



Topik

1. Menampilkan data dengan menggunakan performansi query
2. Pemrograman T-SQL
3. Error Handling

Tujuan

1. Memahami komponen dari query dengan performa yang baik
2. Memahami indexes dan statistics pada SQL Server
3. Mahasiswa memahami cara menggunakan elemen bahasa T-SQL dalam pemrograman dasar.
4. Mahasiswa memahami tentang batches dan bagaimana penanganannya dalam SQL Server.
5. Mahasiswa memahami cara mendeklarasikan & menugaskan variabel dan sinonim.
6. Mahasiswa memahami cara menggunakan blok IF dan WHILE dalam flow program.
7. Mahasiswa memahami bagaimana SQL Server menangani error yang muncul di kode T-SQL.
8. Mahasiswa memahami cara mengimplementasikan penanganan exception yang terstruktur dalam T-SQL.
9. Mahasiswa memahami cara mendapatkan informasi tentang error dari system objects.

Petunjuk Umum

1. Ikuti langkah-langkah pada bagian-bagian praktikum sesuai dengan urutan yang diberikan.
2. Anda dapat menggunakan SQL Server 2012 Standard Edition untuk mencoba praktikum pada jobsheet ini. Sesuaikan dengan kondisi komputer Anda.
3. Jawablah semua pertanyaan bertanda **[Soal-X]** yang terdapat pada langkah-langkah tertentu di setiap bagian praktikum.
4. Dalam setiap langkah pada praktikum terdapat penjelasan yang akan membantu Anda dalam menjawab pertanyaan-pertanyaan pada petunjuk nomor 3, maka baca dan kerjakanlah semua bagian praktikum dalam jobsheet ini.
5. Tulis jawaban dari soal-soal pada petunjuk nomor 3 pada sebuah laporan yang dikerjakan menggunakan aplikasi word processing (Word, OpenOffice, atau yang lain yang sejenis). Eksport sebagai file **PDF** dengan format nama sebagai berikut:
 - **BDL_Tugas13_Kelas_2DigitNomorAbsen_NamaLengkapAnda.pdf**
 - Contoh:
 - o **BDL_Tugas13_TI2Q_99_Suneo.pdf**
 - Perhatikan baik-baik format penamaannya.
 - Kumpulkan file PDF tersebut sebagai laporan praktikum kepada dosen pengampu.
 - Selain pada nama file, cantumkan juga identitas Anda pada halaman pertama laporan tersebut.

TOPIK 13.1 – QUERY PERFORMANCE

Praktikum – Bagian 1: Percobaan Viewing Query Execution Plans

Langkah	Keterangan
1	<p>Pada bagian ini Anda akan melakukan percobaan untuk membuat dan menambahkan isi tabel Sales.TempOrders</p> <p>Untuk menjalankan instruksi-instruksi dibawah ini, pastikan anda tersambung dengan database TSQL2012</p>
2	<p>Departemen IT akan membuat dan mengisi tabel sample bernama Sales.TempOrders dengan perintah berikut ini :</p> <pre>IF OBJECT_ID('Sales.TempOrders') IS NOT NULL DROP TABLE Sales.TempOrders; SELECT orderid, custid, empid, orderdate, requireddate, shippeddate, shipperid, freight, shipname, shipaddress, shipcity, shipregion, shippostalcode, shipcountry INTO Sales.TempOrders FROM Sales.Orders AS o CROSS JOIN dbo.Nums AS n WHERE n.n <= 120;</pre>
3	[Soal-1] Jalankan dan catat hasilnya!
4	[Soal-2] Buat perintah TSQL untuk mengembalikan kolom ordered, custid, dan orderdate dari table Sales.TempOrders
5	Highlight dari perintah nomor 4 dan tunjukkan perkiraan execution plan. Amati elemen-elemen yang ditampilkan.
6	Arahkan kursor ke Table Scan pada Execution Plans dan lihat properti yang ditampilkan di Yellow tooltip box. Perhatikan kata “estimated” di berbagai properti.
7	[Soal-3] Posisikan kursor mouse diatas panah antara operator SELECT dan operator Table Scan dalam execution plans. Properti mana yang akan ditampilkan ?
8	[Soal-4] Tampilkan semua properti operator SELECT dalam execution plans dengan mengklik kanan operator dan memilih properti dari context menu
9	[Soal-5] Klik tombol Include Actual Execution Plan pada SQL Editor toolbar (atau tekan ctrl+M pada keyboard) dan execute dengan query SELECT
10	Analisa actual execution plan dan jelaskan!
11	[Soal-6] Salin perintah SELECT sebelumnya dan lakukan modifikasi untuk mengambil satu baris dengan menggunakan klausa TOP. Tunjukkan estimated execution plan
12	[Soal-7] Bandingkan execution plan dengan salah satu soal sebelumnya. Operator mana yang baru?
13	Salin perintah SELECT pada no 4
14	Salin modifikasi SELECT pada no 11, letakkan setelah perintah select yang sudah di salin sebelumnya (no 13)

