

PEMROGRAMAN BASIS DATA

SQL Programming Language

Kamarudin, M.Kom

kamarudin@amikom.ac.id

<http://coding4ever.net/>

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

SQL Programming Language

- ✓ PL/SQL is a procedural language used by Oracle
- ✓ pgSQL is a procedural language used by PostgreSQL
- ✓ T-SQL is a procedural language used by SQL Server.
- ✓ MySQL is a procedural language used by MySQL

T-SQL (Transact-SQL)

- ✓ Transact-SQL adalah bahasa pemrograman disisi server yang merupakan fitur tambahan (extension) untuk Structured Query Language (SQL).
- ✓ Transact-SQL dapat dipanggil menggunakan bahasa pemrograman konvensional seperti C++, C#, Java, PHP, dll.
- ✓ Transact-SQL = SQL + Kode Program.

T-SQL Framework

- ✓ Transact-SQL dimulai dengan deklarasi variabel kemudian diikuti dengan blok program.

VARIABLE DECLARATION –
DECLARE

BLOK PROGRAM –
WITH OR WITHOUT BEGIN - END

Variabel

VARIABEL

- ✓ Variabel adalah tempat untuk menyimpan nilai atau data sementara pada aplikasi.
- ✓ Variabel isinya tidak tetap artinya dapat berubah-ubah.
- ✓ Variabel yang dideklarasikan tanpa diberikan nilai default akan diberikan nilai NULL.

SYARAT PENAMAAN VARIABEL

- ✓ Harus diawali dengan karakter @.
Contoh : @nama, @alamat, @gajiPokok
- ✓ Tidak boleh menggunakan karakter khusus, misal (, ? ; : !)
- ✓ Tidak bersifat **case sensitive**.

Variabel (Lanjutan)

DEKLARASI VARIABEL

Format penulisan :

```
DECLARE @nama_variabel TIPE DATA
```

Contoh :

```
DECLARE @nama VARCHAR(50)  
DECLARE @gajiPokok INT
```

MEMBERI NILAI VARIABEL

Format penulisan :

```
SET @nama_variabel = nilai  
SELECT @nama_variabel = nilai
```

Contoh :

```
SET @nama = 'Paijo'  
SELECT @alamat = 'Yogyakarta'
```

```
SELECT @gajiPokok = gaji_pokok FROM karyawan  
WHERE nama = @nama
```

Variabel (Lanjutan)

SQLQuery5.sql - (local)\...\roedhi (55))*

```
1  -- deklarasi variabel
2  declare @tha char(9)
3  declare @smt int
4  declare @npm char(10)
5  declare @kodeMK varchar(5)
6  declare @nilai char(1)
7
8  -- pengesetan nilai variabel
9  set @tha = '2017/2018'
10 set @smt = 1
11 set @npm = '14.11.8129'
12 set @kodeMK = 'ST078'
13
14 select @nilai = nilai from krs
15 where thn_ajaran = @tha and semester = @smt and npm = @npm and kode = @kodeMK
16
17 if @nilai is null
18     print 'belum ada nilai'
19 else
20     begin
21         print 'npm: ' + @npm
22         print 'nilai: ' + @nilai
23     end
```

krs	
id_krs	INT
npm	CHAR(10)
thn_ajaran	CHAR(9)
semester	INT
kode	VARCHAR(5)
nilai	CHAR(1)
Indexes	

Blok program

Messages

```
npm: 14.11.8129
nilai: C
```

Variabel Global

- ✓ Variable Global adalah variable yang disiapkan oleh SQL-Server untuk memberikan informasi kepada Client.
- ✓ Variable global bersifat read-only.
- ✓ Nama variable global diawali dengan @@.
- ✓ Contoh :

