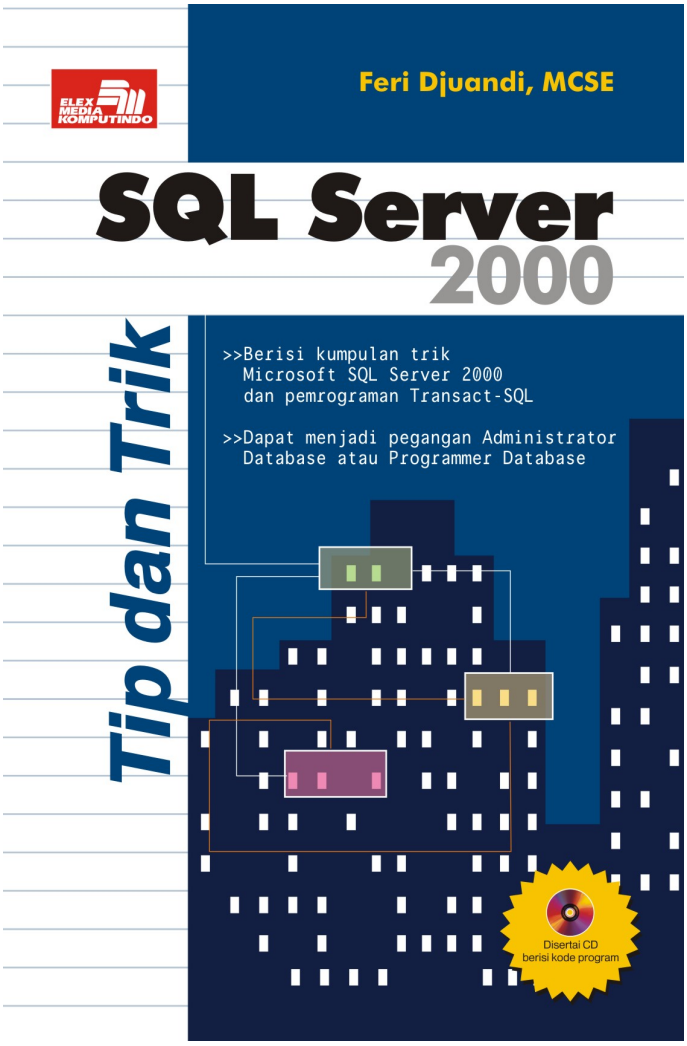


Spesifikasi:

Ukuran: 14x21 cm
Tebal: 196
Harga: Rp 37.800
Terbit pertama: Oktober 2004
Sinopsis singkat:

Microsoft SQL Server 2000 telah dikenal sebagai salah satu server database yang paling banyak digunakan. Salah satu keunggulannya adalah integrasinya dengan sistem operasi Windows sehingga memudahkan operasinya.

Buku ini berisi sejumlah tips dan trik pada Microsoft SQL Server 2000 untuk tujuan administrasi database maupun pemrograman Transact-SQL. Materinya ditujukan bagi Administrator Database dan programmer tingkat lanjut yang telah terbiasa bekerja dengan database dan menguasai perintah-perintah dasar SQL. Beberapa bab pertama dari buku ini membahas tips-tips administrasi database. Pada bagian tersebut, pembaca diajak untuk mengeksplorasi sisi konfigurasi dan optimasi database, pemantauan database dan pemeliharaan database yang berguna pada pekerjaan administratif sehari-hari dalam SQL Server. Bagian terakhir mengulas trik pemrograman T-SQL. Di dalamnya diulas cara berinteraksi dengan SQL Server melalui pembuatan query yang efektif dan efisien. Semua teknik ini dapat dengan mudah diadopsi dan diterapkan pada aplikasi-aplikasi yang sedang dikembangkan.

BAB 4
BACKUP

4.1 Tips Backup Database

Berikut ini adalah 12 tips yang perlu diketahui seorang Administrator Database dalam mem-backup database, terutama jika database yang ditanganinya berukuran besar.

1. Walaupun server Anda dilengkapi tape drive, usahakan melakukan backup database pada harddisk lokal dulu, setelah itu salin atau backup lagi ke dalam tape.

   Kerugian yang dirasakan jika Anda langsung mem-backup database ke dalam tape adalah waktu backup yang dibutuhkan akan lebih lama karena kecepatan tape drive umumnya jauh lebih lambat dari pada harddisk. Kelambanan proses ini juga akan berdampak pada unjuk kerja server karena untuk waktu yang lama server akan terpengaruh proses backup ini, apalagi jika saat itu database juga sedang digunakan banyak pengguna yang melakukan INSERT, UPDATE, dan DELETE terhadap data-data di dalamnya sehingga hal tersebut juga akan membuat proses backup semakin lama.

2. Lakukan backup ke dalam beberapa backup device. Dengan cara ini, SQL Server akan membuat backup *thread* secara terpisah sehingga backup akan dijalankan secara paralel.
3. Backup akan lebih cepat selesai jika dilakukan pada *disk array*. Semakin banyak disk dalam array tersebut, semakin cepat backup diselesaikan.
4. Lakukan backup pada waktu-waktu akses database rendah. Hindari proses backup di tengah hari kerja yang sibuk.
5. Jika Anda mem-backup sebuah database untuk direstorasi lagi di server lain, jalankan *full backup* untuk meminimalkan waktu restorasi database. Walaupun jalannya *full backup* lebih lama daripada *differential* dan *incremental backup*, proses restorasinya merupakan yang tercepat.
6. Jika yang diutamakan adalah lama proses backup, jalankan *incremental backup* karena ia yang tercepat dibandingkan metode yang lainnya. Namun demikian, *incremental backup* akan membutuhkan waktu paling lama saat restorasi database.
7. Pilih *differential backup* dibandingkan *incremental backup* jika data dalam database sering berubah. Hal ini disebabkan differential backup hanya mendeteksi data-data page yang telah berubah sejak backup database terakhir dilakukan. Dengan cara ini, waktu yang dibutuhkan server untuk melakukan rolling forward transaction saat melakukan *recovery* transaction log pada incremental backup bisa banyak dihemat. Pada kasus ini, differential backup akan meningkatkan proses *recovery* secara signifikan.
8. Usahakan untuk menempatkan database Anda pada beberapa file dan *filegroup* sehingga Anda memiliki kesempatan untuk mem-backup file/filegroup tertentu saja.
9. Gunakan Windows NT Performance Monitor atau Windows 2000 System Monitor untuk memantau unjuk kerja server selama proses backup berlangsung. Counter yang bisa Anda pantau adalah:

- ☑ **SQL Server Backup Device: Device Throughput Bytes/sec**, untuk memantau hasil pada backup device (bukan operasi backup/restore secara keseluruhan).
- ☑ **SQL Server Databases: Backup/Restore Throughput/sec**, untuk memantau hasil backup/restore secara keseluruhan.
- ☑ **PhysicalDisk: % Disk Time**, untuk memonitor aktivitas baca/tulis.
- ☑ **Physical Disk Object: Avg. Disk Queue Length**, untuk memantau antrian permintaan akses ke disk.

10. Untuk mengurangi waktu backup, usahakan untuk mem-backup lebih sering (tapi ingat bahwa backup tetap dianjurkan dilakukan pada waktu akses database-nya rendah). Dengan differential backup, semakin sering Anda melakukan backup, semakin cepat prosesnya selesai karena SQL Server hanya menangkap bagian-bagian data yang berubah.