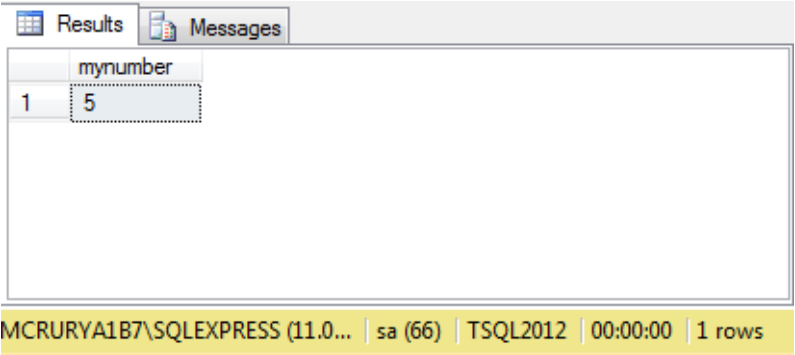
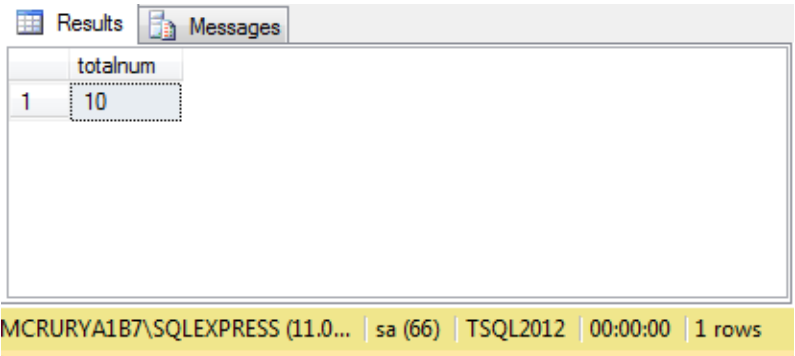
15	[Soal-8]Tunjukkan estimated execution plan kemudian tunjukkan actual execution plan
	Setelah melakukan percobaan pada bagian 1, anda telah mampu menampilkan estimated dan actual execution plan

Praktikum – Bagian 2: Percobaan Viewing Index Usage dan Penggunaan perintah SET STATISTICS

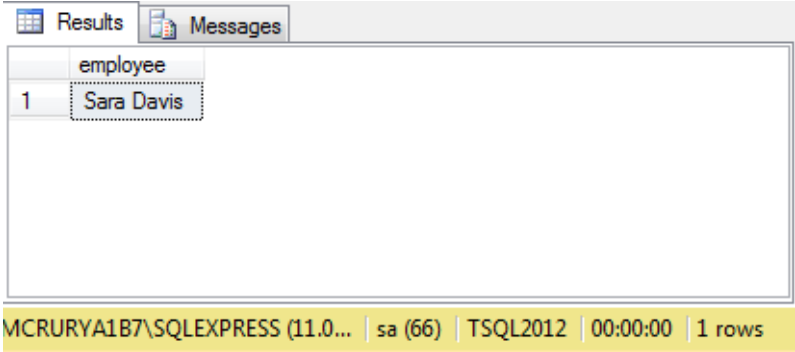
Langkah	Keterangan
1	<p>Pada modul sebelumnya, anda telah mempelajari bagaimana mengimplementasikan transaction, sekarang anda akan mempelajari tentang bagaimana cara mengaktifkan I/O statistic, menggunakan indexing dan mengimplementasikan perintah SET STATISTIC.</p> <p>Untuk menjalankan instruksi-instruksi dibawah ini, pastikan anda tersambung dengan database TSQL2012</p>
2	Eksekusi perintah dibawah ini CREATE CLUSTERED INDEX CX_Sales_TempOrders_orderdate ON Sales.TempOrders (orderdate ASC);
3	[Soal-9]Tuliskan perintah SELECT untuk mengambil kolom orderid, cutid, dan orderdate dari tabel Sales.TempOrders. Tampilkan baris dengan tahun pemesanan 2017 dan bulan pemesanan sama dengan 6.
4	Aktifkan I/O Statistic dengan mengeksekusi perintah SET STATISTIC IO
5	[Soal-10]Salin query pada [soal 9] kemudian eksekusi. Perhatikan jumlah logical reads yang ditampilkan pada message tab. Angka tersebut berdasarkan I/O Statistic.
6	[Soal-11]Salin perintah SELECT pada [soal 9] dan modifikasi dengan mengganti klausa WHERE dengan range berdasarkan kolom orderdate. Tentu saja hasilnya pasti sama. Jalankan kemudian catat hasilnya.
7	Perhatikan angka pada logical reads pad message tab. Berikan analisa anda!
8	Tampilkan query execution plan. Perhatikan operator baru diberi nama Clustered Index Seek.
10	[Soal-12]Salin Perintah Select pada no 3 kemudian tambahkan SELECT Statement yang ada pada [soal 11]. Tandai kedia perintah tersebut kemudian Jalankan!
11	[Soal-13] Perhatikan perbedaan logical reads. Meskipun hasilnya sama , perintah SELECT pada [soal 11] memindai data 25 kali lebih sedikit dibandingkan dari SELECT yang pertama. Analisa kenapa hal tersebut bisa terjadi !
12	[Soal-14] Kembalikan tabel yang sudah dibuat dan menonaktifkan I/O Statistic dengan menjalankan T-SQL code yang sudah disediakan sebelumnya.
13	Setelah melakukan percobaan bagian kedua, anda dapat memahami tentang cara mengaktifkan opsi SET STATISTIC dan perlu diingat untuk menginvestasikan waktu dalam memahami index sehingga anda dapat menulis query yang efisien