@@ERROR, @@CONNECTIONS, @@VERSION, @@IDENTITY

```
PRINT 'Koneksi Aktif : ' + CAST(@@CONNECTIONS AS VARCHAR)
PRINT 'Maksimal Koneksi : ' + CAST(@@MAX_CONNECTIONS AS VARCHAR)
PRINT @@VERSION
```

```
Koneksi Aktif : 18
Maksimal Koneksi : 32767
Microsoft SQL Server 2000 - 8.00.194 (Intel X86)
Aug 6 2000 00:57:48
Copyright (c) 1988-2000 Microsoft Corporation
Personal Edition on Windows NT 6.1 (Build 7600: )
```


Variabel Global (Lanjutan)

Table of Contents

- @@CONNECTIONS
- @@MAX_CONNECTIONS
- @@CPU_BUSY
- @@ERROR
- @@IDENTITY
- @@IDLE
- @@IO_BUSY
- @@LANGID
- @@LANGUAGE
- @@MAXCHARLEN
- @@PACK_RECEIVED
- @@PACK_SENT
- @@PACKET_ERRORS
- @@ROWCOUNT
- @@SERVERNAME
- @@SPID
- @@TEXTSIZE
- @@TIMETICKS
- @@TOTAL_ERRORS
- @@TOTAL_READ / @@TOTAL_WRITE
- @@TRANCOUNT
- @@VERSION

Referensi:

- <https://code.msdn.microsoft.com/Global-Variables-in-SQL-749688ef>

Print

Print adalah fungsi yang digunakan untuk mencetak text dan nilai variabel ke console.

Contoh :

SQLQuery6.sql - (local)\...\roedhi (52))*

```
1  -- deklarasi variabel
2  declare @nama varchar(30)
3
4  -- inisialisasi nilai variabel
5  set @nama = 'Paijo'
6
7  print @nama
8  print 'Nama: ' + @nama
```

<



Messages

Literal string

Paijo

Nama: Paijo

Print (Lanjutan)

Contoh :

SQLQuery7.sql - (local)\...\roedhi (56))*

```
1  -- deklarasi variabel
2  declare @nama varchar(30)
3  declare @gajiPokok int
4
5  -- inisialisasi nilai variabel
6  set @nama = 'Paijo'
7  set @gajiPokok = 1000
8
9  print 'Nama: ' + @nama + ', gaji = ' + cast(@gajiPokok as varchar)
```

Messages

Literal string

Literal string

Nama: Paijo, gaji = 1000

Flow Control

- Runtunan
- Percabangan (IF dan CASE)
- Perulangan (WHILE)

Runtunan

- ✓ Setiap perintah akan dikerjakan satu per satu
- ✓ Akhir dari perintah terakhir merupakan akhir dari program

Contoh :

```
1 declare @namaDepan varchar(50)
2 declare @namaBelakang varchar(50)
3
4 set @namaDepan = 'Tony'
5 set @namaBelakang = 'Hidayat'
6
7 print @namaDepan + ' ' + @namaBelakang
```

<

 Messages

Tony Hidayat

Percabangan

Struktur ini memungkinkan sebuah perintah dieksekusi hanya jika suatu kondisi terpenuhi atau tidak.

T-SQL mengenal dua jenis percabangan yaitu IF dan CASE

Contoh :

```
1 declare @nilaiAngka int
2 declare @nilaiHuruf char(1)
3
4 set @nilaiAngka = 3
5
6 if @nilaiAngka = 4
7     set @nilaiHuruf = 'A'
8 else if @nilaiAngka = 3
9     set @nilaiHuruf = 'B'
10 else if @nilaiAngka = 2
11     set @nilaiHuruf = 'C'
12 else
13     set @nilaiHuruf = 'D'
14
15 print 'nilai: ' + @nilaiHuruf
```

Messages

nilai: B

```
1 declare @nilaiAngka int
2 declare @nilaiHuruf char(1)
3
4 set @nilaiAngka = 3
5
6 set @nilaiHuruf = case @nilaiAngka
7     when 4 then 'A'
8     when 3 then 'B'
9     when 2 then 'C'
10    else 'D'
11 end
12
13 print 'nilai: ' + @nilaiHuruf
```

Messages

nilai: B

Percabangan (Lanjutan)

