T-SQL

- ✓ T-SQL ?
  - Structure penulisan
  - Deklarasi Variable
  - Select Statement
  - Print Statement
  - Global Variable
  - If statement
  - While statement
  - Continue and break
  - Return statement
  - Case statement
- ✓ Function
  - String function
  - Matematik function
  - Date Function
  - System function
- ✓ Konversi
  - Cast
  - Convert
- ✓ Cursor

Tujuan Mata Kuliah

- ✓ Mahasiswa memahami perintah-perintah dasar T-SQL
- ✓ Mahasiswa memahami penggunaan fungsi yang terdapat pada SQL SERVER 2005

Tools yang digunakan

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

Transact-SQL

Transact-SQL (T-SQL) adalah bahasa yang dikembangkan pada SQL Server. T-SQL mengembangkan kemampuan SQL sehingga dapat melengkapi SQL dengan intruksi logic. Hasili proses dapat diolah lebih lanjut dengan menggunakan control logic pemrograman procedural seperti fungsi , procedure, looping, case, if else dan lainnya.

Structure penulisan

```
DECLARE var1 <datatype>,
        var2 <datatype>,
        var...n <datatype>
BEGIN

    sql_statement | statement_block

END
```

Deklarasi Variable

Nama variable dimulai dengan karakter '@'

```
DECLARE @var1 <datatype>,
        @var2 <datatype>,
        @var...n <datatype>
```

```
DECLARE @nrp varchar(10)
DECLARE @nama varchar(6)
--atau
DECLARE @nrp varchar(10),
        @nama varchar(6)
```

Assign Variable

Variable diassign dengan 2 cara

- Dengan SELECT statement
  `SELECT @local_variable = expression/Sql Statement`
- Dengan SET
  `SET @local_variable = expression`

```
DECLARE @nrp varchar(7)

SET @nrp = '3302019'
--atau
SELECT @nrp = '3302019'
```

PRINT Statement

Digunakan untuk menulis ke layar

```
DECLARE @nrp varchar(7)

SET @nrp = '3302019'
--atau
SELECT @nrp = '3302019'

PRINT 'NRP saya : ' + @nrp
```

Assign Variable dari SELECT Statement

Assign variable juga dapat diambil dari query dengan select statement

```
DECLARE @nama_customer varchar(35)

SELECT @nama_customer=CustomerName
FROM Customer
WHERE CustomerNumber='7000001'

Print 'Chino Moreno is ' + @nama customer + ' \m/'
```

Table Variable sebagai Table

```
Declare @mat_table table(
      MaterialNumber int PRIMARY KEY,
      MaterialDescription varchar(50)
)

BEGIN
      INSERT INTO @mat_table VALUES('1','Material 1')
      INSERT INTO @mat_table VALUES('2','Material 2')
      INSERT INTO @mat table VALUES('3','Material 3')

      SELECT * FROM @mat table
END
```

Deklarasi Table Variable

```
Declare variable table(
      <Field1> <dataType>,
      <Field2> <dataType>,
      <Field...n> <dataType>
)
```

Contoh :

```
Declare @mat_table table(
      MaterialNumber int,
      MaterialDescription varchar(50)
)

BEGIN
      INSERT INTO @mat_table
            SELECT MaterialNumber,
                        MaterialDescription
            FROM
            MATERIAL

SELECT * FROM @mat_table

END
```

Assign Variable dari Sub Query

```
Declare @nama_vendor varchar(35)

SELECT @nama_vendor =
            (SELECT vendorName
             FROM VENDOR
             WHERE vendorNumber=5000001)

PRINT @nama_vendor
```

Global Variable

Nama variable global diawali dengan @@

- @@ERROR
  Mengembalikan nilai integer yang mengindikasi 0=no error dan sebaliknya
- @@ROWCOUNT
  Mengembalikan jumlah row pada statement terakhir

```
BEGIN
  UPDATE MATERIAL SET MaterialDescription='BlackBerry'
  WHERE MaterialNumber = 300001
  PRINT @@ROWCOUNT
END
```

- @@TRANCOUNT
  Mengembalikan jumlah row yang berhasil dieksekusi pada transaksi

```
BEGIN
 BEGIN TRANSACTION
  UPDATE MATERIAL SET MaterialDescription='Android'
  WHERE MaterialNumber = 300001
  PRINT @@TRANCOUNT
 COMMIT TRANSACTION
END
```

OPERATOR

Arithmetic Operators

Digunakan untuk operasi aritmetik

Operator	Keterangan
+	Penambahan
-	Pengurangan
*	Perkalian
/	Pembagian
%	Sisa Bagi