TOPIK 13.2 – T-SQL PROGRAMMING

Praktikum Bagian 3 – DEKLARASI VARIABEL & BATCH: Mendeklarasikan variabel dan mendapatkan nilai variabel

Langkah	Keterangan
1	Pastikan SSMS Anda terkoneksi ke database 'TSQL'.
2	<p>[Soal-15] Buatlah sebuah kode T-SQL dengan mendeklarasikan sebuah variable bernama <code>@num</code> yang bertipe data integer bernilai 5. Tampilkan nilai variabel tersebut dengan menggunakan alias <code>mynumber</code> lalu eksekusi.</p> <p>Hasil yang benar ditunjukkan pada tampilan berikut: xxxx</p>  <p>The screenshot shows the SQL Server Enterprise Manager interface. The 'Results' pane is active, displaying a table with one column named 'mynumber' and one row with the value '5'. The status bar at the bottom indicates 'MCRURYA1B7\SQLEXPRESS (11.0... sa (66) TSQL2012 00:00:00 1 rows'.</p>
3	<p>[Soal-16] Dari skrip T-SQL [Soal-1] di atas, tambahkan batch delimiter (GO) setelahnya, lalu buatlah skrip T-SQL baru yang mendefinisikan 2 variabel bernama <code>@num1</code> dan <code>@num2</code> yang sama-sama bertipe data integer. Set nilainya masing-masing 4 dan 6. Tulis sebuah query SELECT yang menampilkan jumlah kedua variable tersebut sebagai <code>totalnum</code>, lalu eksekusi.</p> <p>Hasil yang benar ditunjukkan pada tampilan berikut: xxxx</p>  <p>The screenshot shows the SQL Server Enterprise Manager interface. The 'Results' pane is active, displaying a table with one column named 'totalnum' and one row with the value '10'. The status bar at the bottom indicates 'MCRURYA1B7\SQLEXPRESS (11.0... sa (66) TSQL2012 00:00:00 1 rows'.</p>

Praktikum Bagian 4 – DEKLARASI VARIABEL & BATCH: Memberi nilai terhadap variabel menggunakan query SELECT

Langkah	Keterangan
1	<p>[Soal-17] Buatlah sebuah skrip T-SQL dengan mendefinisikan variabel <code>@empname</code> yang bertipe data <code>nvarchar(30)</code>. Selanjutnya, set nilai variabel tersebut sebagai hasil query <code>SELECT</code> terhadap tabel HR.Employees, yang menggabungkan kolom <i>firstname</i> dan <i>lastname</i> dengan dipisahkan spasi, dimana nilai <i>empid</i>-nya sama dengan 1.</p> <p>Terakhir, tampilkan nilai variabel <code>@empname</code> dengan menggunakan query <code>SELECT</code> dan beri nama alias sebagai <code>employee</code>. Eksekusi skrip tersebut.</p> <p>Hasil yang benar ditunjukkan pada tampilan berikut:</p> 
2	<p>[Soal-18] Apakah yang terjadi apabila query <code>SELECT</code> yang dihasilkan lebih dari 1 baris? Lakukan uji coba misalnya dengan menghilangkan filter <code>WHERE "empid = 1"</code>.</p>

Praktikum Bagian 5 – DEKLARASI VARIABEL & BATCH: Menggunakan sebuah variabel dalam klausa WHERE

Langkah	Keterangan
1	<p>[Soal-19] Salinlah skrip T-SQL dari [Soal- 3] di atas dan lakukan modifikasi dengan mendefinisikan sebuah variabel baru bernama <code>@empid</code> yang bertipe data integer bernilai 5. Lalu, gunakan variabel baru ini dalam klausa <code>WHERE</code> sebagai nilai dari kolom <i>empid</i>. Eksekusi skrip tersebut.</p> <p>Hasil yang benar ditunjukkan pada tampilan berikut:</p>

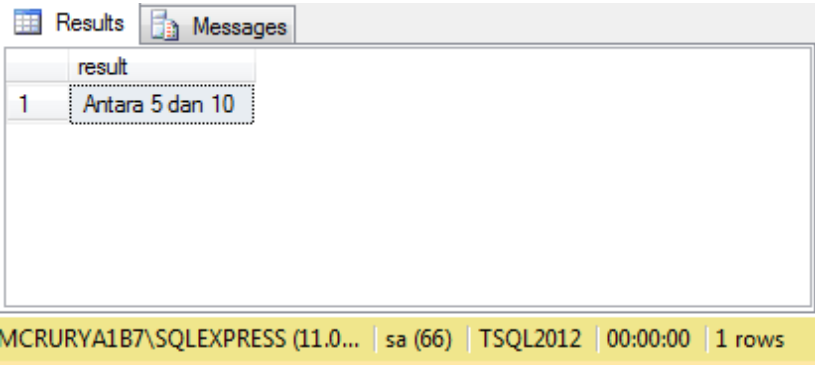
	 <p>MCRURYA1B7\SQLEXPRESS (11.0... sa (66) TSQL2012 00:00:00 1 rows</p>
2	Sebagai uji coba, ubah nilai variabel <code>@empid</code> dari 5 ke 2 lalu eksekusi kembali. Perhatikan perubahan yang terjadi.

