

## PEMROGRAMAN BASIS DATA

### Trigger

**Kamarudin, M.Kom**

[kamarudin@amikom.ac.id](mailto:kamarudin@amikom.ac.id)

<http://coding4ever.net/>

<https://github.com/rudi-krsoftware/open-retail>

# Stored Procedure vs Function

- ✓ Stored procedure merupakan sekumpulan perintah T-SQL untuk melakukan tugas tertentu yang tersimpan dengan nama tertentu dan diproses sebagai sebuah kesatuan.
- ✓ Sedangkan function sama seperti store procedure dengan sedikit perbedaan yaitu *store procedure tidak mengembalikan nilai* sedang *function bisa mengembalikan nilai berupa sebuah nilai atau tabel*.
- ✓ Stored procedure dan function sama-sama disimpan sebagai objek database.

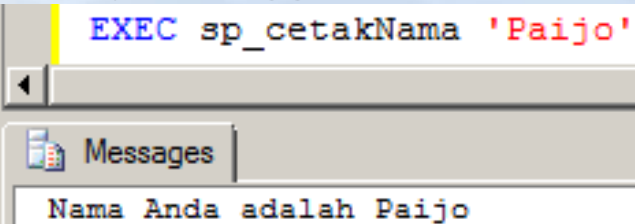
# Stored Procedure vs Function

## ✓ Contoh stored procedure

```
CREATE PROCEDURE sp_cetakNama (  
    @nama VARCHAR(30)  
)  
AS  
BEGIN  
    PRINT 'Nama Anda adalah ' + @nama  
END
```

## Cara pemanggilan

```
EXEC sp_cetakNama 'Paijo'
```



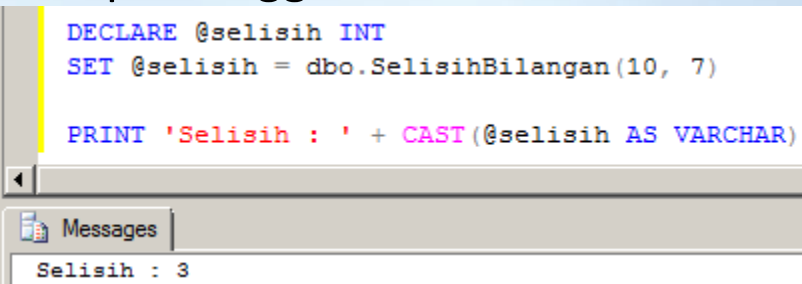
The screenshot shows the SQL Server Messages window with the text "Nama Anda adalah Paijo" displayed.

## ✓ Contoh function

```
CREATE FUNCTION SelisihBilangan (  
    @bilangan1 INT, @bilangan2 INT  
)  
RETURNS INT  
AS  
BEGIN  
    RETURN (@bilangan1 - @bilangan2)  
END
```

## Cara pemanggilan

```
DECLARE @selisih INT  
SET @selisih = dbo.SelisihBilangan(10, 7)  
PRINT 'Selisih : ' + CAST(@selisih AS VARCHAR)
```



The screenshot shows the SQL Server Messages window with the text "Selisih : 3" displayed.

# Trigger

- ✓ Merupakan store procedure yang dijalankan secara otomatis saat user melakukan modifikasi data pada tabel.
- ✓ Modifikasi data yang dilakukan pada tabel yaitu berupa perintah *INSERT*, *UPDATE*, dan *DELETE*.

# Perintah yang digunakan

```
CREATE TRIGGER NAMA_TRIGGER ON NAMA_TABEL  
FOR INSERT | UPDATE | DELETE  
AS  
    DEKLARASI VARIABEL  
BEGIN  
    PERINTAH T-SQL  
END
```