Logical Operators

Digunakan untuk operasi-operasi yang memberikan hasil BOOLEAN (True/False)

Operator	Keterangan
AND	True jika 2 operand bernilai True
OR	True jika salah satu operand true
NOT	Kebalikan dari Boolean operator

Comparison Operators

Opertor yang menyatakan hubungan adari dua buah operand. Hasil dari operasi ini juga berupa nilai BOOLEAN

Operator	Keterangan
=	Sama dengan
>	Lebih besar
<	Lebih kecil
>=	Lebih besar atau sama dengan
<=	Lebih kecil atau sama dengan
<>	Tidak sama dengan

String Concatenation Operator

Untuk menggabungkan string SQL Server menggunakan tanda +

EXCEPTION

Ketika SQL kita dijalankan terdapat error, maka secara otomatis SQL SERVER akan menampilkan pesan error sesuai dengan error number. SQL Server dapat menangkap error dengan menggunakan :

```
BEGIN TRY
     { sql_statement | statement_block }
END TRY
BEGIN CATCH
          [ { sql_statement | statement_block } ]
END CATCH
```

Contoh :

```
BEGIN TRY
    SELECT 1/0;
END TRY
BEGIN CATCH
    PRINT 'ERROR Pembagian dengan 0'
END CATCH;
```

FUNCTION

STRING FUNCTION

Nama Fungsi	Keterangan

ASCII	Mengembalikan nilai kode ASCII.
	`ASCII ( character_expression )`
	Contoh:
	`SELECT ASCII('A')`

CHAR	Kebalikan dari ASCII, yaitu mengembalikan character dari ASCII.
	`CHAR ( integer_expression )`
	Contoh:
	`SELECT CHAR(65)`

LEN	Mengembalikan panjang dari suatu string.
	`LEN ( string_expression )`
	Contoh:
	`SELECT MaterialDescription,LEN(MaterialDescription) Panjang FROM MATERIAL`

LEFT	Mengambil part dari suatu string dengan panjang yang spesifik.
	`LEFT ( character_expression , integer_expression )`
	Contoh:
	`SELECT LEFT(CustomerName,3) FROM CUSTOMER`

RIGHT	Mengambil part dari suatu string dengan panjang yang spesifik.
	`RIGHT ( character_expression , integer_expression )`
	Contoh:
	`SELECT RIGHT(CustomerName,3) FROM CUSTOMER`

SUBSTRING	Mengambil part dari character
	`SUBSTRING ( expression ,start , length )`
	Contoh:
	`SELECT SUBSTRING('LPKIA JAYA',1,5)`

LOWER	Mengambalikan string dengan huruf besar (Kapital)
	`LOWER ( character_expression )`
	Contoh:
	`SELECT LOWER('INI HURUF BESAR')`

UPPER	Mengambalikan string dengan huruf kecil
	`UPPER ( character_expression )`
	Contoh:
	`SELECT UPPER('ini huruf kecil')`

LTRIM	Mengembalikan character dengan menghapus spasi pada character sebelumnya (kiri)
	`LTRIM ( character_expression )`
	Contoh:
	`SELECT LTRIM(' No one is secure')`

RTRIM	Mengembalikan character dengan menghapus spasi pada character setelah (kanan)
	`RTRIM ( character_expression )`
	Contoh:
	`SELECT RTRIM('No one is secure ')`

REVERSE	Megembalikan dengan character terbalik
	`REVERSE ( character_expression )`
	Contoh:
	`SELECT REVERSE('kebalik')`

REPLICATE	Mengulang string dengan spesifik number
	`REPLICATE ( string_expression ,integer_expression )`
	Contoh:
	`SELECT REPLICATE('6',3)`

SPACE	Mengulang spase dengan spesifik number
	`SPACE ( integer_expression )`
	Contoh:
	`SELECT SPACE(5)+'lima spasi'`

STR	Konversi dari numeric ke string
	`STR ( float_expression [ , length [ , decimal ] ] )`
	Contoh:
	`SELECT STR(10000)`

REPLACE	Merubah spesifik suatu string dengan string lain
	`REPLACE (string_expression1, string_expression2 , string_expression3 )`
	Contoh:
	`SELECT REPLACE('UJUNG BERUNG','BERUNG','BRONX')`

SYSTEM FUNCTION

Nama Fungsi	Keterangan

ISDATE	Memeriksa sebuah ekpresi tanggal dan Mengembalikan nilai integer 1 jika valid 0 jika tidak valid
	`ISDATE ( expression )`
	Contoh:
	`SELECT ISDATE('10/10/2012')`