Praktikum Bagian 6 – DEKLARASI VARIABEL & BATCH: Menambah sebuah batch delimiter

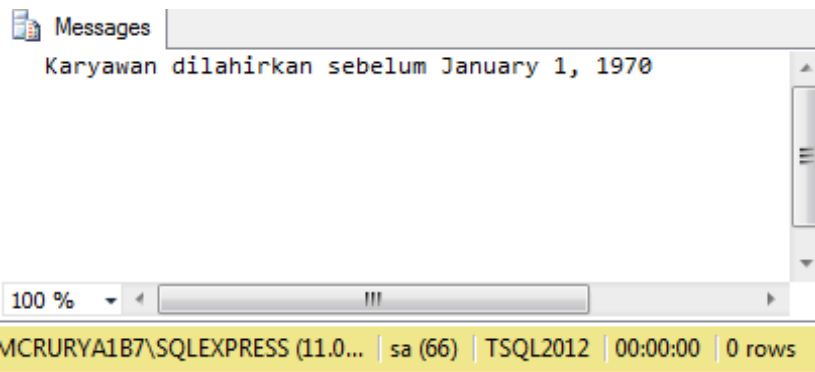
Langkah	Keterangan
1	<p>[Soal-20] Salinlah skrip T-SQL dari [Soal-5] di atas dengan menambahkan <i>batch delimiter</i> GO sebelum query SELECT seperti di bawah ini:</p> <pre>...</pre> <pre>GO</pre> <pre>SELECT @empname AS employee;</pre> <p>Setelah mengeksekusi skrip tersebut, apakah yang terjadi? Mengapa demikian?</p>

Praktikum Bagian 7 – CONTROL OF FLOW: Membuat conditional logic sederhana

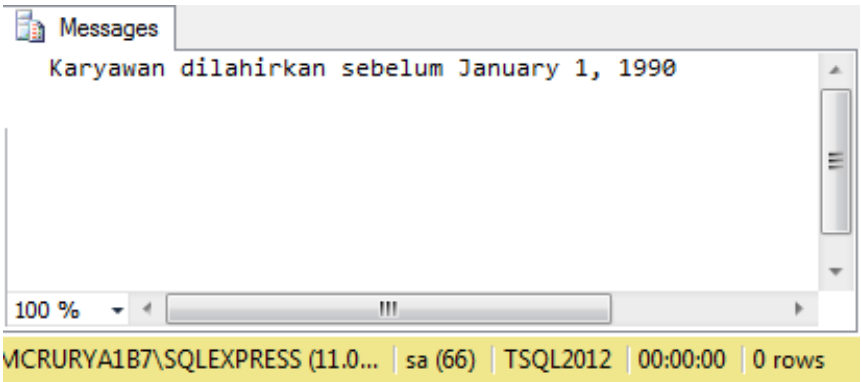
Langkah	Keterangan
1	<p>[Soal-21] Buatlah sebuah skrip T-SQL dengan mendeklarasikan variabel <code>@result</code> bertipe <code>nvarchar(20)</code> dan variabel <code>@i</code> bertipe integer bernilai 8. Tambahkan statement IF yang memenuhi <i>logic</i> di bawah ini:</p> <ul style="list-style-type: none"> Jika variabel <code>@i</code> bernilai kurang dari 5, set nilai variabel <code>@result</code> menjadi “Kurang dari 5” Jika variabel <code>@i</code> bernilai antara 5 dan 10, set nilai variabel <code>@result</code> menjadi “Antara 5 dan 10” Jika variabel <code>@i</code> bernilai lebih dari 10, , set nilai variabel <code>@result</code> menjadi “Lebih dari 10” Selain dari itu, , set nilai variabel <code>@result</code> menjadi “Unknown” <p>Di bagian akhir, tambahkan sebuah query SELECT untuk menampilkan nilai variabel <code>@result</code> dengan memberi alias <code>result</code>.</p>

	<p>Eksekusi skrip yang sudah dibuat dan bandingkan dengan hasil berikut ini:</p>  <p>MCRURYA1B7\SQLEXPRESS (11.0... sa (66) TSQL2012 00:00:00 1 rows</p>
2	<p>[Soal-22] Modifikasilah skrip T-SQL dari [Soal-7] di atas dengan mengganti statement IF menjadi ekspresi CASE dan pastikan hasilnya sama.</p>

Praktikum Bagian 8 – CONTROL OF FLOW: Mengecek tanggal lahir karyawan

Langkah	Keterangan
1	<p>[Soal-23] Ikuti langkah berikut ini untuk membuat kode T-SQL yang mengecek tanggal lahir karyawan:</p> <ul style="list-style-type: none"> • Pertama, deklarasikan 2 variabel, yakni @birthdate dan @cmpdate (keduanya bertipe data date). • Set nilai variabel @birthdate sebagai hasil dari query SELECT terhadap kolom birthdate dari tabel HR.Employees, dimana empid-nya adalah 5. • Set variabel @cmpdate berisi tanggal January 1, 1970 • Buatlah pernyataan kondisional IF dengan membandingkan nilai @birthdate dan @cmpdate. Apabila @birthdate lebih kecil dari @cmpdate, gunakan perintah PRINT untuk menampilkan pesan “Karyawan dilahirkan sebelum Januari 1, 1970”. Selain itu, tampilkan pesan “Karyawan dilahirkan pada atau setelah Januari 1, 1970”. • Eksekusi keseluruhan skrip T-SQL di atas. <p>Hasil yang benar ditunjukkan pada tampilan berikut:</p>  <p>MCRURYA1B7\SQLEXPRESS (11.0... sa (66) TSQL2012 00:00:00 0 rows</p>