    Tentunya keuntungan melakukan backup yang lebih sering adalah Anda mendapatkan backup data yang lebih *up-to-date* sehingga kalaupun terjadi musibah, Anda bisa mengandalkan backup tersebut.

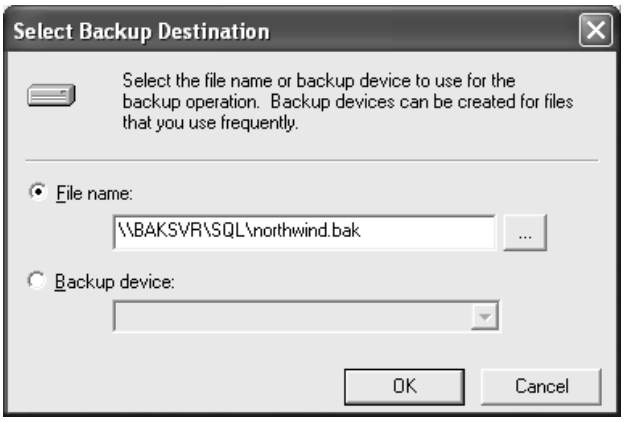
11. Tempatkan tape drive pada SCSI bus yang terpisah dari disk drive atau CD-ROM drive. Kebanyakan tape drive akan bekerja lebih baik jika ia memiliki SCSI bus tersendiri untuk setiap tape drive yang digunakan. Keberadaan bus untuk setiap tape drive akan memaksimalkan unjuk kerja backup dan mencegah konflik dengan drive lain. Microsoft menyarankan penggunaan SCSI bus tersendiri untuk setiap tape drive dengan rasio transfer lebih dari 50% dari kecepatan SCSI bus-nya.

12. Gunakan *snapshot backup* dari SQL Server 2000 untuk database yang berukuran sangat besar. Teknologi snapshot backup dan restore mendukung hardware dan software khusus yang dibuat oleh vendor pihak ketiga. Keuntungan paling besar dari snapshot backup adalah waktu operasinya yang dapat menjadi sangat singkat.

4.2 Backup ke Network Share

Saat Anda hendak mem-backup sebuah database dengan SQL Enterprise Manager, perhatikan bahwa program tersebut tampaknya hanya mengizinkan Anda untuk menentukan lokasi backup device pada drive atau direkrori lokal dari komputer yang bersangkutan. Sekalipun Anda pernah membuat mapping drive dengan sebuah network share (misalnya menggunakan perintah *NET USE*), drive tersebut tidak akan muncul pada SQL Enterprise Manager. Dengan kata lain, sepertinya SQL Server tidak memungkinkan Anda untuk menempatkan hasil backup sebuah database ke network share.

Kenyataannya tidak begitu. Anda bisa menempatkan hasil backup sebuah database ke network share. Hal ini memang tidak didokumentasikan, namun caranya sangat mudah. Saat SQL Enterprise Manager menanyakan lokasi hasil backup, isilah network share dan nama file backup-nya pada kolom **File name** sesuai *aturan Uniform Naming Convention (UNC)*. Setelah itu klik tombol **OK** dan proses backup akan berjalan normal.



Gambar 4.1 Menentukan lokasi hasil backup

Pastikan bahwa network share tersebut dapat diakses secara penuh (*full control*) oleh service account SQL Server tersebut.

4.3 Menyusutkan Transaction Log

Jika Anda memiliki sebuah database yang sudah beroperasi dalam jangka waktu lama, Anda akan menemukan bahwa transaction log database tersebut membengkak menjadi ukuran yang cukup besar. Bukan tidak mungkin bahwa ukurannya melebihi ukuran datanya itu sendiri. Hal ini bisa terjadi karena beberapa sebab:

1. Anda tidak atau jarang sekali mem-backup database tersebut. Jika Anda tidak melakukan backup, transaction log tidak pernah dikosongkan sehingga semua catatan aktivitas database akan terus ditambahkan ke dalam transaction log dan ukurannya akan terus membengkak, menyita ruang harddisk komputer. Jika Anda mem-backup data atau transaction log dari database tersebut, SQL Server secara otomatis akan mengosongkan transaction log sehingga di dalamnya akan tersedia cukup ruang kosong untuk mencatat aktivitas baru tanpa harus menambah ukuran transaction log tersebut.
2. Transaction log yang terus membengkak juga disebabkan konfigurasi database yang mengaktifkan opsi *autogrow* sehingga saat SQL Server mendeteksi bahwa transaction log tidak memadai lagi untuk menampung catatan aktivitas database, SQL Server secara otomatis akan memperbesar ukuran transaction log tersebut. Walaupun fitur ini meringankan pekerjaan seorang Administrator Database, hal itu juga akan mengurangi kontrol Anda terhadap pertumbuhan database. Apabila Anda jarang memantau ukuran database, tidak aneh jika suatu saat Anda dikejutkan oleh transaction log yang berukuran raksasa.

Untuk menghindari ukuran transaction log yang besar sekali, secara periodik Anda harus melakukan pemeliharaan database. Ada beberapa hal yang bisa dilakukan, yaitu:

1. Mem-backup data dari database. Pada SQL Server dikenal dua macam cara untuk mem-backup database, yaitu *full backup* dan *incremental backup*. Selain akan mem-backup data ke dalam file atau media lain (misalnya tape), kedua cara ini juga akan mengosongkan transaction log. Untuk mengetahui cara

mem-backup data, silahkan membaca *SQL Server Online Books*.

2. Mem-backup transaction log dari database. Sama halnya dengan data, Anda juga dapat mem-backup transaction log agar setelah langkah ini dilakukan maka akan tersedia ruang kosong pada transaction log tersebut. Untuk mengetahui cara mem-backup transaction log, silahhkan membacanya *SQL Server Online Books*.

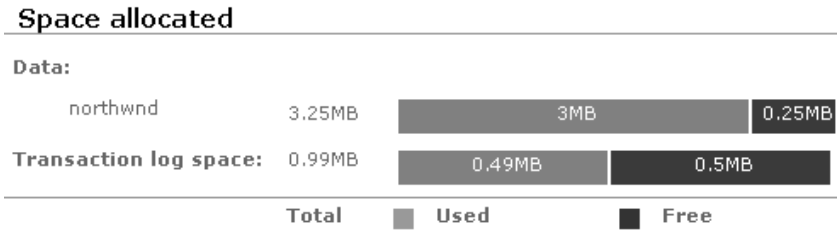
   Apabila Anda hanya ingin mengosongkan transaction log, sebetulnya Anda tidak perlu mem-backup-nya ke dalam file atau media, tetapi Anda bisa langsung mengosongkannya. Sebagai contoh, perintah berikut ini akan menghapus isi transaction log.

   ```
   BACKUP LOG northwind WITH NO_LOG
   ```

   Walaupun menggunakan perintah BACKUP, instruksi di atas akan mengosongkan transaction log tanpa sungguh-sungguh mem-backup-nya.

Apabila Anda menjalankan perintah BACKUP, yang dilakukan pada transaction log hanyalah mengosongkan isinya. Ukuran fisik transaction log itu sendiri tidak akan berubah. Jika Anda terlanjur memiliki transaction log yang berukuran besar, Anda tidak bisa berharap bisa memperkecilnya dengan hanya melakukan BACKUP. Untuk lebih memahami penjelasan ini, kita akan membuat sebuah eksperimen dengan database *Northwind*.