ISNULL	Mengganti NULL dengan spesifik value
	`ISNULL ( check_expression , replacement_value )`
	Contoh:
	`SELECT ISNULL(CustomerName,'Unknown') FROM Customer`

ISNUMERIC	Memeriksa sebuah ekpresi numeric dan Mengembalikan nilai integer 1 jika valid 0 jika tidak valid
	`ISDNUMERIC ( expression )`
	Contoh:
	`SELECT ISNUMERIC('a1000')`

NULLIF	Mengembalikan value NULL jika kedua ekspresi sama
	`NULLIF ( expression1 , expression2 )`
	Contoh:
	`SELECT NULLIF(CustomerName,'PT. ABC') FROM Customer`

DATE FUNCTION

Nama Fungsi	Keterangan
GETDATE	Mengembalikan tanggal sekarang dari server dimana SQL terinstall Contoh: `SELECT GETDATE()`

DAY	Mengembalikan nilai integer part tanggal dari tanggal
	`DAY ( date )`
	Contoh:
	`SELECT DAY(GETDATE())`

MONTH	Mengembalikan nilai integer bulan dari tanggal
	`MONTH ( date )`
	Contoh:
	`SELECT MONTH(GETDATE())`

YEAR	Mengembalikan nilai integer tahun dari tanggal
	`YEAR ( date )`
	Contoh:
	`SELECT YEAR(GETDATE())`

DATEADD	Mengembalikan datetime baru berdasarkan penambahan interval ke specific tanggal
	`DATEADD (datepart , number, date )`
	Contoh:
	`SELECT DATEADD(day,5,'02/28/2012')`
	Dimana datepart:

Date Part	Singkatan
year	yy, yyyy
quarter	qq, q
month	mm, m
dayofyear	dy, y
day	dd, d
week	wk, ww
weekday	dw, w
hour	Hh
minute	mi, n
second	ss, s
millisecond	Ms

DATEDIFF	Mengembalikan number dari batasan tanggal dan waktu diantara dua tanggal
	`DATEDIFF ( datepart , startdate , enddate )`
	Contoh:
	`SELECT (DATEDIFF(Day,'04/21/1983',GETDATE()))/365 Taun`

DATEPART	Mengembalikan nilai integer bagian dari tanggal
	`DATEPART ( datepart , date )`
	Contoh:
	`SELECT DATEPART(month, GETDATE()) AS 'Bulan'`

DATENAME	Mengembalikan string datepart dari sebuah specific tanggal
	`DATENAME ( datepart ,date )`
	Contoh:
	`SELECT DATENAME(month, GETDATE()) AS 'Nama Bulan';`

NUMERIC FUNCTION

Nama Fungsi	Keterangan

ABS	Mengembalikan value positive dari ekspresi numeric
	Contoh:
	`ABS ( numeric_expression )`
	`SELECT ABS(-1)`

CEILING	Mengembalikan pembulatan kebawah, lebih besar atau samadengan ekpresi numeric
	`CEILING ( numeric_expression )`
	Contoh:
	`SELECT CEILING(123.45)`

FLOOR	Mengembalikan pembulatan keatas, lebih kecil atau samadengan ekpresi numeric
	`FLOOR ( numeric_expression )`
	Contoh:
	`SELECT GETDATE()`

ROUND	Pembulatan dengan panjang presisi tertentu
	`ROUND ( numeric_expression , length [ ,function ] )`
	Contoh:
	`SELECT ROUND(123.345, 0)`

SQRT	Mengembalikan nilai akar dari value float
	`SQRT ( float_expression )`
	Contoh:
	`SELECT SQRT(16)`

POWER	Mengembalikan pangkat dari value flot
	`POWER ( float_expression , y )`
	Contoh:
	`SELECT POWER(2,4)`

KONVERSI

Digunakan untuk mengkonversi dari satu tipe date ke tipe data lain

Konversi dengan CAST

```
CAST ( ekpresi AS data_type [ (panjang) ])
```

Contoh :

```
SELECT CAST(getdate() as varchar(10))
```

```
SELECT CAST('10' as int)+CAST('5' as int)
```

Konversi dengan CONVERT

```
CONVERT ( data type [ ( panjang ) ] , ekpresi [ , style ] )
```

Contoh :

```
SELECT CONVERT(varchar,getdate(),103)
```

```
SELECT CONVERT(int,'10' )+CONVERT(int,'5' )
```