Praktikum Bagian 9 – CONTROL OF FLOW: Membuat dan mengeksekusi stored procedure

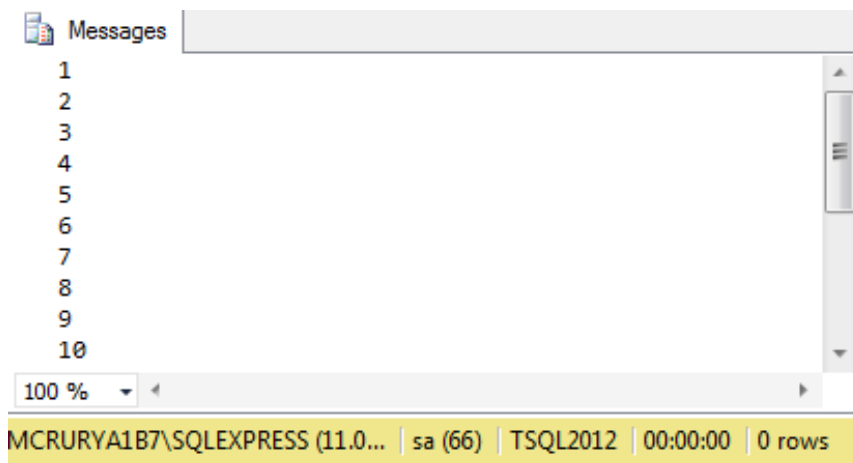
Langkah	Keterangan
1	<p>Salin & jalankan skrip T-SQL berikut ini:</p> <pre> CREATE PROCEDURE Sales.CheckPersonBirthDate @empid int, @cmpdate date AS DECLARE @birthdate date; SET @birthdate = (SELECT birthdate FROM HR.Employees WHERE empid = @empid); IF @birthdate < @cmpdate PRINT 'Karyawan dilahirkan sebelum ' + FORMAT(@cmpdate, 'MMMM d, yyyy', 'en-US'); ELSE PRINT 'Karyawan dilahirkan pada atau setelah ' + FORMAT(@cmpdate, 'MMMM d, yyyy', 'en-US');</pre>
2	<p>[Soal-24] Stored procedure bernama Sales.CheckPersonBirthDate pada Langkah 1 di atas mempunyai 2 parameter, yakni @empid (untuk menentukan ID karyawan) dan @cmpdate (untuk perbandingan tanggal).</p> <p>Lakukan perintah EXECUTE pada stored procedure tersebut dengan memasukkan parameter @empid = 3 dan @cmpdate yang di-set ke tanggal January 1, 1990.</p> <p>Hasil yang benar ditunjukkan pada tampilan berikut:</p> 

Praktikum Bagian 10 – CONTROL OF FLOW: Melakukan loop/ pengulangan menggunakan pernyataan WHILE

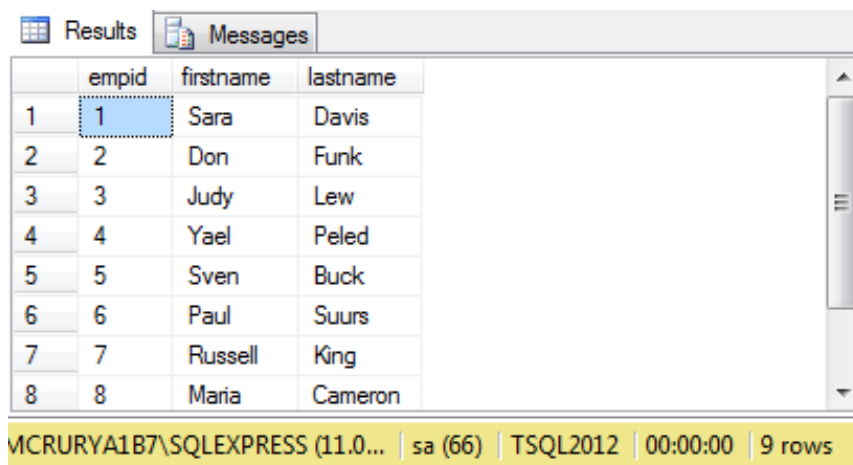
Langkah	Keterangan
1	<p>[Soal-25] Buatlah sebuah skrip T-SQL yang berisi looping/ pengulangan dengan mengikuti langkah berikut:</p> <ul style="list-style-type: none"> Pertama, deklarasikan sebuah variabel @i yang bertipe data integer bernilai 1 Lalu buatlah sebuah pengulangan dengan menggunakan pernyataan WHILE, dimana selama nilai variabel @i kurang dari 10, tampilkan/ cetak variabel @i dan tambahkan

nilai @i secara incremental dengan menambah 1 (@i+1).