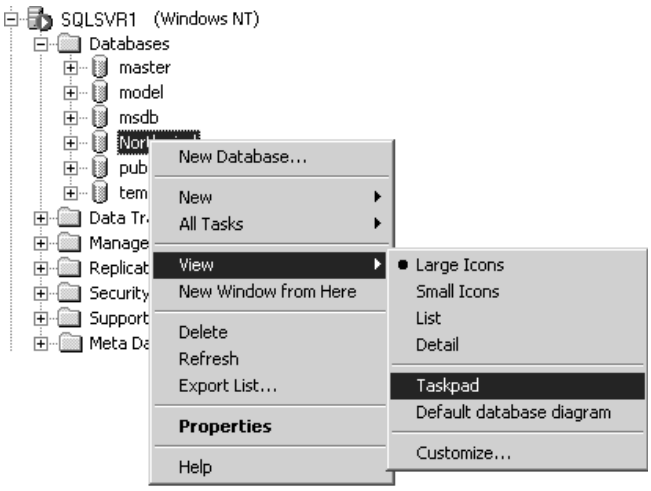
Pada gambar di bawah ini diperlihatkan diagram mengenai ukuran database *Northwind*.



Gambar 4.2 Ukuran database mula-mula

Anda bisa mendapatkan gambar ini pada program **SQL Server Enterprise Manager** dengan cara:

1. Pada treeview, klik database **Northwind**.
2. Klik-kanan pada database **Northwind**, kemudian pilih menu **View | Taskpad**.



Gambar 4.3 Menampilkan informasi database

Pada diagram database tersebut, Anda melihat bahwa ukuran datanya adalah 3.25 MB (3 MB terisi data dan 0.25 MB kosong), sedangkan ukuran transaction log-nya adalah 0.99 MB (0.49 MB terisi data dan 0,5 MB kosong).

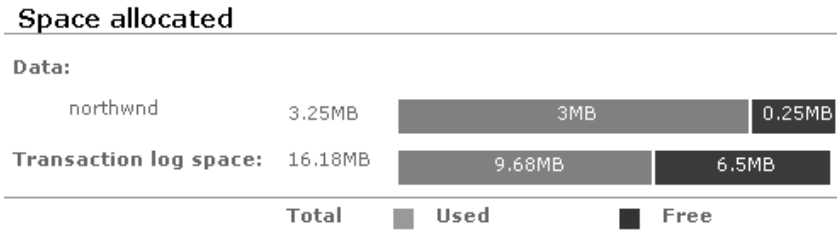
Sekarang kita akan membuat simulasi untuk mengisi 10.000 baris data ke dalam sebuah table. Dengan cara ini, SQL Server akan mencatat aktivitas database tersebut ke dalam transaction log sehingga dengan sengaja kita akan membuatnya penuh dan ukurannya membengkak (sebelumnya, pastikan bahwa opsi *autogrow* untuk transaction log telah diaktifkan).

Silahkan eksekusi kode program di bawah ini pada **SQL Query Analyzer**. Tunggu beberapa menit sampai program selesai.

Perhatikan bahwa selama waktu tersebut aktivitas SQL Server akan meningkat drastis.

```
USE Northwind
GO
DECLARE @I int

IF NOT EXISTS(SELECT * FROM sysobjects WHERE
name='Dummy Customer')
  SELECT *
  INTO Dummy Customer
  FROM Customers
  WHERE 1=0

SET @I=0

BEGIN TRAN
WHILE @I < 10000
BEGIN
  INSERT Dummy_Customer
  SELECT TOP 1 *
  FROM Customers
  ORDER BY NEWID()

  DELETE Dummy_Customer

  SET @I=@I + 1
END
IF @@TRANCOUNT > 0
  COMMIT TRAN
GO
DROP TABLE Dummy_Customer
```

Setelah program berakhir, kembalilah ke **SQL Server Enterprise Manager** dan *refresh* tampilan diagram database *Northwind* tadi. Sekarang Anda bisa melihat bahwa ukuran transaction log telah membengkak jauh melampaui ukuran sebelumnya.



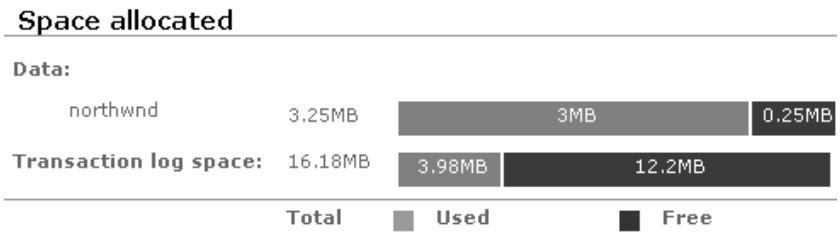
Gambar 4.4 Ukuran database yang bertambah besar

Pada gambar di atas tampak bahwa ukuran transaction log berubah menjadi 16,18 MB dimana ruang yang terisi adalah 9.68 MB. Fantastis bukan? Hanya dalam beberapa menit, Anda telah membuat ukurannya berlipat ganda. Ini juga bisa terjadi pada database Anda. Bayangkan seandainya Anda terus-menerus menggunakan database tanpa pernah mempedulikan transaction log!

Jika Anda menjalankan lagi program tadi berulang-ulang, ukuran transaction log akan terus bertambah besar. Agar ukurannnya tidak semakin membengkak, Anda perlu menghapus isi transaction log. Pengosongan transaction log dapat dilakukan dengan menjalankan sebaris perintah berikut ini pada **SQL Query Analyzer**.

```
BACKUP LOG northwind WITH NO_LOG
```

Setelah perintah tersebut dijalankan, *refresh* lagi tampilan diagram database pada **SQL Server Enterprise Manager**. Sekarang Anda bisa melihat bahwa sebagian besar isi transaction log telah dihapus sehingga tersisa 12,2 MB ruang kosong. Pada saat ini, jika Anda menjalankan program simulasi tadi sekali lagi maka ukuran transaction log tidak akan bertambah karena di dalamnya ada ruang kosong yang cukup besar untuk mencatat aktivitas database. Inilah gunanya Anda mengosongkan transaction log.



Gambar 4.5 Kondisi database setelah pengosongan transaction log

Walaupun transaction log telah dihapus, ukuran fisiknya tetap belum berubah, yaitu tetap 16,18 MB, baik sebelum maupun sesudah perintah BACKUP LOG dieksekusi. Jika Anda ingin agar ukurannya kembali ke ukuran semula (sekitar 1 MB), Anda dapat menjalankan perintah berikut ini.

```
DBCC SHRINKFILE ( Northwind_log,1)
```

Perintah DBCC SHRINKFILE akan menyusutkan ukuran transaction log ke ukuran yang diinginkan. Parameter yang harus dimasukkan ke dalam perintah tersebut adalah:

1. Parameter pertama adalah nama transaction log. Nama yang dimaksud di sini adalah nama logikal, bukan nama file fisiknya. Untuk mengetahui nama-nama logikal dari file-file database, Anda dapat menjalankan perintah berikut ini.