Style value datetime dan small datetime

Format yy	Format yyyy	Standar	Output
-	0 or 100	Default	mon dd yyyy hh:miAM (or PM)
1	101	U.S	mm/dd/yyyy
2	102	ANSI	yy.mm.dd
3	103	Britis/French	dd/mm/yy
4	104	German	dd.mm.yy
5	105	Italian	dd-mm-yy
6	106		dd mon yy
7	107		Mon dd, yy
8	108		hh:mi:ss
-	9 or 109	Default+Millisecond	mon dd yyyy hh:mi:ss:mmmAM (or PM)
10	110	USA	mm-dd-yy
11	111	JAPAN	yy/mm/dd
12	112	ISO	yymmdd
-	13 or 113	Europe default +Millisecond	dd mon yyyy hh:mi:ss:mmm(24h)
14	114		hh:mi:ss:mmm(24h)

IF….ELSE

Seleksi Kondisi alur program

```
IF Boolean ekspresi { sql statement | statement block }
[ ELSE { sql_statement | statement_block } ]
```

```
DECLARE @bilngan int

SET @bilngan = 75

IF @bilngan >=60
  PRINT 'LULUS'
ELSE
  PRINT 'GAGAL'
```

WHILE Statement

Digunakan untuk mengeksekusi satu blok perulangan sampain kondisi menjadi false

```
WHILE Boolean ekspresi
     { sql_statement | statement_block | BREAK | CONTINUE }
```

```
DECLARE @i int

SELECT @i=1
WHILE @i<=10
BEGIN
    PRINT @I
    SET @I = @I + 1
END
```

Pada looping while dapat juga digunakan statement

- CONTINUE
  Digunakan untuk skip perulangan

```
DECLARE @i int
SELECT @i=0
WHILE @i<10
BEGIN
    SET @I = @I + 1
      IF @I=5 CONTINUE
    PRINT @I
END
```

- BREAK
  Digunakan untuk keluar dari loop

```
DECLARE @i int
SELECT @i=0
WHILE @i<10
BEGIN
    SET @I = @I + 1
      IF @I=5 RETURN
    PRINT @I
END
```

CASE Statement

If statement sama halny dengan if..else, untuk seleksi kondisi. Ada dua cara penggunaanya

CASE Pertama (Simple Case)

```
CASE ekspresi
     WHEN when_ekpresi THEN hasil_expression [ ...n ]
     [ ELSE else hasil ekspresi ]
END as Field[Alias]
```

```
DECLARE @bil int,
          @terbilang varchar(15)
SET @bil = 4
SELECT @terbilang =
CASE
      WHEN 1 THEN 'Satu'
      WHEN 2 THEN 'Dua'
      WHEN 3 THEN 'Tiga'
      WHEN 4 THEN 'Empat'
      WHEN 5 THEN 'Lima'
      ELSE 'Loba teuing'
END

PRINT @terbilang
```

CASE Kedua (Searched Case)

```
Field[Alias] =
CASE
     WHEN Boolean_expression THEN hasil_ekspresi [ ...n ]
     [ ELSE else hasil ekpresi ]
END
```

```
DECLARE @bil int,
          @terbilang varchar(15)

SET @bil = 4
SELECT @terbilang =
CASE
      WHEN @bil = 1 THEN 'Satu'
      WHEN @bil = 2 THEN 'Dua'
      WHEN @bil = 3 THEN 'Tiga'
      WHEN @bil = 4 THEN 'Empat'
      WHEN @bil = 5 THEN 'Lima'
      ELSE 'Loba teuing'
END

PRINT @terbilang
```

CURSOR

Cursor adalah sejenis variable yang digunakan menampung banyak nilai berupa baris atau record. Untuk menggunakan cursor ada beberapa tahapan : deklarasi (Declare), buka (Open), ambil data (Fetch), dan tutup (Close), Serta membersihkan memory reference(Deallocate)

```
Declare @materialNumber varchar(10),
            @materialDescription varchar(50)

DECLARE material_Cursor CURSOR FOR
SELECT materialNumber,materialDescription FROM material;

OPEN material_Cursor;
FETCH NEXT FROM material_Cursor
INTO @materialNumber,@materialDescription
WHILE @@FETCH STATUS = 0
   BEGIN
        PRINT @materialNumber
      FETCH NEXT FROM material_Cursor;
   END;

CLOSE material_Cursor;

DEALLOCATE material_Cursor;
```

@@FETCH_STATUS

Return Value	Keterangan
0	Pengambilan data berhasil
-1	Pengambilan data gagal atau melebihi resultset
-2	Data yang diambil tidak ada

LATIHAN

1. Buatlah sebuah block T-SQL yang menampilkan tulisan 'STOCK MIN, SILAHKAN DI ORDER' dari table material yang nama barangnya '`IBM series X350`' jika stok kurang dari 5.
2. Buatlah T-SQL untuk menampilkan nama-nama bulan pada sebuah nilai integer sebagai inputannya
3. Buatlah satu