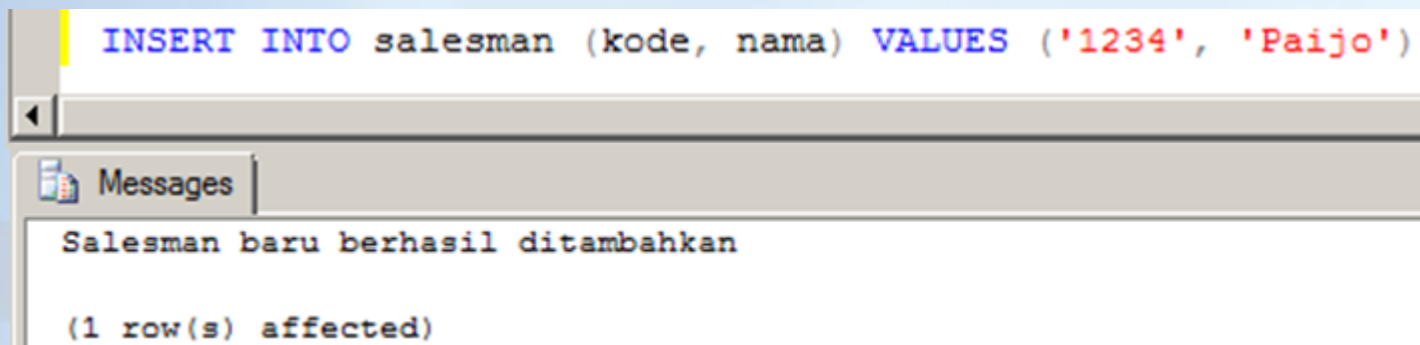
## Penjelasan :

- ✓ *NAMA\_TRIGGER*, harus mengikuti aturan identifier dan harus unik di dalam satu database.
- ✓ *TABLE*, tempat dimana trigger tersebut berada dan dieksekusi.
- ✓ *FOR INSERT | UPDATE | DELETE*, event atau pada saat apa trigger akan dieksekusi.
- ✓ *DEKLARASI VARIABEL*, optional jika dibutuhkan.
- ✓ *PERINTAH T-SQL*, kondisi atau perintah yang ada saat trigger dijalankan.

# Contoh #1 – Event Insert

```
CREATE TRIGGER tg_addSalesman ON salesman
FOR INSERT
AS
BEGIN
    PRINT 'Salesman baru berhasil ditambahkan'
END
```

Apa yang terjadi jika kita mengeksekusi perintah berikut ?



```
INSERT INTO salesman (kode, nama) VALUES ('1234', 'Paijo')
```

Messages

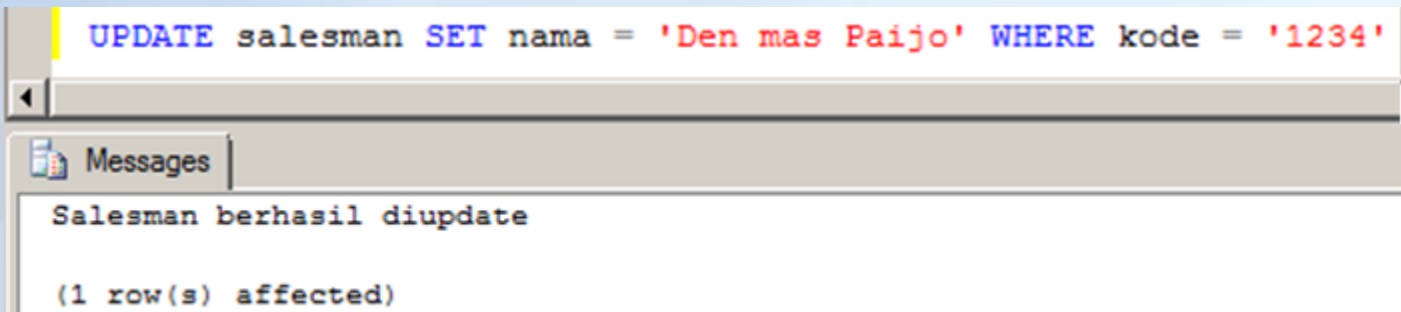
Salesman baru berhasil ditambahkan

(1 row(s) affected)

# Contoh #2 – Event Update

```
CREATE TRIGGER tg_updateSalesman ON salesman
FOR UPDATE
AS
BEGIN
    PRINT 'Salesman berhasil diupdate'
END
```

Apa yang terjadi jika kita mengeksekusi perintah berikut ?