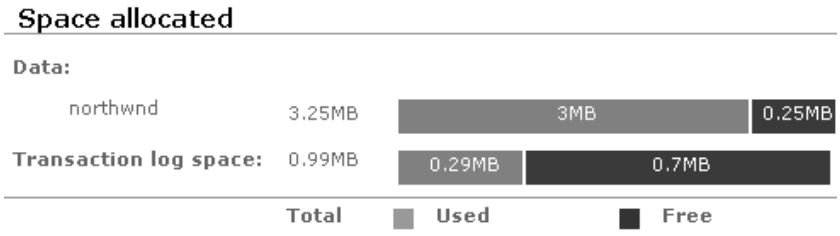
```
sp_helpdb northwind
```

```
name            db_size       owner
--------------- ------------- --------------------
Northwind            19.43 MB sa

name            fileid filename
--------------- ------ ----------------------------
Northwind       1      C:\SQL_data\northwnd.mdf
Northwind_log   2      C:\SQL_data\northwnd.ldf
```

Gambar 4.6 Nama-nama logical dari file-file database

2. Parameter kedua adalah ukuran yang diinginkan dalam satuan MB. Anda tidak perlu kuatir untuk menentukan ukurannya. Jika SQL Server mendeteksi bahwa transaction log tidak mungkin disusutkan sampai ke ukuran tersebut, SQL Server akan melaporkannya pada Anda.

Jika perintah tersebut berhasil dieksekusi, Anda akan melihat pada diagram database bahwa ukuran transaction log telah menyusut ke ukuran yang diinginkan.



Gambar 4.7 Kondisi database setelah penyusutan transaction log

Dari eksperimen ini, kita dapat menarik beberapa kesimpulan:

1. Anda tidak boleh mengabaikan prosedur pemeliharaan database. Sebagai seorang Administrator Database, Anda harus senantiasa memantau database dan pertumbuhannya. Secara periodik Anda harus melakukan backup database atau mengosongkan transaction log agar ukurannya tidak membengkak tidak terkendali.
2. Walaupun pengosongan transaction log mutlak harus dilakukan, penyusutan ukuran fisik transaction log tidak perlu dilakukan kecuali Anda memiliki alasan yang khusus. Penjelasannya adalah karena bagaimana pun Anda tidak bisa mencegah transaction log untuk bertumbuh. Jika database Anda sedang beroperasi dan saat itu aktivitasnya sedang tinggi sehingga SQL Server harus menambah ukuran transaction log, Anda tidak bisa menahan hal itu terjadi. Jika sesudahnya Anda menyusutkan lagi transaction log ke ukuran yang kecil tapi kemudian aktivitas database tinggi lagi maka transaction log akan membesar lagi ke ukuran tertentu. Jadi Anda tidak perlu melakukan hal yang sia-sia seperti itu, bukan? Bahkan sebaliknya, ukuran transaction log yang besar juga bisa memberikan keuntungan. Anda harus menyadari bahwa setiap kali SQL Server menambah ukuran transaction log, kinerja database akan melorot drastis untuk beberapa detik hingga beberapa menit. Hal ini dikarenakan adanya tambahan pekerjaan yang dibebankan kepadanya untuk memperbesar transaction log. Akan tetapi jika transaction log Anda sudah cukup besar, overhead ini bisa dieliminasi sehingga performa database relatif stabil.

4.4 Memindahkan File Database

Jika Anda memiliki sebuah database dengan ukuran yang besar sementara ruang kosong pada disk penyimpanan database itu akan segera habis, Anda perlu memindahkan file-file tersebut ke disk lain sebelum mereka betul-betul memenuhi isi disk itu. Untuk memindahkan file-file database, silahkan ikuti langkah-langkah berikut ini:

1. Pastikan bahwa tidak satu pengguna pun yang terkoneksi ke database tersebut.
2. Dari database *master*, jalankanlah stored procedure *sp_helpdb* untuk melihat informasi dimana file-file database-nya diletakkan. Pada umumnya file-file data akan mempunyai ekstensi MDF atau NDF, sedangkan file transaction log akan berekstensi LDF. Contoh:

   ```
   EXEC sp_helpdb 'pubs'
   ```

   Jumlah file-file itu akan bervariasi, bergantung pada bagaimana database itu dulu dibuat.
3. Bila Anda telah siap, jalankanlah stored procedure *sp_detach_db* dari database *master*, diikuti nama database-nya dengan diapit tanda kutip. Contoh:

   ```
   EXEC sp_detach_db 'pubs'
   ```
4. Setelah menjalankan *sp_detach_db*, database tersebut akan "terlepas" dari SQL Server dan database itu tidak akan dikenali lagi. Namun demikian, file-file databasenya (data dan transaction log) masih tetap ada pada tempat semula.
5. Silahkan pindahkan file-file tersebut secara manual ke disk lain.
6. Setelah file-file selesai dipindahkan, silahkan kembali ke database *master* dan jalankan stored procedure *sp_attach_db* diikuti nama database dan semua nama filenya berikut lokasi lengkapnya. Contoh:

   ```
   EXEC sp attach db @dbname = 'pubs',
   @filename1 = 'd:\SQLData\pubs.mdf',
   @filename2 = 'd:\SQLData\pubs_log.ldf'
   ```

   Bila langkah ini berhasil dijalankan, database tersebut akan kembali terdaftar pada SQL Server dan dapat digunakan seperti biasa.

Catatan:

Cara lain untuk mengatasi masalah file-file yang memenuhi disk adalah dengan menambahkan file-file database baru yang dibuat pada disk lain dan membiarkan file-file yang ada sebelumnya pada tempat semula. Agar file-file yang semula tidak semakin membengkak, Anda harus menonaktifkan opsi autogrow pada masing-masing file.

4.5 Membuat Skrip Objek

Dengan SQL Enterprise Manager, Anda dapat membuat skrip pembuatan objek-objek database (CREATE). Namun demikian, hal tersebut juga bisa dilakukan pada SQL Query Analyzer dengan membuat sebuah program SQL yang cukup pendek. Cara kerja program ini adalah dengan memanfaatkan fungsi-fungsi di komponen **SQL Distributed Management Objects** (**SQL-DMO**) yang dipanggil menggunakan stored procedure *sp_OACreate*, *sp_OASetProperty*, dan *sp_OAMethod*.

Stored procedure *sp_OACreate* digunakan untuk membuat sebuah instan objek OLE pada SQL Server.

Sintaks:

```
sp_OACreate progid, | clsid, objecttoken OUTPUT
[, context ]
```

Keterangan

progid : String, adalah programmatic identifier (ProgID) dari objek OLE yang akan dibuat. Untuk SQL-DMO, ProgID-nya adalah 'SQLDMO.SQLServer'.

clsid : String, adalah class identifier (CLSID) dari objek OLE yang akan dibuat. CLSID adalah sebuah rangkaian karakter yang menggambarkan ID class dari objek tersebut. Untuk SQL-DMO, CLSID-nya adalah:

'{00026BA1-0000-0000-C000-000000000046}'

Untuk menjalankan sp_OACreate, Anda boleh

menggunakan ProgID atau CLSID, tapi sebaiknya Anda menggunakan ProgID karena tentunya lebih mudah dibaca.

objecttoken [OUTPUT] : Integer, adalah handle dari instan objek yang dibuat. Handle ini harus dimasukkan ke dalam sebuah variabel yang dideklarasikan lokal. Handle ini digunakan untuk mengacu instan objek tersebut selama penggunaannya.

context : Integer, opsional. Menentukan konteks eksekusi dari instan yang dibuat. Nilai-nilai yang mungkin:

1 = In-process (.dll) OLE server only

4 = Local (.exe) OLE server only

5 = Both in-process and local OLE server allowed

Defaultnya adalah 5.

Stored procedure ini mengembalikan nilai integer, yaitu 0 jika berhasil atau nilai lain jika gagal. Untuk menampilkan pesan kesalahan, Anda dapat menggunakan *sp_OAGetErrorInfo*.