Hasil yang benar ditunjukkan pada tampilan berikut:

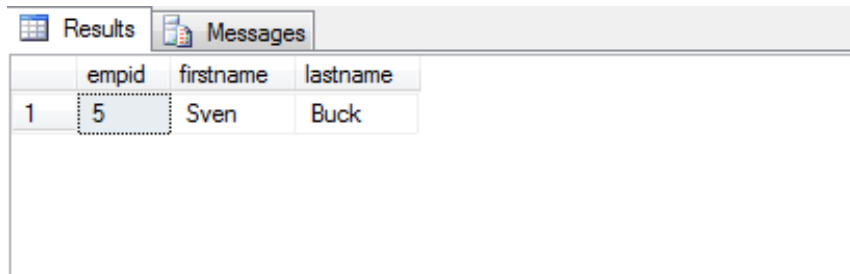


Praktikum Bagian 11 – DYNAMIC SQL: Membuat SQL dinamis tanpa parameter

Langkah	Keterangan
1	<p>[Soal-26] Buatlah skrip T-SQL dengan mendeklarasikan variabel bernama @SQLstr bertipe data <code>nvarchar(200)</code>. Lalu, isikan dengan <u>string berupa statement SELECT</u> yang mengambil kolom <code>empid</code>, <code>firstname</code>, dan <code>lastname</code> dari tabel HR.Employees.</p> <ul style="list-style-type: none">• Perhatikan bahwa yang diambil adalah string statement-nya, bukan hasil query-nya. Contoh: <code>SET @SQLstr = N'SELECT....'</code>; Bukan <code>SET @SQLstr = (SELECT....)</code>;• Lakukan perintah EXECUTE yang mengeksekusi pernyataan query yang sudah di-set dalam variabel @SQLstr. (Petunjuk: gunakan <code>sp_executesql</code>) <p>Hasil yang benar ditunjukkan pada tampilan berikut:</p>  <p>The screenshot shows the 'Results' window in SQL Server Enterprise Manager. It displays a table with 9 rows and 4 columns: <code>empid</code>, <code>firstname</code>, and <code>lastname</code>. The first row is highlighted. The status bar at the bottom indicates 'MCRURYA1B7\SQLEXPRESS (11.0... sa (66) TSQL2012 00:00:00 9 rows'.</p>
2	<p>[Soal-27] Salinlah skrip T-SQL dari [Soal-26] di atas, lakukan modifikasi dengan mengikuti langkah berikut:</p>

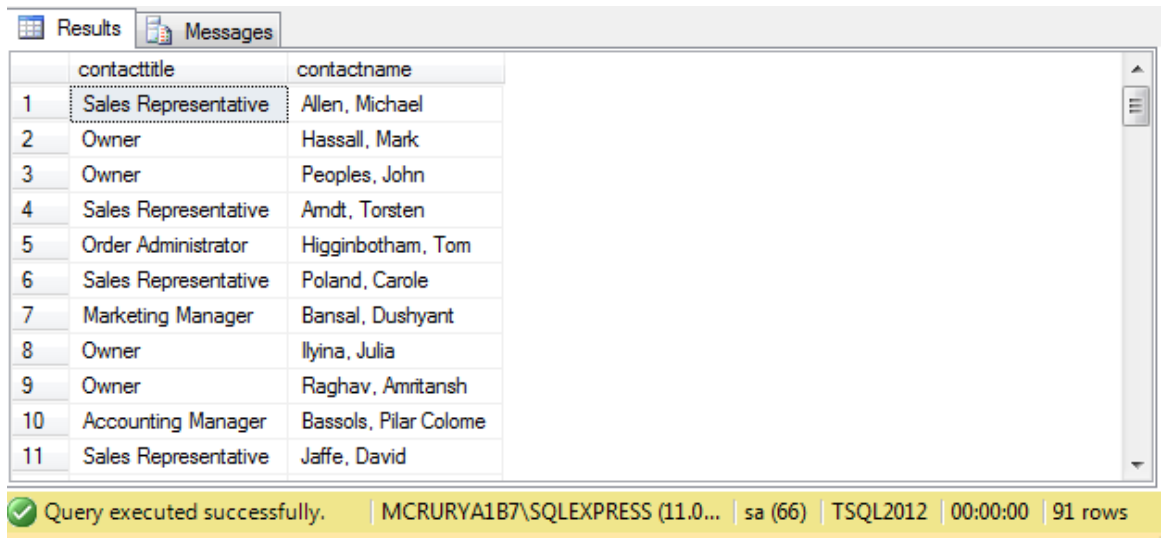
- Tambahkan filter WHERE dimana `empid = @empid` pada string pernyataan SELECT yang disimpan pada variabel `@SQLstr`
- Deklarasikan variabel baru bernama `@SQLParam` bertipe data `nvarchar(100)` untuk menyimpan string definisi dari `@empid` yang bertipe data int
- Seperti halnya pada **[Soal-13]**, lakukan perintah EXECUTE dengan menambahkan parameter `@empid = 5`