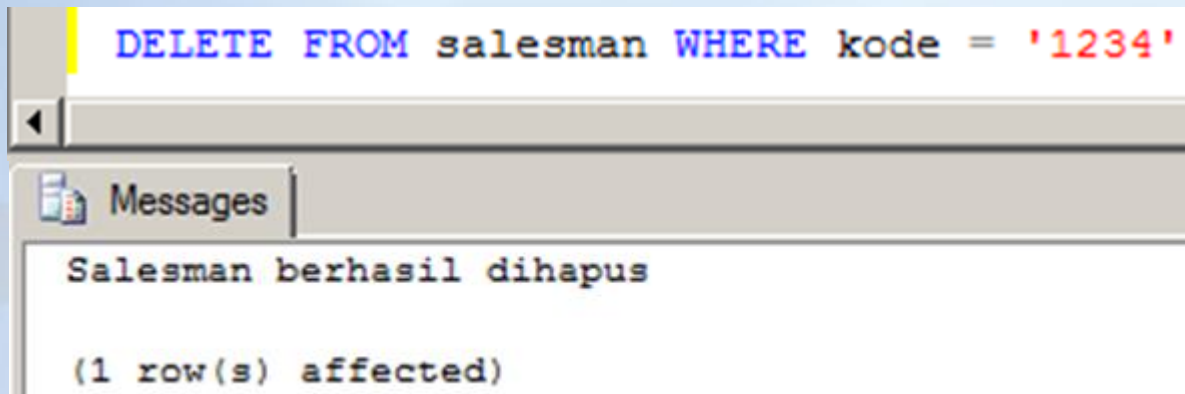


The screenshot shows a SQL execution window with a command bar at the top containing the text: `UPDATE salesman SET nama = 'Den mas Paijo' WHERE kode = '1234'`. Below the command bar is a scrollable area. At the bottom of this area, there is a section titled "Messages" which contains the output of the execution: `Salesman berhasil diupdate` followed by `(1 row(s) affected)` on a new line.

# Contoh #3 – Event Delete

```
CREATE TRIGGER tg_deleteSalesman ON salesman
FOR DELETE
AS
BEGIN
    PRINT 'Salesman berhasil dihapus'
END
```

Apa yang terjadi jika kita mengeksekusi perintah berikut ?



The screenshot shows a SQL execution window with a command bar at the top containing the text: `DELETE FROM salesman WHERE kode = '1234'`. Below the command bar is a tab labeled "Messages". The message pane displays the output: `Salesman berhasil dihapus` followed by `(1 row(s) affected)` on a new line.



# Tabel Virtual

Pada saat trigger dijalankan ada dua tabel virtual yang otomatis tercipta yaitu tabel *inserted* dan *deleted*.

Event	Tabel Virtual	
	inserted	deleted
INSERT	v	NULL
UPDATE	v	v
DELETE	NULL	v

# Tabel Virtual (Lanjutan)

## GOLONGAN \*

	Column Name	Data Type	Length	Allow Nulls
🔑	KODE	varchar	6	
	KETERANGAN	varchar	50	✓

```
INSERT INTO golongan (kode, keterangan)
VALUES ('NEW', 'HARDWARE NEW')
```

Tabel : inserted

KODE	KETERANGAN
NEW	HARDWARE NEW

Tabel : deleted

KODE	KETERANGAN
NULL	NULL

```
UPDATE golongan SET kode = 'BARU', keterangan = 'HARDWARE BARU'
WHERE kode = 'NEW'
```

Tabel : inserted

KODE	KETERANGAN
BARU	HARDWARE BARU

Tabel : deleted

KODE	KETERANGAN
NEW	HARDWARE NEW

```
DELETE FROM golongan WHERE kode = 'BARU'
```

Tabel : inserted

KODE	KETERANGAN
NULL	NULL

Tabel : deleted

KODE	KETERANGAN
BARU	HARDWARE BARU

# Contoh #1 – Event Insert

```
CREATE TRIGGER tg_add_supplier ON suppliers
FOR INSERT
AS
    DECLARE @nama VARCHAR(50)
BEGIN
    SELECT @nama = name FROM inserted

    PRINT 'Supplier dengan nama ' + @nama + ' berhasil ditambahkan'
END
```

Apa yang terjadi jika kita mengeksekusi perintah berikut ?

```
1  INSERT INTO suppliers (supplier_id, name, address)
2  VALUES ('SUP-000031', 'Pixel Cell', 'Yogyakarta')
3
```

Messages

Supplier dengan nama Pixel Cell berhasil ditambahkan

(1 row(s) affected)

# Contoh #2 – Event Update

```
ALTER TRIGGER tg_update_supplier ON suppliers
FOR UPDATE
AS
    DECLARE @namaLama VARCHAR(50)
    DECLARE @namaBaru VARCHAR(50)
BEGIN
    SELECT @namaLama = name FROM deleted
    SELECT @namaBaru = name FROM inserted

    PRINT 'Supplier dengan nama ' + @namaLama + ' berhasil diupdate menjadi ' + @namaBaru
END
```