Stored procedure *sp_OASetProperty* digunakan untuk menset property dari sebuah instan objek.

Sintaks:

```
sp_OASetProperty objecttoken, propertyname,
    newvalue [ , index... ]
```

Keterangan

objecttoken : Integer, adalah handle dari instan objek.

propertyname : String. Nama property.

newvalue : Nilai yang akan diset ke dalam property tersebut. Tipe datanya harus berkesesuaian dengan property yang bersangkutan.

index : Integer. Index dari property jika ada.

Stored procedure ini mengembalikan nilai integer, yaitu 0 jika berhasil atau nilai lain jika gagal.

Stored procedure *sp_OAMethod* digunakan untuk menjalankan fungsi dari sebuah instan objek.

Sintaks:

```
sp_OAMethod objecttoken, methodname
    [ , returnvalue OUTPUT ]
    [ , [ @parametername = ] parameter [ OUTPUT ]
    [ ...n ] ]
```

Keterangan

objecttoken	:	Integer, adalah handle dari instan objek.
methodname	:	String. Nama fungsi.
returnvalue [OUTPUT]	:	Nilai yang dikembalikan oleh fungsi tersebut. Tipe datanya harus bersesuaian dengan kembalian dari fungsi yang bersangkutan.
[@parametername =] parameter [OUTPUT]	:	Nilai parameter yang dimasukkan ke dalam fungsi tersebut (jika ada). Apabila sebuah parameter adalah *pass-by-value*, Anda bisa langsung memasukkan nilainya ke dalam stored procedure ini, tetapi jika parameter tersebut adalah *pass-by-reference*, Anda harus menggunakan sebuah variabel lokal diikuti OUTPUT. Anda dapat mengirim beberapa parameter dengan masing-masing dipisahkan oleh koma. Urutan parameter-parameter yang dikirim harus bersesuaian dengan ukuran parameter dari fungsi tersebut. Namun demikian, Anda dapat juga mengirimkan parameter-parameter dengan urutan yang tidak sesuai jika Anda menggunakan *@parametername*. Perhatikan bahwa *@parametername* bukan sebuah variabel lokal, melainkan nama parameter dari fungsi tersebut.

Stored procedure ini mengembalikan nilai integer, yaitu 0 jika berhasil, atau nilai lain jika gagal.

Anda juga dapat menggunakan *sp_OAMethod* untuk mengambil nilai sebuah property.

Stored procedure *sp_OADestroy* digunakan untuk menghapus sebuah instan. Sintaks:

```
sp_OADestroy objecttoken
```

Berikut ini adalah kode program SQL untuk membuat skrip objek.

```
CREATE PROCEDURE p_object_script @obj_name varchar(50)=NULL AS
/************************************************
Author  : Feri Djuandi
Created : 21 May 2003
Desc. : Generate Object Script
************************************************/
DECLARE @db_name  varchar(50),
  @obj type char(2),
  @command varchar(255),
  @svr object int,
  @ret int,
  @obj_script varchar(5000)
SET NOCOUNT ON
--Get current database name
SELECT @db_name=LEFT(the_db.name , 50)
FROM master..sysprocesses the_spid, master..sysdatabases the_db
WHERE the_spid.spid=@@SPID AND the_spid.dbid=the_db.dbid

EXEC @ret = sp_OACreate 'SQLDMO.SQLServer', @svr_object OUT

IF @ret <> 0
BEGIN
   EXEC sp_OAGetErrorInfo @svr_object
   RETURN
END

--Win NT login
EXEC @ret = sp OASetProperty @svr object, 'LoginSecure', TRUE

/* SQL Server Login
EXEC @ret = sp OASetProperty @svr object, 'Login', 'sa'
EXEC @ret = sp_OASetProperty @svr_object, 'password', '******'
*/

SET @command = 'Connect('+RTRIM(@@SERVERNAME) +')'
EXEC @ret = sp_OAMethod @svr_object,@command

SET @obj_type=NULL
SET @obj_type=(SELECT dbobj.xtype FROM sysobjects dbobj
  WHERE dbobj.id=OBJECT_ID(@obj_name) )
```

```
SET @command = (CASE
 WHEN  @obj type  IS  NULL  THEN  'Databases("'  +  @db name  +
'").Script(4)'
 WHEN   @obj type='P'   THEN   'Databases("'   +   @db name   +
'").StoredProcedures("'+ @obj_name + '").Script(4)'
 WHEN   @obj_type='V'   THEN   'Databases("'   +   @db_name   +
'").Views("'+ @obj_name + '").Script(4)'
 WHEN   @obj_type='U'   THEN   'Databases("'   +   @db_name   +
'").Tables("'+ @obj_name + '").Script(532750364)'
 END)

EXEC @ret = sp_OAMethod @svr_object, @command, @obj_script OUTPUT
EXEC @ret = sp_OADestroy @svr_object

PRINT @obj_script
```

Program ini bisa digunakan untuk membuat skrip database, table, view dan stored procedure. Contoh untuk membuat skrip dari table Order:

```
EXEC p_object_script 'Orders'
```

Ouput:

```
CREATE TABLE [Orders] (
 [OrderID] [int] IDENTITY (1, 1) NOT NULL ,
 [CustomerID] [nchar] (5) COLLATE SQL_Latin1_General_CP1_CI_AS
 NULL , [EmployeeID] [int] NULL , [OrderDate] [datetime] NULL ,
 [RequiredDate] [datetime] NULL ,
 [ShippedDate] [datetime] NULL , [ShipVia] [int] NULL ,
 [Freight] [money] NULL CONSTRAINT [DF_Orders_Freight] DEFAULT
 (0), [ShipName] [nvarchar] (40) COLLATE
 SQL Latin1 General CP1 CI AS NULL ,
 [ShipAddress] [nvarchar] (60) COLLATE
 SQL Latin1 General CP1 CI AS NULL ,
 [ShipCity] [nvarchar] (15) COLLATE SQL_Latin1_General_CP1_CI_AS
 NULL , [ShipRegion] [nvarchar] (15) COLLATE
 SQL_Latin1_General_CP1_CI_AS NULL ,
 [ShipPostalCode] [nvarchar] (10) COLLATE
 SQL_Latin1_General_CP1_CI_AS NULL ,
 [ShipCountry] [nvarchar] (15) COLLATE
 SQL_Latin1_General_CP1_CI_AS NULL ,
 CONSTRAINT [PK_Orders] PRIMARY KEY  CLUSTERED
 (
  [OrderID]
 )  ON [PRIMARY] ,
 CONSTRAINT [FK_Orders_Customers] FOREIGN KEY
 (
  [CustomerID]
 ) REFERENCES [Customers] (
  [CustomerID]
 ),
 CONSTRAINT [FK Orders Employees] FOREIGN KEY
 (
```

```
    [EmployeeID]
  ) REFERENCES [Employees] (
    [EmployeeID]
  ),
  CONSTRAINT [FK Orders Shippers] FOREIGN KEY
  (
    [ShipVia]
  ) REFERENCES [Shippers] (
    [ShipperID]
  )
) ON [PRIMARY]
GO

 CREATE     INDEX   [CustomerID]   ON   [Orders]([CustomerID])   ON
[PRIMARY]
GO

 CREATE    INDEX  [CustomersOrders]  ON  [Orders]([CustomerID])  ON
[PRIMARY]
GO

 CREATE     INDEX   [EmployeeID]   ON   [Orders]([EmployeeID])   ON
[PRIMARY]
GO

 CREATE    INDEX  [EmployeesOrders]  ON  [Orders]([EmployeeID])  ON
[PRIMARY]
GO

 CREATE  INDEX [OrderDate] ON [Orders]([OrderDate]) ON [PRIMARY]
GO

 CREATE     INDEX   [ShippedDate]   ON   [Orders]([ShippedDate])   ON
[PRIMARY]
GO

 CREATE     INDEX   [ShippersOrders]   ON   [Orders]([ShipVia])   ON
[PRIMARY]
GO

 CREATE  INDEX [ShipPostalCode] ON [Orders]([ShipPostalCode]) ON
[PRIMARY]
GO
```

