinsert.

IDENT_CURRENT

Sama sperti @@IDENTITY dan SCOPE_IDENTITY hanya saja IDENT_CURRENT tidak terbatas oleh session dan scope karena IDENT_CURRENT menyebutkan secara explicit table yang memiliki identity.

```
SELECT IDENT_CURRENT('vendor')
```

LATIHAN 1

1. BUATLAH STORED PROCEDURE UNTUK MENENTUKAN NAMA MATERIAL BERDASARKAN MATERIAL NUMBER SEBAGAI INPUT PARAMETERNYA
2. BUATLAH STORED PROCEDURE UNTUK MENGINSERT DATA MATERIAL DENGAN KETENTUAN MENAMPILAN KALIMAT 'NO MATERIAL XXXXX TELAH DIINSERT'
3. BUATLAH STORED PROCEDURE UNTUK MENGINSERT TABLE MATERIAL_GROUP DENGAN PARAMETER JUMLAH DATA SEBAGAI INPUTNYA DENGAN KETENTUAN MATERIAL GROUP A01,A02,…A0…n DAN NAMA DATA DIMULAI DENGAN GROUP-A, GROUP-B, …GROUP…N. (JUMLAH RECORD YANG AKAN DIINSERT DITENTUKAN DENGAN PARAMETER JUMLAH DATA)
4. BUATLAH STORED PROCEDURE UNTUK MENAMPILKAN FUNGSI SCALAR TOTAL VALUE (QTY*PRICE) SEBAGAI OUTPUT PADA PENJUALAN DENGAN PARAMETER CUSTOMER NUMBER
5. BUATLAH STORED PROCEDURE UNTUK MENG-INSERT DATA PENJUALAN (ORDER)
6. BUATLAH STORED PROCEDURE UNTUK MENG-INSERT DATA PEMBELIAN (PURCHASE ORDER)
7. BUATLAH STORED PROCEDURE UNTUK MENG-COPY TABLE BERSERTA ISINYA DENGAN PARAMETER TABLE_SOURCE DAN TABLE_DESTINATION SEBAGAI PARAMTERNYA
8. BUATLAH STORED PROCEDURE UNTUK MENGAHAPUS DATABASE OBJECT (TABLE,VIEW,STORED PROCEDURE) DENGAN NAMA DATABASE OBJECT SEBAGAI INPUTANNYA.
9. BUATLAH STORED PROCEDURE UNTUK MENGHAPUS SEMUA TABLE YANG ADA PADA SESI-5. GUNAKAN EXCEPTION JIKA TERJADI ERROR
10. BUATLAH STORED PROCEDURE UNTUK MENGHAPUS DATABASE DENGAN NAMA DATABASE SEBAGAI INPUT PARAMETERNYA. GUNAKAN EXCEPTION JIKA TERJADI ERROR

~~EOF~~

#roots