Contoh :

```
SQLQuery8.sql - (local)\...roedhi (55)* SQLQuery7.sql - not connected*
1 declare @nilaiAngka int
2 declare @nilaiHuruf char(1)
3
4 set @nilaiAngka = 75
5
6 if @nilaiAngka >= 70
7     set @nilaiHuruf = 'A'
8 else if @nilaiAngka >= 60 and @nilaiAngka < 70
9     set @nilaiHuruf = 'B'
10 else if @nilaiAngka >= 50
11     set @nilaiHuruf = 'C'
12 else
13     set @nilaiHuruf = 'D'
14
15 print 'nilai: ' + @nilaiHuruf
```

Messages
nilai: A

```
1 declare @nilaiAngka int
2 declare @nilaiHuruf char(1)
3
4 set @nilaiAngka = 75
5
6 set @nilaiHuruf = case
7     when @nilaiAngka >= 70 then 'A'
8     when @nilaiAngka >= 60 and @nilaiAngka < 70 then 'B'
9     when @nilaiAngka >= 50 then 'C'
10    else 'D'
11 end
12
13 print 'nilai: ' + @nilaiHuruf
```

Messages
nilai: A

Perulangan

Struktur ini memungkinkan suatu pernyataan untuk dieksekusi secara berulang selama kondisi yang disyaratkan masih terpenuhi. T-SQL hanya mengenal satu jenis perulangan yaitu WHILE.

Contoh:

Blok program

```
1 declare @i int
2
3 set @i = 1
4
5 while @i <= 10
6 begin
7     print 'perulangan ke: ' + cast(@i as varchar)
8     set @i = @i + 1
9 end
```

Messages

```
perulangan ke: 1
perulangan ke: 2
perulangan ke: 3
perulangan ke: 4
perulangan ke: 5
perulangan ke: 6
perulangan ke: 7
perulangan ke: 8
perulangan ke: 9
perulangan ke: 10
```


Break dan Continue

Ada 2 keyword yang sering digunakan dalam proses perulangan yaitu *break* dan *continue*.

Kedua keyword ini digunakan untuk merubah alur program.

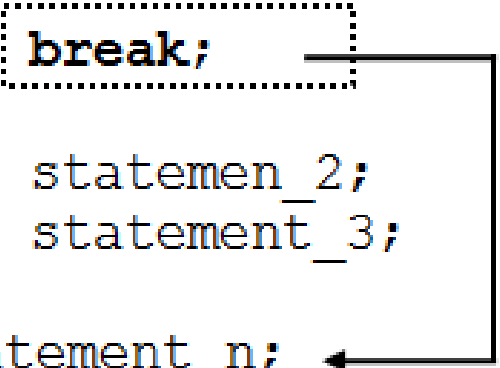
BREAK

Digunakan untuk keluar dari blok program

```
while (ekspresiPenguji)
{
    stateman_1;

    break;

    statemen_2;
    statement_3;
}
statement_n;
```



Ilustrasi break

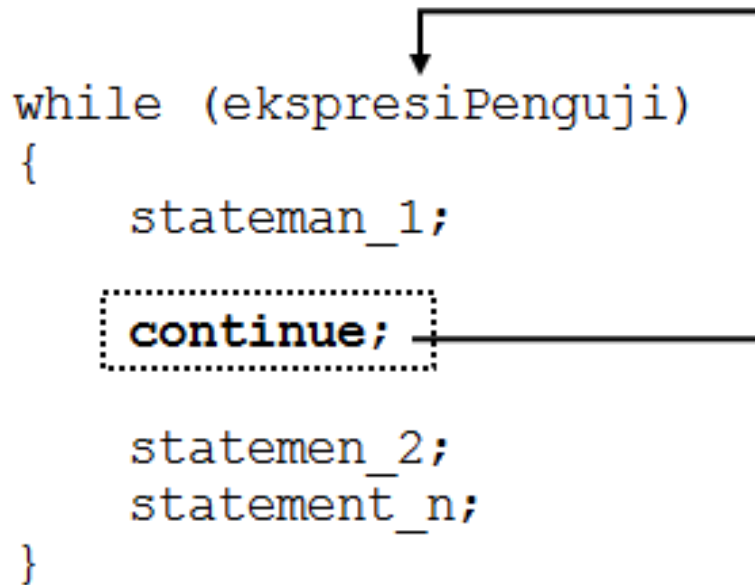
Contoh :

```
1 declare @i int
2
3 set @i = 1
4
5 while @i <= 10
6 begin
7     print 'perulangan ke: ' + cast(@i as varchar)
8     if @i = 3 break
9
10    set @i = @i + 1
11 end
```