Hasil yang benar ditunjukkan pada tampilan berikut:



	empid	firstname	lastname
1	5	Sven	Buck

Praktikum Bagian 12 – SYNONYMS: Membuat dan menggunakan synonym

Langkah	Keterangan
1	<p>[Soal-28] Tulislah sebuah skrip T-SQL yang membuat SYNONYM bernama dbo.Pelanggan dari tabel Sales.Customers dalam database TSQL. Eksekusi skrip tersebut.</p> <p>Kemudian, buatlah query SELECT terhadap synonym dbo.Pelanggan yang mengambil kolom <i>contacttitle</i> dan <i>contactname</i>. Eksekusi kembali skrip T-SQL ini.</p> <p>Hasil yang benar ditunjukkan pada tampilan berikut:</p> 
2	<p>Untuk menghapus SYNONYM yang dibuat sebelumnya, jalankan skrip berikut:</p> <pre>DROP SYNONYM dbo.Pelanggan;</pre>

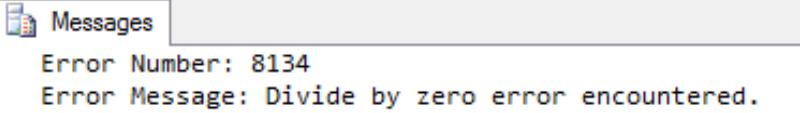
TOPIK 13.3 – ERROR HANDLING

Praktikum Bagian 13 – TRY / CATCH: Membuat blok TRY / CATCH sederhana

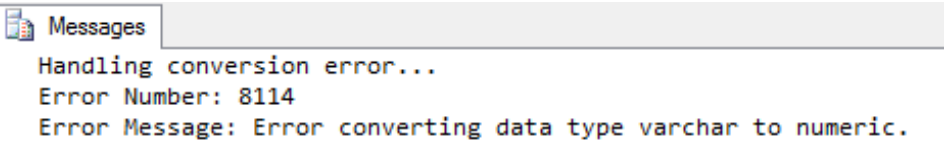
Langkah	Keterangan
1	<p>Salin dan eksekusi pernyataan SELECT berikut ini:</p> <pre>SELECT CAST(N'Ini teks loh' AS int);</pre> <p>Perhatikan error yang terjadi saat dieksekusi.</p>
2	<p>[Soal-29] Buatlah konstruksi TRY / CATCH dengan menempatkan query pada Langkah 1 di atas dalam blok TRY. Sedangkan dalam blok CATCH, isikan perintah untuk menampilkan teks “Error”. Jalankan skrip T-SQL tersebut, lalu pada tab Messages, akan menampilkan pesan:</p> <pre>(0 row(s) affected) Error</pre>

Praktikum Bagian 14 – TRY / CATCH: Menampilkan kode & pesan error

Langkah	Keterangan
1	<p>Salin dan eksekusi skrip T-SQL berikut ini dan perhatikan hasilnya:</p> <pre>DECLARE @num varchar(20) = '0'; BEGIN TRY PRINT 5. / CAST(@num AS numeric(10,4)); END TRY BEGIN CATCH END CATCH;</pre>
2	<p>[Soal-30] Pada langkah 1 di atas, jika memperhatikan nilai variabel @num, semestinya dihasilkan error “division by zero”, tetapi nyatanya tidak. Mengapa demikian?</p>
3	<p>[Soal-31] Modifikasilah skrip T-SQL pada Langkah 1 di atas dengan menambahkan 2 (dua) pernyataan PRINT pada bagian blok CATCH. Pernyataan yang pertama untuk menampilkan nomer error dengan menggunakan fungsi ERROR_NUMBER dan pernyataan kedua untuk menampilkan pesan error dengan memakai fungsi ERROR_MESSAGE.</p> <p>Untuk memperjelas, tambahkan string label “Error Number:” pada pesan pertama dan string label “Error Message:” pada pesan kedua.</p> <p>Eksekusi dan bandingkan hasilnya dengan tampilan berikut:</p>

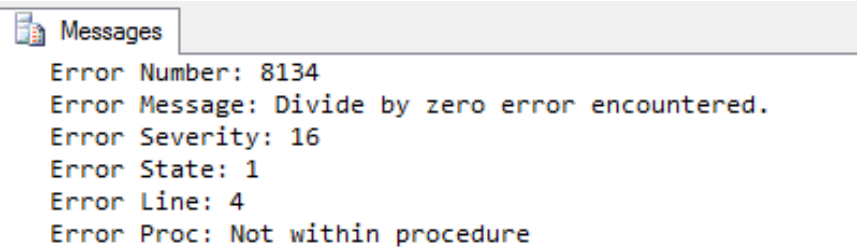
	
4	Sebagai uji coba, masih dengan skrip T-SQL yang sama, ubah nilai variabel @num dari 0 menjadi 'A'.
5	Sebagai uji coba, masih dengan skrip T-SQL yang sama, ubah nilai variabel @num dari 'A' menjadi 1000000000.