Apa yang terjadi jika kita mengeksekusi perintah berikut ?

```
1 UPDATE suppliers SET name = 'Pixel Comp'
2 WHERE supplier_id = 'SUP-000031'
```

<

Messages

Supplier dengan nama Pixel Cell berhasil diupdate menjadi Pixel Comp

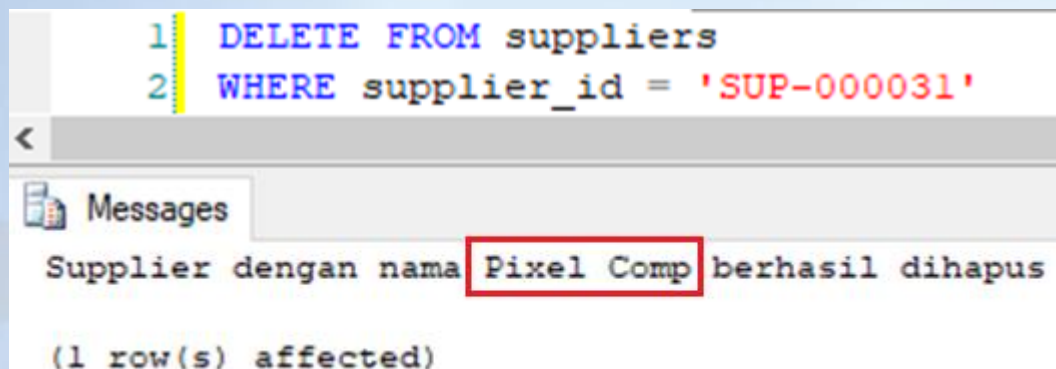
(1 row(s) affected)

# Contoh #3 – Event Delete

```
CREATE TRIGGER tg_delete_supplier ON suppliers
FOR DELETE
AS
    DECLARE @nama VARCHAR(50)
BEGIN
    SELECT @nama = name FROM deleted

    PRINT 'Supplier dengan nama ' + @nama + ' berhasil dihapus'
END
```

Apa yang terjadi jika kita mengeksekusi perintah berikut ?



The screenshot shows a SQL query window with the following text:

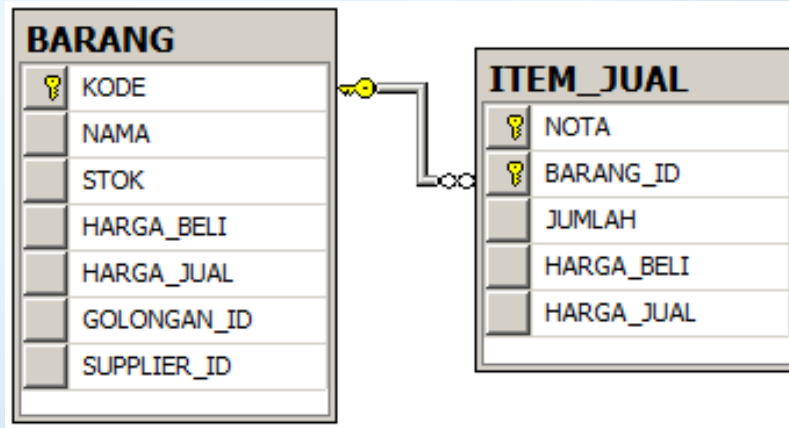
```
1 DELETE FROM suppliers
2 WHERE supplier_id = 'SUP-000031'
```

Below the query window, a 'Messages' pane displays the output of the execution:

```
Supplier dengan nama Pixel Comp berhasil dihapus
(1 row(s) affected)
```

# Contoh Kasus 1

*“Bagaimana caranya agar setiap terjadi proses penjualan barang maka stok yang ada pada tabel barang otomatis **berkurang**.”*

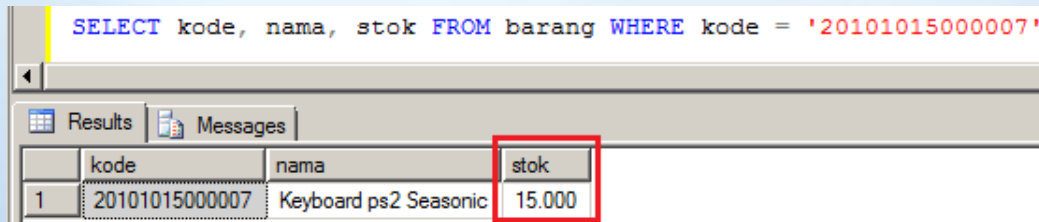


Berdasarkan kasus di atas buatlah sebuah trigger dengan nama *tg\_kurangi\_stok*.