Break dan Continue (Lanjutan)

CONTINUE

Digunakan untuk kembali ke awal perulangan sebelum semua perintah yang berada di dalam blok perulangan dikerjakan.



Ilustrasi continue

Contoh :

```
1 declare @i int
2
3 set @i = 0
4
5 while @i < 5
6 begin
7     set @i = @i + 1
8
9     if @i = 3 continue
10    print 'perulangan ke: ' + cast(@i as varchar)
11 end
```

Cursor

Cursor adalah objek database yang digunakan untuk mengambil data dari hasil SELECT, sehingga bisa diolah secara baris per baris. Objek ini mirip dengan objek DataReader di C#.

Deklarasi cursor

```
DECLARE nama_cursor CURSOR FOR  
    pernyataan SELECT
```

Contoh

```
DECLARE cursor_barang CURSOR FOR  
    SELECT kode, nama, harga_jual FROM barang
```

Untuk mengaktifkan cursor yang sudah dibuat gunakan perintah *OPEN nama_cursor*.

```
OPEN cursor_barang
```

Cursor (lanjutan)

Langkah berikutnya adalah berpindah ke record pertama dari object cursor dengan menggunakan perintah **FETCH NEXT**.

```
FETCH NEXT FROM nama_cursor INTO @variable_name [ ,...n ]
```

Contoh :

```
FETCH NEXT FROM cursor_barang  
INTO @kode_barang, @nama, @harga_jual
```

Terakhir tutup cursor dengan memanggil perintah *CLOSE* dan *DEALLOCATE*.

Contoh :

```
CLOSE cursor_barang  
DEALLOCATE cursor_barang
```

Contoh #1

```
-- deklarsi variabel
DECLARE @kode_barang VARCHAR(25)
DECLARE @nama VARCHAR(50)
DECLARE @harga_jual NUMERIC(10, 2)

-- deklarasi cursor
DECLARE cursor_barang CURSOR FOR
    SELECT kode, nama, harga_jual FROM barang

-- buka cursor
OPEN cursor_barang

-- pindah ke record pertama dari objek cursor
FETCH NEXT FROM cursor_barang INTO @kode_barang, @nama, @harga_jual

WHILE (@@FETCH_STATUS = 0)
BEGIN
    PRINT @kode_barang + ', ' + @nama

    -- pindah ke record berikutnya
    FETCH NEXT FROM cursor_barang INTO @kode_barang, @nama, @harga_jual
END

-- tutup cursor
CLOSE cursor_barang
DEALLOCATE cursor_barang
```

Contoh #2

```
-- declarsi variabel
DECLARE @kode_barang VARCHAR(25)
DECLARE @nama VARCHAR(50)
DECLARE @harga_jual NUMERIC(10, 2)

-- deklarasi cursor
DECLARE cursor_barang CURSOR FOR
    SELECT kode, nama, harga_jual FROM barang

-- buka cursor
OPEN cursor_barang

-- pindah ke record pertama dari objek cursor
FETCH NEXT FROM cursor_barang INTO @kode_barang, @nama, @harga_jual

WHILE (@@FETCH_STATUS = 0)
BEGIN
    IF (@harga_jual > 800000)
        PRINT @kode_barang + ', ' + @nama

    -- pindah ke record berikutnya
    FETCH NEXT FROM cursor_barang INTO @kode_barang, @nama, @harga_jual
END

-- tutup cursor
CLOSE cursor_barang
DEALLOCATE cursor_barang
```