Skrip: *4_5.SQL*

4.6 Menyimpan Skrip Stored Procedure

Apakah Anda pernah memodifikasi sebuah stored procedure dan kemudian menyadari bahwa Anda telah tidak sengaja membuat kesalahan sehingga Anda membutuhkan skrip yang sebelumnya? Tidak perlu ragu-ragu menjawab “ya” karena Anda tidak sendirian.

Anda masih beruntung jika memiliki backup database, namun jika tidak maka Anda harus bekerja ekstra untuk merekonstuksi skrip sebelumnya. Jika sering mengalami hal seperti ini, Anda pasti setuju dengan saya bahwa akan lebih baik jika Anda dapat menyimpan skrip-skrip stored procedure secara historikal sehingga Anda bisa memiliki versi-versi skrip dari stored procedure yang bersangkutan. Keuntungan lainnya datang jika Anda mengenkripsi skrip stored procedure. Jika Anda mengenkripsi sebuah stored procedure sementara Anda tidak memiliki salinan skripnya maka itu adalah petaka besar. Sebuah stored procedure yang sudah dienkripsi tidak akan bisa dibaca lagi skripnya dan tidak ada satu carapun untuk mendekripsinya.

Salah satu cara termudah untuk menyimpan skrip stored procedure adalah dengan menyimpannya sebagai file text. Namun demikian ini tidak menjamin bahwa Anda secara disiplin menyimpan setiap perubahan stored procedure, apalagi jika Anda melakukan perubahan pada banyak stored procedure dan ada kemungkinan tertinggal satu atau dua buah.

Ide lainnya adalah menggunakan Visual Source Safe. Software ini cukup bagus dan sangat membantu jika Anda memiliki beberapa programmer yang bekerja dalam sebuah proyek. Visual Source Safe dapat menyimpan versi-versi stored procedure dan ia pun dapat mencegah update simultan (timpa-menimpa). Walaupun ini adalah konsep yang bagus, software ini tetap tidak bisa menjamin seseorang langsung masuk ke database (*by-pass* Visual Source Safe) dan membuat banyak perubahan di sana. Namun demikian, hal ini tidak akan terjadi jika semua orang mau bekerja sama dan konsisten menggunakan Visual Source Safe. Akan tetapi percayalah, di saat situasi sangat mendesak dan Anda berada di bawah tekanan untuk menangani sebuah masalah dengan cepat, Anda akan mengabaikan semua aturan-aturan itu agar masalahnya selesai sesegera mungkin.

Daripada menggunakan cara di atas, sebaiknya kita melakukannya dengan pendekatan yang lain. Ingat bahwa tujuan kita hanyalah menyimpan salinan skrip-skrip stored procedure termasuk versi-versinya, dan ini sebetulnya bisa dilakukan dengan mudah. Hal yang perlu kita lakukan adalah membuat sebuah skrip

yang dijalankan untuk menyimpan teks stored procedure ke dalam sebuah table. Skrip tersebut bisa dijalankan satu per satu untuk setiap stored procedure atau sekaligus untuk menyimpan semua skrip stored procedure di dalam sebuah database.

Untuk langkah pertama, buatlah sebuah table pada database Anda yang fungsinya menampung teks stored procedure. Perintah untuk membuat table tersebut adalah sebagai berikut.

```
CREATE TABLE [dbo].[SPscript] (
[sp_id] [int] NOT NULL ,
  [sp_name] [varchar] (50) NOT NULL ,
  [sp_version] [int] NOT NULL ,
  [colid] [int] NOT NULL ,
  [insert_date] [datetime] NOT NULL default GETDATE(),
[insert_by] [varchar] (30) NOT NULL default USER_NAME(),
  [sp_text] [varchar] (4000) NULL
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[SPscript] WITH NOCHECK ADD
  CONSTRAINT [PK_SPscript] PRIMARY KEY  CLUSTERED
  (
    [sp_id], [sp_version], [colid]
  )  ON [PRIMARY]
GO
```

Langkah berikutnya adalah membuat sebuah stored procedure yang berisi program pendek untuk menyimpan skrip stored procedure ke dalam table SPscript di atas. Skrip sebuah stored procedure disimpan di dalam table sistem *syscomments* sehingga dengan mudah Anda bisa menyalin teksnya dan memasukkannya ke kolom *sp_text* dari table SPscript.

Di table *syscomments*, skrip stored procedure disimpan di dalam kolom bernama *text* yang bertipe nvarchar(4000). Oleh karena panjang karakter sebuah stored procedure bisa melampaui 4000 karakter, SQL Server akan memotong stored procedure tersebut menjadi beberapa potongan. Sebagai akibatnya, sebuah stored procedure yang sama bisa disimpan menjadi beberapa baris di dalam table *syscomments*. Di table tersebut juga ada sebuah kolom yang bernama *colid*. Kolom ini berisi nomor urut dari potongan-potongan skrip. Jadi, Anda bisa menggabungkan kembali potongan-potongan stored procedure berdasarkan nomor-nomor *colid* tersebut untuk memperoleh skrip selengkapnya.

```
CREATE PROCEDURE p_save_SP_script @sp_name varchar(50) AS
```

```
/*************************************************
Author  : Feri Djuandi
Created : September 2003
Desc. : Save SP text into table
*************************************************/
DECLARE @obj_id int,
  @obj_name  varchar(50),
  @i int,
  @last_version int

SET NOCOUNT ON
IF RTRIM(@sp_name)<>'*'
BEGIN
  IF EXISTS (SELECT * FROM sysobjects
  WHERE name=@sp_name AND xtype='P' AND status > 0)
  BEGIN
    SELECT @obj_id=id FROM sysobjects
    WHERE name=@sp_name

  SET @last_version=ISNULL((SELECT MAX(sp_version)
     FROM SPscript WHERE sp_id=@obj_id),0)

    SET @last version=@last version + 1

    INSERT SPscript
    (sp id, sp name, sp version, sp text, colid)
  SELECT @obj_id, @sp_name, @last_version, a.text, a.colid
    FROM syscomments a
    WHERE a.id=@obj_id
  END

  RETURN
END

DECLARE cursor_obj INSENSITIVE  CURSOR
FOR SELECT id, name FROM sysobjects WHERE xtype='P' AND status >
0
FOR READ ONLY

OPEN cursor_obj

IF @@CURSOR_ROWS > 0
BEGIN
  SET @i=0
  WHILE @i=0
  BEGIN
  FETCH NEXT FROM cursor_obj INTO @obj_id, @obj_name
    SET @i=@@FETCH_STATUS
    IF @i =0
      EXEC p_save_SP_script @obj_name
  END
END

CLOSE cursor_obj

DEALLOCATE cursor_obj
```