Praktikum Bagian 15 – TRY / CATCH: Menambahkan conditional logic pada blok CATCH

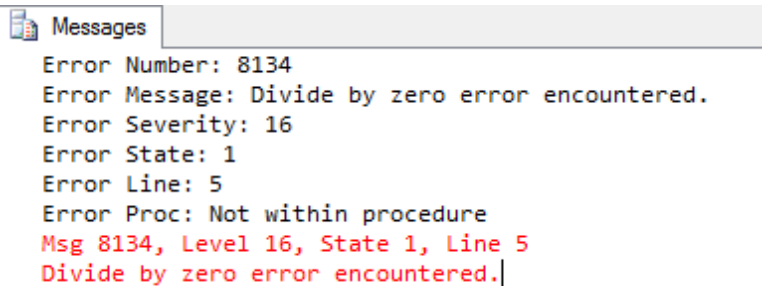
Langkah	Keterangan
1	<p>[Soal-32] Dengan tetap menggunakan skrip T-SQL pada <u>Bagian 12 Langkah 1</u>, lakukan modifikasi dengan menambahkan pernyataan IF pada bagian blok CATCH sebelum pernyataan PRINT.</p> <ul style="list-style-type: none"> Pernyataan IF tersebut untuk mengecek apakah nomer error = 245 atau 8114. <ul style="list-style-type: none"> Apabila kondisi ini terpenuhi, tampilkan pesan “Handling conversion error...” dengan perintah PRINT. Jika tidak sama dengan 245 atau 8114, <ul style="list-style-type: none"> tampilkan pesan “Handling NON conversion error...”. Terakhir, set nilai variabel @num sebagai 'A', lalu eksekusi skrip T-SQL tersebut. <p>Bandingkan hasilnya dengan tampilan berikut:</p> 
2	Sebagai uji coba, masih dengan skrip T-SQL yang sama, ubah nilai variabel @num dari 'A' menjadi 0.



Praktikum Bagian 16 – TRY / CATCH: Mengeksekusi stored procedure pada blok CATCH

1	Salin dan eksekusi skrip T-SQL yang membuat sebuah stored procedure dbo.GetErrorInfo di bawah ini.
---	--

	<pre>CREATE PROCEDURE dbo.GetErrorInfo AS PRINT 'Error Number: ' + CAST(ERROR_NUMBER() AS varchar(10)); PRINT 'Error Message: ' + ERROR_MESSAGE(); PRINT 'Error Severity: ' + CAST(ERROR_SEVERITY() AS varchar(10)); PRINT 'Error State: ' + CAST(ERROR_STATE() AS varchar(10)); PRINT 'Error Line: ' + CAST(ERROR_LINE() AS varchar(10)); PRINT 'Error Proc: ' + COALESCE(ERROR_PROCEDURE(), 'Not within procedure');</pre>
2	<p>[Soal-33] Buatlah sebuah konstruksi TRY / CATCH, dimana pada bagian blok CATCH, lakukan eksekusi stored procedure yang telah dibuat pada Langkah 1 di atas, lalu jalankan.</p>  <p>Messages</p> <p>Error Number: 8134 Error Message: Divide by zero error encountered. Error Severity: 16 Error State: 1 Error Line: 4 Error Proc: Not within procedure</p>

Praktikum Bagian 17 – THROW: Menggunakan THROW untuk mengirimkan kembali pesan error

1	<p>[Soal-34] Modifikasi skrip T-SQL dari Soal-19 di atas dengan menambahkan perintah THROW yang ditempatkan setelah pernyataan eksekusi stored procedure.</p>  <p>Messages</p> <p>Error Number: 8134 Error Message: Divide by zero error encountered. Error Severity: 16 Error State: 1 Error Line: 5 Error Proc: Not within procedure Msg 8134, Level 16, State 1, Line 5 Divide by zero error encountered.</p>
2	<p>[Soal-35] Modifikasi skrip T-SQL dari Soal-20 di atas dengan mengganti perintah THROW dengan pernyataan IF.</p> <ul style="list-style-type: none"> • Pernyataan IF tersebut untuk mengecek apakah nomer error = 8114. • Apabila kondisi ini terpenuhi, tampilkan pesan “Handling division by zero...” dengan perintah PRINT. • Jika tidak, tampilkan pesan “Throwing original error...”. • Terakhir, set nilai variabel @num sebagai ‘A’, lalu eksekusi skrip T-SQL tersebut. <p>Bandingkan hasilnya dengan tampilan berikut:</p>

	 Messages <pre>Error Number: 8114 Error Message: Error converting data type varchar to numeric. Error Severity: 16 Error State: 5 Error Line: 6 Error Proc: Not within procedure Throwing original error Msg 8114, Level 16, State 5, Line 6 Error converting data type varchar to numeric.</pre>
3	<p>[Soal-36] Salinlah skrip T-SQL berikut ini:</p> <pre>DECLARE @msg AS varchar(2048); SET @msg = 'Anda sedang mengerjakan Jobsheet 14 pada ' + FORMAT(CURRENT_TIMESTAMP, 'MMMM d, yyyy', 'en-US') + '. Ini bukan error, bosque';</pre> <p>Lalu, tambahkan pernyataan THROW setelah skrip di atas. Sebagai argumen pertama, isikan 50001, untuk argumen kedua, isikan variabel @msg, sedangkan argumen ketiga, isikan nilai 1.</p> <p>Bandingkan hasilnya dengan tampilan berikut:</p>  Messages <pre>Msg 50001, Level 16, State 1, Line 4 Anda sedang mengerjakan Jobsheet 14 pada January 1, 2018. Ini bukan error, bosque</pre>
5	<p>Untuk menghapus stored procedure yang dibuat sebelumnya, jalankan skrip berikut:</p> <pre>DROP PROCEDURE Sales.CheckPersonBirthDate; -- Praktikum 7 - Langkah 1 DROP PROCEDURE dbo.GetErrorInfo;</pre>

--- Selamat Mengerjakan ----