# Testing Trigger contoh kasus 1

1. Cek data barang yang ada di tabel barang, misal data barang dengan kode '20101015000007', kemudian perhatikan nilai stoknya.

```
SELECT kode, nama, stok FROM barang WHERE kode = '20101015000007'
```



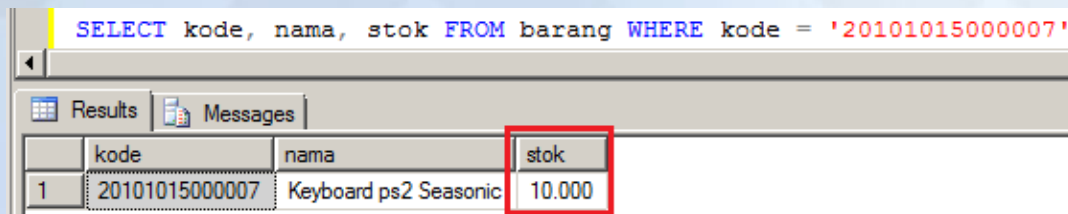
	kode	nama	stok
1	20101015000007	Keyboard ps2 Seasonic	15.000

2. Lakukan penambahan data ke tabel item\_jual untuk kode barang '20101015000007' dengan jumlah penjualan sebanyak 5 item.

```
INSERT INTO item_jual (nota, barang_id, jumlah)  
VALUES ('RB-VGA3', '20101015000007', 5)
```

3. Cek ulang data barang dengan kode '20101015000007'.

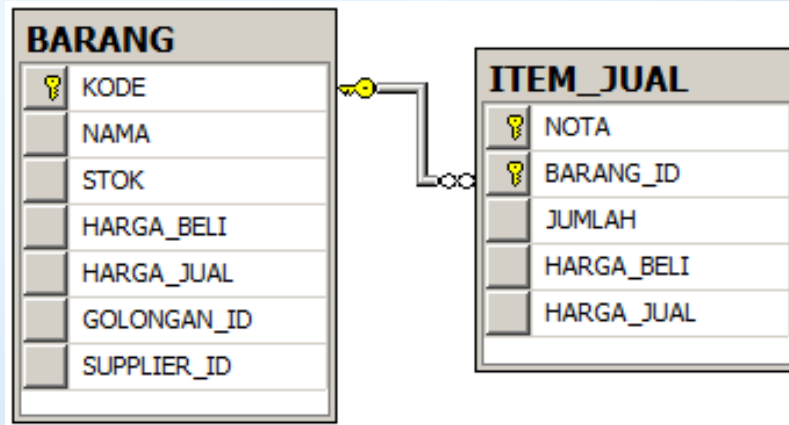
```
SELECT kode, nama, stok FROM barang WHERE kode = '20101015000007'
```



	kode	nama	stok
1	20101015000007	Keyboard ps2 Seasonic	10.000

# Contoh Kasus 2

*“Pada saat terjadi proses **pembatalan** penjualan barang maka stok yang ada pada tabel barang otomatis **bertambah**.”*



Berdasarkan kasus di atas buatlah sebuah trigger dengan nama *tg\_tambah\_stok*.



# Testing trigger contoh kasus 2

1. Cek data barang yang ada di tabel barang, misal data barang dengan kode '20101015000007', kemudian perhatikan nilai stoknya.

```
SELECT kode, nama, stok FROM barang WHERE kode = '20101015000007'
```

	kode	nama	stok
1	20101015000007	Keyboard ps2 Seasonic	10.000

2. Lakukan penghapusan data ke tabel item\_jual untuk kode barang '20101015000007' dan nota 'RB-VGA3'.

```
DELETE FROM item_jual  
WHERE nota = 'RB-VGA3' AND barang_id = '20101015000007'
```

3. Cek ulang data barang dengan kode '20101015000007'.

```
SELECT kode, nama, stok FROM barang WHERE kode = '20101015000007'
```

	kode	nama	stok
1	20101015000007	Keyboard ps2 Seasonic	15.000