Untuk menggunakan program ini, Anda cukup mengirimkan sebuah parameter yang berisi nama sebuah stored procedure. Jika Anda hendak menyimpan skrip semua stored procedure sekaligus, Anda dapat mengirimkan parameter asterisk (*). Sebagai contoh, untuk menyimpan skrip stored procedure yang bernama *AllSalesByProduct* Anda dapat mengetikkan perintah:

```
EXEC p_save_SP_script 'AllSalesByProduct'
```

Setelah perintah tersebut dieksekusi, table *SPscript* akan berisi sebuah baris seperti diperlihatkan pada gambar di bawah ini.

sp_id	sp_name	sp_ver	colid	insert_date	insert_by	sp_text
1701581	AllSalesByProduct	1	1	9/2/2003 11:	dbo	CREATE PROCEDURE AllSalesByProduct

Gambar 4.8 Skrip stored procedure disimpan dalam table

Jika Anda menjalankan perintah yang sama sekali lagi, pada table tersebut akan dimasukkan lagi sebuah baris yang sama namun dengan versi yang lebih baru.

sp_id	sp_name	sp_ver	colid	insert_date	insert_by	sp_text
1701581	AllSalesByProduct	1	1	9/2/2003 11:	dbo	CREATE PROCEDURE AllSalesByProduct
1701581	AllSalesByProduct	2	1	9/2/2003 11:	dbo	CREATE PROCEDURE AllSalesByProduct

Gambar 4.9 Skrip stored procedure dengan versi berbeda

Untuk menyimpan semua stored procedure sekaligus, Anda dapat menjalankan perintah:

```
EXEC p_save_SP_script '*'
```

Di bawah ini diperlihatkan isi table SPscript setelah perintah tersebut dieksekusi.

sp_id	sp_name	sp_ver	colid	insert_date	insert_by	sp_text
3409916	p_get_invouce_nur	1	1	9/2/2003 11:	dbo	CREATE PROCEDURE p_get_invouce_n
146099	p_display_modified	1	1	9/2/2003 11:	dbo	CREATE PROCEDURE p_display_modifie
162099	p_calendar	1	1	9/2/2003 11:	dbo	CREATE PROCEDURE p_calendar @the
5141008	p_save_SP_script	1	1	9/2/2003 11:	dbo	CREATE PROCEDURE p_save_SP_scripl
6421013	p_view_SP_script	1	1	9/2/2003 11:	dbo	CREATE PROCEDURE p_view_SP_script
7415776	Ten Most Expensiv	1	1	9/2/2003 11:	dbo	create procedure "Ten Most Expensive
7575777	Employee Sales by	1	1	9/2/2003 11:	dbo	create procedure "Employee Sales by C
7735777	Sales by Year	1	1	9/2/2003 11:	dbo	create procedure "Sales by Year"
8215775	CustOrderHist	1	1	9/2/2003 11:	dbo	CREATE PROCEDURE CustOrderHist @
8375780	SalesByCategory	1	1	9/2/2003 11:	dbo	CREATE PROCEDURE SalesByCategory
1701581	AllSalesByProduct	1	1	9/2/2003 11:	dbo	CREATE PROCEDURE AllSalesByProduct
1701581	AllSalesByProduct	2	1	9/2/2003 11:	dbo	CREATE PROCEDURE AllSalesByProduct
1701581	AllSalesByProduct	3	1	9/2/2003 11:	dbo	CREATE PROCEDURE AllSalesByProduct
1749581	p_ref_objects	1	1	9/2/2003 11:	dbo	CREATE PROCEDURE p_ref_objects @
1765581	p_object_script	1	1	9/2/2003 11:	dbo	CREATE PROCEDURE p_object_script @
1781581	p_display_table_ro	1	1	9/2/2003 11:	dbo	CREATE PROCEDURE p_display_table_
1797581	p_amount_in_worc	1	1	9/2/2003 11:	dbo	CREATE PROCEDURE p_amount_in_wo
1797581	p_amount_in_worc	1	2	9/2/2003 11:	dbo	THEN 'TIGA '
1861581	p_hexdc_to_int	1	1	9/2/2003 11:	dbo	CREATE PROCEDURE p_hexdc_to_int @

Gambar 4.10 Semua skrip stored procedure disimpan dalam table

Sebagai langkah terakhir, buatlah sebuah program yang gunanya untuk menampilkan kembali stored procedure yang disimpan di dalam table SPscript. Hal ini penting jika Anda ingin memperoleh skrip stored procedure untuk versi tertentu.

```
CREATE    PROCEDURE    p_view_SP_script    @sp_name    varchar(50),
@sp_version int AS
/************************************************
Author  : Feri Djuandi
Created : September 2003
Desc. : Display SP text from table
************************************************/
DECLARE @the_text varchar(4000),
 @i int

IF NOT EXISTS( SELECT * FROM SPscript
 WHERE sp_name=@sp_name AND sp_version=@sp_version)
 RETURN

DECLARE cursor_sp INSENSITIVE  CURSOR
FOR SELECT sp_text FROM SPscript
 WHERE sp name=@sp name AND sp version=@sp version
 ORDER BY colid
FOR READ ONLY

OPEN cursor_sp

IF @@CURSOR_ROWS > 0
BEGIN
 SET @i=0
 WHILE @i=0
 BEGIN
  FETCH NEXT FROM cursor_sp INTO @the_text
  SET @i=@@FETCH_STATUS
  IF @i =0
   PRINT @the_text
```

```
  END
END

CLOSE cursor sp

DEALLOCATE cursor_sp
```

Untuk menggunakan program ini, Anda cukup mengirimkan dua buah parameter, yaitu nama stored procedure dan versinya. Contoh:

```
EXEC p_view_SP_script 'AllSalesByProduct',1
```

Selanjutnya, agar versi-versi skrip stored procedure disimpan secara otomatis sebaiknya Anda membuat job schedule yang akan menjalankan program *p_save_SP_script* secara reguler.

Skrip: *4_6.SQL*

4.7 Membuat Skrip SQL INSERT

Jika Anda sedang mencari cara untuk membuat skrip INSERT table, tips ini bisa membantu. Kode program berikut ini dapat membuat sebuah file yang berisi perintah-perintah SQL INSERT untuk semua baris-baris data yang ada di dalam sebuah table. Sebagai contoh, jika Anda memiliki sebuah table bernama *TABLE1* dan berisi data-data:

```
kolom1     kolom2     kolom3
---------- ---------- ----------
amir       100        2001-05-18 13:55:17
wati       98         2003-07-02 03:22:23
iwan       213        2002-12-15 03:10:55
budi       34         2003-01-10 21:12:26
```

maka program ini akan membuat sebuah skrip berisi perintah-perintah:

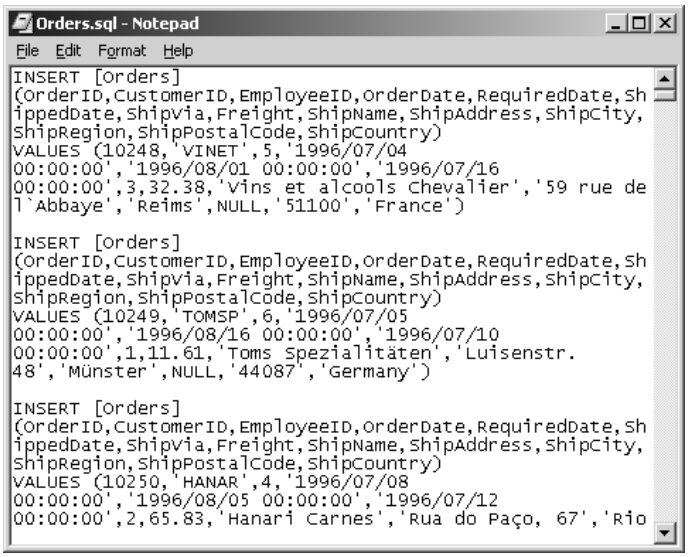
```
INSERT TABLE1 (kolom1,kolom2,kolom3)
VALUES ('amir',100,'2001-05-18 13:55:17')

INSERT TABLE1 (kolom1,kolom2,kolom3)
VALUES ('wati',98,'2003-07-02 03:22:23')

INSERT TABLE1 (kolom1,kolom2,kolom3)
VALUES ('iwan',213,'2002-12-15 03:10:55')

INSERT TABLE1 (kolom1,kolom2,kolom3)
VALUES ('budi',34,'2003-01-10 21:12:26')
```

Program ini ditulis dalam bahasa Visual Basic dan Anda bisa memodifikasinya sesuai dengan keperluan.

```
'************************************************
'Author : Feri Djuandi
'Created : May 2003
'Desc.  : Generate INSERT table SQL script
'************************************************
Option Explicit

Dim objFields As ADODB.Recordset, objRecords As ADODB.Recordset
Dim str_db_parm As String, str_sql_select As String, str_values
As String
Dim i As Integer, j As Integer
Dim str_table As String, str_sql_insert As String, str_col_list
As String
Dim field_value, str_col_type As String

'PLEASE MODIFY these values as you need
str_table = "Orders"
Open "c:\Orders.sql" For Output As #1
str_db_parm = "Driver={SQL
Server};SERVER=.;DATABASE=Northwind;Trusted_Connection=yes"

str_sql_select = "SELECT the_column.colid AS 'column_id', " + _
    "LEFT(the_column.name,50) AS 'column_name', " + _
    "LEFT(col_type.name,15) AS 'column_type', " + _
    "(CASE " +
    "WHEN col type.xtype IN (48, 52, 56, 59, 60, 62, 106, 108,
    122, 127) THEN 'NUMERIC' " +
    "WHEN col_type.xtype IN (58, 61) THEN 'DATETIME' " + _
    "WHEN col_type.xtype IN (167, 175, 231, 239) THEN 'STRING' "
    + _
    "ELSE '' END) AS 'column_xtype', " + _
    "the_column.length AS 'length', " + _
    "the_column.xprec 'precision', " + _
    "the_column.xscale 'scale' " + _
"FROM syscolumns the_column, systypes col_type " + _
"WHERE OBJECT_NAME(id)='" + str_table + "' AND
col_type.xtype=the_column.xtype AND " + _
    "col_type.status IN (0,2)"

Set objFields = CreateObject("ADODB.Recordset")
objFields.CursorLocation = 3
objFields.Open str_sql_select, str_db_parm, 1, 1

str_sql_insert = "INSERT [" + str_table + "] ("
i = 0
str col list = ""
Do While Not objFields.EOF
    i = i + 1
    str_col_list = str_col_list + RTrim(objFields.Fields(1)) +
    ","
    objFields.MoveNext
Loop
```

```
str_col_list = Mid(str_col_list, 1, Len(str_col_list) - 1)
str_sql_insert = str_sql_insert + str_col_list + ")"

Set objRecords = CreateObject("ADODB.Recordset")
objRecords.CursorLocation = 3
objRecords.Open "SELECT " + str_col_list + " FROM " + str_table,
str_db_parm, 1, 1
Do While Not objRecords.EOF
    str_values = "VALUES ("
    objFields.MoveFirst
    For j = 1 To i
        field_value = objRecords.Fields(j - 1)
        If IsNull(field_value) Then
            str_values = str_values + "NULL,"
        Else
            str_col_type = objFields.Fields("column_xtype")
            If str_col_type = "STRING" Then
                str_values = str_values + "'" +
                Replace(Trim(CStr(field_value)), "'", "`") +
                "',"
            ElseIf str_col_type = "DATETIME" Then
                str_values = str_values + "'" +
                Format(field_value, "yyyy/mm/dd hh:mm:ss") +
                "',"
            ElseIf str_col_type = "NUMERIC" Then
                str_values = str_values + CStr(field_value) + ","
            Else
                str_values = str_values + "NULL,"
            End If
        End If
        objFields.MoveNext
    Next
    str_values = Mid(str_values, 1, Len(str_values) - 1) + ")"
    Print #1, str_sql_insert
    Print #1, str_values, vbCrLf

    objRecords.MoveNext
Loop

Close #1

objFields.Close
objRecords.Close

Set objFields = Nothing
Set objRecords = Nothing
```

Contoh isi sebuah file skrip:



Gambar 4.11 Output skrip

Ketika akan membuat perintah SQL, program ini akan memeriksa tipe data dari setiap kolom. Perhatikan variabel *str_sql_select* yang berisi perintah SQL SELECT:

```
SELECT the_column.colid AS 'column_id',
    LEFT(the_column.name,50) AS 'column_name',
    LEFT(col_type.name,15) AS 'column_type',
    (CASE
    WHEN col_type.xtype IN (48, 52, 56, 59, 60, 62, 106, 108,
122, 127) THEN 'NUMERIC'
    WHEN col_type.xtype IN (58, 61) THEN 'DATETIME'
    WHEN col_type.xtype IN (167, 175, 231, 239) THEN 'STRING'
    ELSE '' END) AS 'column_xtype',
    the_column.length AS 'length',
    the_column.xprec 'precision',
    the_column.xscale 'scale'
FROM syscolumns the_column, systypes col_type
WHERE                OBJECT_NAME(id)='Orders'                AND
col_type.xtype=the_column.xtype AND col_type.status IN (0,2)
```

sehingga recordset objFields akan berisi baris-baris data:

```
column_id column_name     column_type column_xtype ...
--------- --------------------------- ------------
```

```
1          OrderID          int          NUMERIC
2          CustomerID       nchar        STRING
3          EmployeeID       int          NUMERIC
4          OrderDate        datetime     DATETIME
5          RequiredDate     datetime     DATETIME
6          ShippedDate      datetime     DATETIME
7          ShipVia          int          NUMERIC
8          Freight          money        NUMERIC
9          ShipName         nvarchar     STRING
10         ShipAddress      nvarchar     STRING
11         ShipCity         nvarchar     STRING
12         ShipRegion       nvarchar     STRING
13         ShipPostalCode   nvarchar     STRING
14         ShipCountry      nvarchar     STRING
```

Program ini hanya dapat menangani tipe-tipe date numerik, string, dan datetime. Tipe-tipe data lainnya seperti text, image, dan binary tidak akan bisa ditangani sehingga pada perintah INSERT, nilai untuk kolom tersebut akan diisi dengan NULL. Ketidakmampuan ini semata-mata disebabkan keterbatasan perintah SQL INSERT untuk tipe-tipe data tersebut. Untuk memasukkan data-data text, image, dan binary, Anda harus menjalankan perintah WRITEXT, bukan perintah INSERT biasa.

Untuk mengetahui semua tipe data yang didukung SQL Server, jalankan perintah ini:

```
SELECT LEFT(name,30),xtype, length, xprec, xscale
FROM systypes WHERE status IN (0,2)
ORDER BY xtype
```

Skrip: *4_7.BAS*